Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних систем та робототехніки

«Затверджую»
в.о. завідуючого кафедри
комп'ютерних систем та робототехніки
_____ к. ф.-м. н., доцент Максим Хруслов
«___» грудня 2024 р.

Пояснювальна записка

до кваліфікаційної роботи
магістра

на тему: «МОДЕЛЬ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ СЕРВОПРИВОДАМИ ПЛАСКОГО МАНІПУЛЯТОРА»

Спеціальність 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Галузь знань 17 – Електроніка, автоматизація та електронні комунікації.
Освітня програма «Комп'ютеризовані системи управління та автоматика»

Захищено на засіданні
Екзаменаційної комісії № 44
протокол № __ від __.12.2024 р.
Оцінка _____ / _____
Голова Екзаменаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
Студент групи КУ– 61
СОЛЯНИК Віктор Андрійович_____

Керівник: к.ф.-м.н., доцент кафедри КСР
КОТВИЦЬКИЙ Альберт Тадеушевич
_____

Рецензент:
к.т.н., доцент; доцент кафедри загальної фізики фізичного факультету ХНУ
ГРЕСЬ Валерія Юріївна
_____

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до магістерської атестаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел і двох додатків. Загальний обсяг роботи складає 77 сторінок, із яких 54 сторіноки основної частини з 37 рисунками, 2 таблицями, списку використаних джерел із 25 найменувань та чотирма додатками.

Метою кваліфікаційної роботи – є візуалізація процесу програмування мікроконтролера для керування сервоприводами за допомогою створення інтерактивного WEB-застосунку, який дозволяє користувачам налаштовувати параметри PWM-сигналів та аналізувати сигнали на виходах мікроконтролера які призводять до зміни положення робочої точки маніпулятора.

Об'єкт дослідження – процес мікроконтролерного керування сервоприводами дволанкового маніпулятора.

Предмет дослідження – методи візуалізації та алгоритми обробки даних для моделювання налаштувань таймера-лічильника мікроконтролера ATmega2560 та їх впливу на положення маніпулятора.

Проблема, яка вирішується в роботі, полягає у створенні інтерактивної платформи для навчання основам мікроконтролерного керування з можливістю моделювання параметрів регістрів, аналізу PWM-сигналів та відображення руху маніпулятора.

Область застосування – навчальні процеси, які передбачають вивчення мікроконтролерного програмування, роботи з сервоприводами та налаштування таймерів у системах автоматизації. Розроблена платформа може використовуватись у навчальних закладах, на тренінгах і для самостійного вивчення студентами та розробниками.

Ключові слова: мікроконтролер, таймер-лічильник, ATmega2560, сервопривод, плаский маніпулятор, PWM-сигнал, візуалізація, навчальна платформа, регістр, інтерфейс, React, JavaScript, WebStorm.

ABSTRACT

The explanatory note for the master's thesis consists of an introduction, three chapters, conclusions, a list of references, and two appendices. The total volume of the work is 77 pages, including 54 pages of the main text, with 37 figures, 2 tables, a reference list of 25 sources, and four appendices.

The aim of the qualification work is to visualize the process of microcontroller programming for servo control by developing an interactive WEB application that allows users to configure PWM signal parameters and analyze the output signals of the microcontroller, which result in changes to the manipulator's end-effector position.

The object of the research is the process of microcontroller control of servos in a two-link manipulator.

The subject of the research is the visualization methods and data processing algorithms for modeling the settings of the timer-counter of the ATmega2560 microcontroller and their impact on the manipulator's position.

The problem addressed in the work lies in the creation of an interactive platform for teaching the fundamentals of microcontroller control, with the ability to model register parameters, analyze PWM signals, and visualize the manipulator's motion.

The application area includes educational processes focused on studying microcontroller programming, working with servos, and configuring timers in automation systems. The developed platform can be used in educational institutions, training sessions, and for self-study by students and developers.

Keywords: microcontroller, timer-counter, ATmega2560, servo, flat manipulator, PWM signal, visualization, educational platform, register, interface, React, JavaScript, WebStorm.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

CTC – Clear To Compare;

IDE – Integral Development Environment (інтегроване середовище розробки);

PDM – Pulse Duration Modulation;

PWM – Pulse Width Modulation (широтно-імпульсна модуляція).

RC – Radiocontrol;

SD – Servo drive (сервопривод);

TC – Timer/Counter (таймер-лічильник);

ШІМ – Широтно-імпульсна модуляція

ВСТУП

Актуальність роботи У сучасному світі автоматизація процесів і впровадження робототехнічних систем стали невід'ємною частиною промисловості, побуту й освітньої сфери. Одними із ключових компонентів таких систем є сервоприводи, які забезпечують точне позиціонування і контроль руху. Для ефективного використання сервоприводів необхідно вміти програмувати мікроконтролери, що дозволяє керування різними параметрами, такими як частота і ширина імпульсів PWM-сигналу.

Проте, програмування мікроконтролерів часто сприймається студентами та початківцями як складний процес через абстрактність команд, необхідність розуміння низькорівневих параметрів і відсутність візуального зворотного зв'язку. Візуалізація процесу налаштування параметрів і роботи сервоприводів може значно спростити навчання, дозволяючи наочно демонструвати вплив змін на поведінку пристрою.

У зв'язку з цим особливого значення набуває створення інтерактивних програмних інструментів, які не лише автоматизують процес керування, але й забезпечують візуалізацію цього процесу.

Таким чином, дослідження, спрямоване на створення моделі керування сервоприводами, що візуалізує процес програмування, є актуальним у контексті підвищення якості освітніх методик, розвитку інтерактивного навчання та вдосконалення підходів до роботи з робототехнічними системами.

Метою дослідження є візуалізація процесу програмування мікроконтролера для керування сервоприводами за допомогою створення інтерактивного WEB-застосунку, який дозволяє користувачам налаштовувати параметри PWM-сигналів та аналізувати сигнали на виходах мікроконтролера які призводять до зміни положення робочої точки маніпулятора.

Об'єкт дослідження – процес мікроконтрольного керування сервоприводами дволанкового маніпулятора.

Предмет дослідження – WEB-застосунок, що забезпечує візуалізацію програмування мікроконтролера ATmega2560, налаштування параметрів PWM-сигналів і візуальне відображення положення дволанкового маніпулятора та його роботи.

Методи дослідження: аналіз літературних джерел з теорії роботи сервоприводів та мікроконтролерів; методи системного аналізу для розробки апаратної частини керування сервоприводами; методи об’єктно-орієнтованого проєктування для створення WEB-застосунку; методи експериментального моделювання для перевірки роботи сервоприводів; методи візуалізації даних для відображення результатів роботи сервоприводів і маніпулятора.

Завдання дослідження

1. Провести аналіз літературних джерел для визначення класифікації сервоприводів, їх основних характеристик та методів керування за допомогою мікроконтролерів.
2. Виконати огляд існуючих програмних рішень для моделювання процесів програмування мікроконтролерів і керування сервоприводами.
3. Обґрунтувати вибір мікроконтролера ATmega2560 для реалізації задачі керування сервоприводами дволанкового маніпулятора.
4. Розробити апаратну схему підключення двох сервоприводів до одного 16-бітного таймера мікроконтролера.
5. Розробити WEB-застосунок.
6. Провести тестування розробленого WEB-застосунку з демонстрацією роботи дволанкового маніпулятора.
7. Виконати аналіз результатів тестування для оцінки ефективності створеного рішення.

РОЗДІЛ 1

АНАЛІЗ ЛІТЕРАТУРИ ТА ІСНУЮЧИХ ПРОГРАМНИХ РЕАЛІЗАЦІЙ

1.1 Сервоприводи: визначення, класифікація та основні характеристики.

1.1.1 Поняття сервопривода та його поширеність

Сервоприводи (SD) складаються з чотирьох основних компонентів: керуючої плати, елементів зворотного зв’язку, двигуна та редуктора див. рис. 1.1 і зазвичай мають щонайменше три виводи: плюс живлення, земля та керуючий сигнал [1]. Перші два виводи використовуються для подачі живлення, тоді як на третій передаються керуючі сигнали. Усі SD підтримують PWM (широтно-імпульсну модуляцію) або Pulse Duration Modulation (PDM), а сучасні цифрові моделі також оснащені послідовним інтерфейсом передачі даних, що дозволяє не лише керувати сервоприводом, але й отримувати зворотний зв’язок. Деякі моделі мають п’ять виводів, які різні виробники використовують для додаткового живлення двигуна, передачі зворотного зв’язку або підключення програматорів.

Завдяки універсальності та компактності, сервоприводи широко використовуються в різних галузях [2]:

- Робототехніка: управління рухомими частинами роботів.
- Авіамоделювання та моделювання: дрони, моделі літаків і автомобілів.
- Промисловість: точне управління виробничими механізмами та обладнанням.
- RC-проєкти: завдяки доступності малогабаритних моделей, сервоприводи активно використовуються аматорами для самостійних розробок.

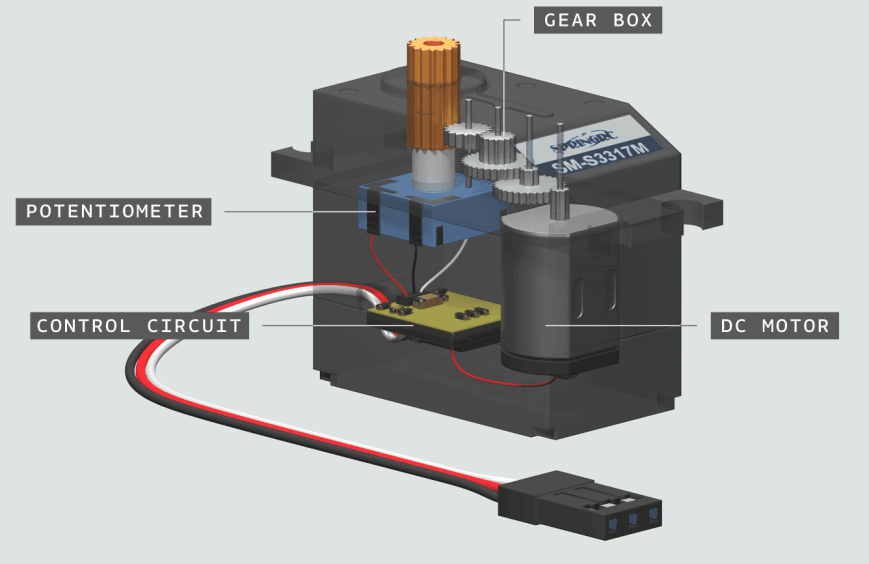

Рисунок 1.1 – Елементи сервопривода: potenciometer – елемент зворотнього з’вязку, gear box – редуктор, control circuit – керуюча плата, DC motor – мотор

Особливістю сервоприводів є їх інтеграція в системи, де необхідна точність, надійність і стабільність. Сучасні цифрові моделі дозволяють реалізовувати складні завдання за рахунок високої швидкодії та можливості отримання зворотного зв’язку.

2.1.2 Основні параметри сервоприводів

Сервоприводи представлені широким спектром моделей, що відрізняються низкою ключових характеристик. Основні параметри, які використовуються для опису їх роботи, включають [3]:

- Частоту роботи (Гц);
- Ширину імпульсу (мс);
- Швидкість обертання (с/60°);
- Максимальний кут обертання (°);
- Мертву зону (мкс);
- Напругу живлення (В);
- Обертаючий момент (кгF·см);

- Габарити (мм × мм × мм);
- Маса (кг).

Частота роботи характеризує швидкість оновлення даних щодо положення валу сервоприводу, тобто PWM-сигналу. Наприклад, для багатьох сервоприводів ця частота становить 50 Гц , що відповідає одному імпульсу кожні 20 мс. Однак основним параметром залишається тривалість імпульсу, а не частота. Також вважається, що SD працюють за технологією PWM, але це фактично PDM (Pulse Duration Modulation).

Ширина імпульсу визначає тривалість керуючого сигналу, що надходить на сервопривід, і зазвичай вказується у трьох значеннях: мінімальна, нейтральна та максимальна. Наприклад, для сервопривода SG92R з кутом обертання 180° мінімальна ширина (544 мкс) відповідає куту 0°, нейтральна (1500 мкс) – 90°, а максимальна (2400 мкс) – 180° [4]. Імпульси з тривалістю за межами цього діапазону ігноруються.

Швидкість обертання визначає час, необхідний для повороту валу сервоприводу на 60°. Цей показник залежить від напруги живлення і може вказуватися як діапазон.

Максимальний кут обертання відображає найбільший кут повороту валу, який може досягти сервопривід.

Мертва зона вказує мінімальний приріст тривалості імпульсу, що не впливатиме на роботу. Цей параметр забезпечує уникнення дрібних коливань валу поблизу заданого положення, що сприяє зниженню шуму і підвищенню стабільності.

Напруга живлення зазвичай задається у вигляді діапазону і визначає рівень напруги, необхідний для роботи сервоприводу.

Обертаючий момент вказує на максимальне зусилля, яке може створити сервопривід, зазвичай виражене в кгF·см. Наприклад, значення 2.5 кгF·см означає, що сервопривід може утримувати масу до 2.5 кг на плечі довжиною 1 см або 1.25 кг на плечі 2 см. Деякі виробники використовують

позначення кг/см, що може призводити до непорозумінь. У нашому випадку правильна розмірність обертаючого моменту – Н·см. Проте замість ньютонів часто застосовують одиницю кілограм-сила (кгF), яка враховує стандартне прискорення вільного падіння, що становить 9,80665 м/с². Таким чином, при вимірюванні сили змінною є лише маса, звідки й походить ця назва. Застосовуючи це до формули для обчислення обертаючого моменту, отримуємо значення у форматі кгF·см.

Габарити та маса відображають фізичні характеристики сервоприводу і важливі при виборі пристрою для конкретного застосування.

2.1.3 Класифікація сервоприводів

Сервоприводи класифікуються за низкою ознак:

За типом керування:

- PDM (Pulse Duration Modulation): найпоширеніший метод керування, що дозволяє задавати положення валу або швидкість [5];
- Послідовні команди: дозволяють передавати інформацію через UART, SPI чи I2C.

За конструкцією:

- Аналогові: прості у використанні, підтримують частоту до 50 Гц;
- Цифрові: високоточні, працюють на частотах 200 Гц і більше [6].

За матеріалом редуктора [7]:

- Пластикові – дешеві, але менш довговічні;
- Карбонові – міцніші, ніж пластикові, але мають схожі обмеження;
- Металеві – надійні та довговічні, використовуються для великих навантажень.

За розміром:

- **Мікро:** легкі (до 40 г), використовуються в моделях літаків та дронів;

- **Стандартні:** універсальні, маса 40–80 г;
- **Гігантські:** маса понад 100 г, застосовуються у промисловості.

Сервоприводи є важливими елементами сучасних систем автоматизації завдяки своїй точності, універсальності та широкому спектру характеристик (див. табл. 1.1). Їхня класифікація дозволяє обрати оптимальну модель для конкретного завдання.

Таблиця 1.1

Таблиця різноманітних сервоприводів та їх основних параметрів

Назва	Частота, Гц	Ширина імпульсу, мс	Швидкість, s/60 °	Обертаючий момент, $кг_F$ * см	Мертва зона, мкс	Кут, °
HS-125MG	50	0.9-2.1	0.17 (4.8В) 0.13 (6В)	3	5	180
SG92R	50	1-2	0.1 (4.8В)	2.5	10	180
FrSky Xact BLS5401H	333	1-2	0.05 (6.0В) 0.03 (9.0В)	17.2 (6.0В) 22.8 (9.0В)	4	360
Power HD Servo LF-20MG	50	0.8-2.2	0.18 (4.8В) 0.16(6.6В)	16.5	2	180
HS-805BB	333	0.9-2.1	0.19 (4.8В) 0.14 (6В)	19.8 (4.8В) 25.7 (6В)	8	199
MG90S	50	1-2	0.1 (4.8В), 0,08(6В)	1.8	5	180
SO5NF	50	1-2	0.2 (4.8В) 0.18 (6В)	2.8 (4.8В) 3.2 (6В)	10	160

1.2 Існуючі підходи до керування сервоприводами за допомогою мікроконтролерів

1.2.1 Параметри мікроконтрольного керування сервоприводом

Сервоприводи є ключовим компонентом багатьох автоматизованих систем. Їх керування здійснюється через мікроконтролер, котрий надсилає сигнали керування у вигляді широтно-імпульсної модуляції (PWM) або модуляції тривалості імпульсу (PDM).

Мікроконтролери, що мають вбудовані таймери, підтримують PWM та дозволяють генерувати сигнали з різною тривалістю імпульсу. Де тривалість імпульсу встановлює певне положення вихідного валу SD:

- Тривалість імпульсу визначає кут повороту валу. Наприклад, імпульс 1500 мкс задає нейтральне положення (90°), імпульс 1000 мкс – мінімальне положення (0°), а 2000 мкс – максимальне (180°);
- PWM сигнал прийнято надсилати із частотою 50 Гц, що відповідає періоду 20 мс.

Кожен SD підтримує PDM, що значить що можна надсилати імпульси із частотою 200–333 Гц, доки період не стане перевищувати чутливість зворотнього зв’язку SD.

Мікроконтролери забезпечують зручні інструменти для генерації PWM-сигналів завдяки вбудованим таймерам, котрі, на відміну від програмної генерації, не залежать від плину програми. Тобто таймери забезпечують апаратну генерацію PWM. Налаштування таймеру, та відповідних його пінів відбувається так::

1. Налаштування таймера мікроконтролера:
    - Вибір режиму роботи (Fast PWM, Phase Correct PWM);
    - Встановлення дільника тактової частоти для формування необхідної частоти сигналу.

2. Регулювання ширини імпульсу на спеціальних виходах:
    - Завантаження значення до відповідного регістра для встановлення необхідного положення валу сервопривода.

1.2.2 Особливості підходів до керування

Використання спеціальних виходів (каналів таймеру)–головне обмеження у кількісті доступних виходів із підтримкою PWM [8].

Використання спеціалізованих драйверів – наприклад, драйвери на базі PCA9685, які дозволяють керувати багатьма сервоприводами через єдиний послідовний інтерфейс. Його недолік полягає у розпаралелювальнілише одного каналу, тобто для точного керування багатьма SD всеодно знадобиться багато каналів.

Послідовний інтерфейс керування – використовується в цифрових сервоприводах для передачі даних через UART, I2C чи SPI. Забезпечує не тільки передачу команд, а й зворотний зв’язок від сервоприводів. Він часто використовується, через швидкодію сучасних мікроконтролерів. Але він повністю програмно залежний, тобто для надточних обчислень не підійде.

1.2.3 Переваги та недоліки обраного підходу

Для даної роботи було обрано використання каналів, оскільки для візуалізації буде достатньо двох SD маніпулятора, тобто двох точних каналів. У більшості мікроконтролерів є таймери від 2-х каналів.

Переваги:

- Наочність використання каналів – оскільки для двох з трьох методів необхідно знати як налаштувати канали;
- Висока гнучкість у налаштуванні параметрів сигналу – багато різних режимів роботи у купі із кількістю режимів каналів надають можливість зробити сигнал бажаної тривалості різними способами.

Недоліки:

- Обмежена кількость PWM-виходів мікроконтролера;
- Залежність від таймерів.

1.3 Аналіз популярних середовищ моделювання для роботи з мікроконтролерами та сервоприводами

Середовище моделювання є важливим інструментом для проєктування та тестування робототехнічної системи. Вона дозволяює створювати віртуальні схеми, програмувати мікроконтролери та симулювати їх роботу без необхідності використовувати фізичне обладнання. Проте майже всі доступні платформи зосереджені на інтерфейсному або програмному керуванні, у той час як мікроконтрольне керування, яке передбачає налаштування регістрів керування мікроконтролера, реалізується дуже обмежено або взагалі відсутнє.

Розглянемо популярні середовища та покажемо їх переваги і недоліки. А також визначимо, порівнявши із даною роботою, головну відмінність реалізуємого підходу.

1.3.1 Пакет програм для автоматизованиого проектування Proteus

Proteus – це професійне програмне забезпечення для моделювання електронних схем та їх тестування в умовах, наближених до реальних [9]. Його широко використовують інженери, студенти та розробники для створення, перевірки та оптимізації електронних пристроїв.

Переваги:

- Багата бібліотека компонентів, зокрема мікроконтролерів і сервоприводів;
- Симуляція реальних умов роботи за допомогою завантаження коду в мікроконтролер;

Візуалізація сигналів на виходах GPIO.

Обмеження:

- Налаштування регістрів мікроконтролера не інтерактивне – акцент зроблений на завантаження готового коду;

- Ліцензія є платною;
- Результат залежить від машини, на котрій запущена програма.

1.3.2 Онлайн-платформа Wokwi

Wokwi – це сучасна онлайн-платформа для моделювання мікроконтролерів та їх периферійних пристроїв [10]. Вона створена для розробників, студентів та ентузіастів, які хочуть швидко тестувати свої ідеї без потреби у фізичному обладнанні.

Переваги:

- Доступність через браузер, безкоштовний доступ;
- Підтримка популярних мікроконтролерів, таких як Arduino і ESP32;
- Простота роботи з інтерфейсними командами для керування сервоприводами.

Обмеження:

- Орієнтація на програмне керування, без інтерактивного налаштування регістрів;
- Лімітована бібліотека компонентів порівняно з професійними платформами.

1.3.3 Онлайн-платформа TinkerCAD

TinkerCAD – це безкоштовна онлайн-платформа, розроблена компанією Autodesk, яка дозволяє моделювати електронні пристрої, створювати 3D-моделі та симулювати роботу мікроконтролерів [11]. Платформа орієнтована на освітні цілі та початківців у галузі програмування й електроніки.

Переваги:

- Простий і зрозумілий інтерфейс;
- Можливість базового програмування мікроконтролерів Arduino для роботи із сервоприводами.

Обмеження:

- Відсутність підтримки налаштування регістрів мікроконтролера;
- Обмежені можливості для моделювання складних систем.

1.3.4 Графічне середовище програмування для моделювання і симуляції MATLAB/Simulink

MATLAB/Simulink – це програмний комплекс, розроблений компанією MathWorks, який використовується для математичного моделювання, симуляції динамічних систем та автоматизації інженерних розрахунків [12]. MATLAB забезпечує програмування математичних моделей, а Simulink – їх графічну симуляцію. Переваги:

- Потужні інструменти для аналізу систем та оптимізації;
- Можливість моделювання роботи сервоприводів і динамічних систем.

Обмеження:

- Відсутність прямого доступу до налаштування регістрів мікроконтролера;
- Висока вартість ліцензії.

1.3.5 Мікроконтрольне керування

На відміну від розглянутих середовищ, які акцентуються на завантаженні програмного коду або простих інтерфейсних командах, дана робота орієнтована на мікроконтрольне керування. Цей підхід включає:

- Інтерактивне налаштування регістрів мікроконтролера (наприклад, TCCR1A, TCCR1B);
- Вибір режимів роботи таймера, дільника частоти та конфігурації сигналів на виходах;
- Візуалізацію впливу змін параметрів регістрів на роботу сервоприводів у реальному часі.

Щоб побачити суттєву різницю від інших платформ, зробимо порівняльну таблицю перерахованих середовищ (див. табл. 1.2).

Таблиця 1.2

Таблиця для порівняння існуючих застосунків

Середовище	Тип керування	Цільова аудиторія	Ліцензія
Proteus	Програмне	Професіонали	Платна
Wokwi	Інтерфейсне, програмне	Студенти	Безкоштовна
TinkerCAD	Інтерфейсне	Студенти	Безкоштовна
MATLAB/ Simulink	Програмне	Професіонали	Платна
WEB-застосунок даної роботи	Мікроконтролерне	Студенти	Безкоштовна

1.4 Порівняння мікроконтролерів для керування сервоприводами

Сучасний ринок пропонує широкий вибір мікроконтролерів, кожен із яких має свої переваги та недоліки. Для цієї роботи знадобиться обрате, який саме обрати мікроконтролер, для котрого основними критеріями є:

- Кількість таймерів та їх функціональність;
- Кількість апаратних PWM-виходів;
- Доступність і простота інтеграції.

Для порівняння оберемо сучасні найпоширеніші мікроконтролери [13]:

1. STM32 – це серія 32-бітних мікроконтролерів на базі архітектури ARM Cortex-M [14].

   Основні переваги**:**

   - Висока продуктивність: Тактова частота від 48 до 216 МГц, що дозволяє реалізовувати складні алгоритми в реальному часі;

- Розширений функціонал: Велика кількість периферійних пристроїв (таймери, UART, SPI, I2C, CAN тощо);
- Гнучкість PWM: STM32 забезпечує підтримку до 16 апаратних каналів PWM, залежно від моделі;
- Енергозбереження: Мікроконтролери цієї серії мають кілька режимів роботи з низьким енергоспоживанням.

Недоліки:

- Висока складність налаштування регістрів через велику кількість функцій та надбудов;
- Необхідність використання спеціалізованих середовищ розробки (CubeMX, Keil, STM32CubeIDE).

STM32 ідеально підходить складних завдань із високими вимогами до продуктивності, але є менш зручним для початківців.

2. ESP32 – це мікроконтролер із вбудованою підтримкою бездротових технологій (Wi-Fi та Bluetooth), розроблений компанією Espressif Systems [15].
3. Основні переваги:
    - Інтеграція Wi-Fi та Bluetooth: Підходить для IoT-проєктів, що потребують підключення до мережі;
    - Висока тактова частота: До 240 МГц, що забезпечує виконання складних обчислень;
    - PWM-модуляція: Підтримка до 16 каналів для керування периферійними пристроями;
    - Доступність і низька ціна: ESP32 є одним із найбільш економічних мікроконтролерів у своїй категорії.

Недоліки:

- Відсутність великої кількості таймерів (зазвичай два);
- Орієнтованість на мережеві функції більше, ніж на низькорівневе налаштування периферії.

ESP32 підходить для IoT-проєктів, але його спеціалізація ускладнює процес керування регістрами напряму.

4. Microchip – це широка лінійка мікроконтролерів, яка включає AVR і PIC серії [16].

Основні переваги:

- Широка підтримка периферійних пристроїв: AVR серія підтримує кілька таймерів (до 6) та до 15 PWM-виходів.
- Простота використання: ATmega-серія має широку підтримку в середовищах розробки (наприклад, Arduino IDE).
- Велика спільнота користувачів: Доступність численних прикладів і бібліотек для швидкого старту.
- Надійність: Висока стабільність роботи навіть у складних умовах.

Недоліки:

- Нижча тактова частота (до 16 МГц у AVR серії).
- Обмежений функціонал порівняно з STM32 та ESP32.

Microchip є оптимальним вибором для проєктів, де потрібне зручне мікроконтрольне налаштування регістрів.

Для реалізації задачі мікроконтрольного керування сервоприводами було обрано мікроконтролер ATmega2560 через такі властивості:

- ATmega2560 має 15 апаратних каналів PWM, що дозволяє одночасно керувати декількома сервоприводами без необхідності використання зовнішніх драйверів.
- Шість таймерів (два 8-бітних і чотири 16-бітних) забезпечують гнучке налаштування частоти та тривалості імпульсів.
- Простота інтеграції – ATmega2560 підтримується численними бібліотеками та середовищами розробки, такими як Arduino IDE, що полегшує створення і тестування проєкту.

- Доступність – цей мікроконтролер широко доступний і має розвинену спільноту користувачів, що полегшує вирішення можливих проблем під час розробки.

ATmega2560 поєднує в собі необхідний функціонал і зручність для реалізації мікроконтрольного керування сервоприводами. Він дозволяє налаштовувати регістри так, як вони реалізовані на апаратному рівні, за допомогою бітів, створювати точні PWM-сигналів і підтримує під'єднання до трьох SD на один таймер. Найголовніше те, що даний мікроконтролер дуже доступний та має таймер-лічильник, за допомогою котрого можна зручно керувати пласким маніпулятором.

1.5 Таймер-лічильник мікроконтролера ATmega2560

Таймер-лічильник є ключовим елементом периферії мікроконтролерів, який використовується для формування сигналів PWM, керування перериваннями у реальному часі та обробки зовнішніх сигналів [17]. МікроконтролерATmega2560 має шість таймерів: два 8-бітних (TC0 і TC2) та чотири 16-бітних (TC1, TC3, TC4, TC5) [18]. Вони забезпечують генерування точних, програмно-незалежних ШІМ-сигналів.

Для роботи було обрано перший ТС із таких причин:

- Кількість виходів – TC1 підтримує три канали порівняння (A, B, C), що дозволяє одночасно формувати PWM-сигнали для трьох сервоприводів;
- Кількість режимів – TC1 підтримує всі стандартні режими роботи: Normal, CTC, Fast PWM, Phase Correct PWM;
- Широкий діапазон частот – використання дільників тактової частоти дозволяє отримати необхідну частоту для PWM-сигналів (наприклад 50 Гц для сервоприводів) при цьому можна обрати часовий крок для визначення тривалості;
- Зручність налаштування – таймер має окремий регістр для встановлення верхньої межі лічильника (ICR1), що дає змогу

точно визначати період сигналу. Окремі регістри порівняння (OCR1A, OCR1B, OCR1C) дозволяють задавати поведінку імпульсу для кожного виходу.

Висновки за розділом 1

У цьому розділі було проаналізовано роботу сервоприводів, можливості керування ними через мікроконтролери та роль таймерів у цьому процесі. Та зроблено висновки:

1. Сервоприводи є ключовими елементами для точного керування рухом маніпуляторів. Їх робота базується на формуванні PWM-сигналів, що визначають положення валу.
2. Таймери мікроконтролера є основним інструментом для генерації таких сигналів. Обраний TC1 ATmega2560 забезпечує необхідну точність та різноманітність завдяки трьом незалежним виходам та можливостям налаштувати режим роботи та поведінку виходів.
3. Виявлено, що більшість існуючих рішень зосереджуються на інтерфейсному або програмному керуванні, тоді як мікроконтрольне керування через регістри забезпечує більше можливостей для дослідження та налаштування роботи маніпулятора.

Ці результати створюють основу для реалізації візуалізації процесу мікроконтрольного керування пласким маніпулятором та апаратної частини.

РОЗДІЛ 2

РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ КЕРУВАННЯ СЕРВОПРИВОДАМИ ПЛАСКОГО МАНІПУЛЯТОРА

2.1 Вибір компонентів для системи керування

Для розробки системи керування сервоприводами було обрано такі компоненти:

Плата розробника MEGA2560 – обрана через її доступність, популярність серед розробників і сумісність із мікроконтролером ATmega2560. Ця плата є готовим модулем, який не потребує пайки, що спрощує процес під'єднання до електричної схеми. Мікроконтролер ATmega2560 також має децілька вбудованих елементів, що покращують роботу із мікроконтролером. Зокрема виходи для програматора.

Сервоприводи SG92R – ці серводвигуни вибрано через їхню поширеність та росповсюдженим характеристикам. Вони забезпечують достатній обертаючий момент для роботи з легким дволанковим маніпулятором.

Блок живлення RD DPS5020 Power Supply 50V 20A – абезпечує стабільне живлення для сервоприводів. Надає можливість обрати напргугу та обмежити струм, що захищає компоненти від перевантаження.

Програматор AVRISP mkII – необхідний для завантаження програми до мікроконтролера. Повністю сумісний із інтегрованим середовищем розробки Microchip Studio [19].

Логічний аналізатор Saleae Logic 16– використовується для перевірки правильності формування PWM-сигналів, тобто забезпечує візуалізацію сигналів на етапі тестування [20].

Макетна плата SYB-120 та з'єднувальні джампери – використовуються для зручності з'єднання компонентів системи. Через макетну плату

здійснюється підключення живлення до сервоприводів та зняття сигналів керування до логічного аналізатора.

Інтегроване середовище розробки Microchip Studio – обрано через його сумісність із ATmega2560 та підтримку програматорів від Microchip.

Ці компоненти, див. рис. 2.1-2.3 є основою системи керування сервоприводами. Вибір компонентів базувався на їхній доступності, функціональності та відповідності вимогам проєкту.

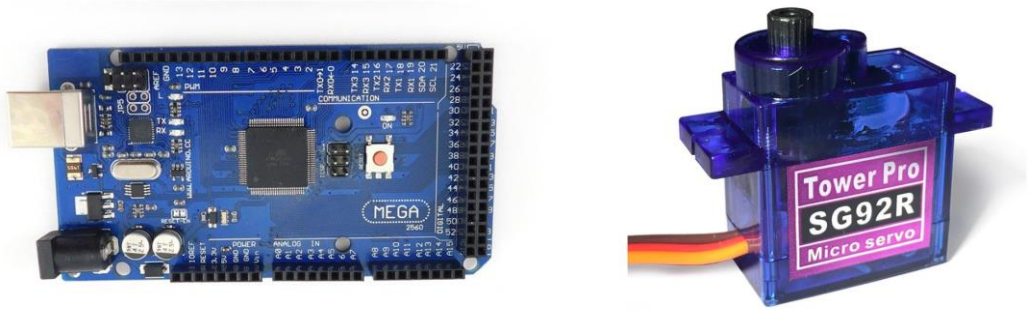

Рисунок 2.1 – Плата розробника MEGA2560 зліва та SD EMAX справа

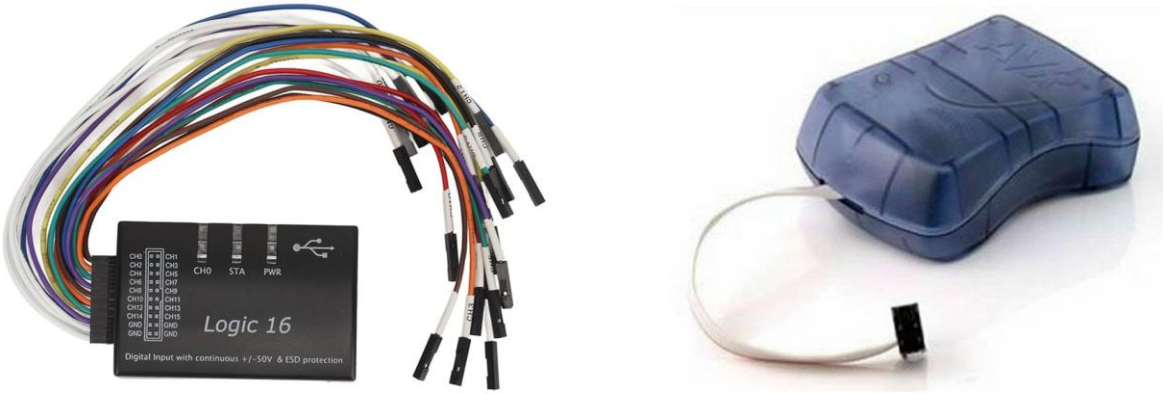

Рисунок 2.2 – Логічний аналізатор зліва та програматор справа

Рисунок 2.3 – Одноканальний лабораторний блок живлення

2.2 Схема підключення двох сервоприводів дволанкового маніпулятора до першого 16-бітного таймер-лічильника

Для живлення сервопривода (SD) необхідно подавати +5В від зовнішнього блоку живлення, тоді як плата розробника отримує живлення через інший блок живлення +12В. Важливо об’єднати землю плати та блоку живлення для забезпечення спільного потенціалу для запобігання шумів, див. рис. 2.4.

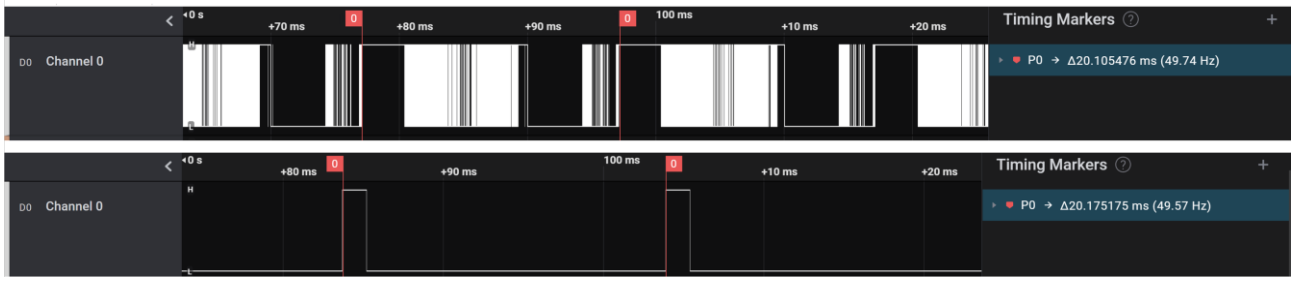

Рисунок 2.4 – Приклад виду сигналу без заземлення, згори, та із заземленнямм, внизу

Живлення SD слід здійснювати виключно від зовнішнього джерела, а не від плати розробника, оскільки остання не розрахована на забезпечення струмів, необхідних для роботи двигунів. Наприклад, максимальний струм, який може забезпечити плата розробника, обмежений стабілізатором напруги до 800 мА.

Із зазначеними застереженнями та правилами, розроблено схему підключення усіх необхідних елементів, див. рис. 2.5 та приблизна схема на макетній платі, див. рис. 2.6. На ньому видно, що плата та сервопривід під’єднані у два різних кола, при цьому зі спільною землею. Також на схемі зображено два інформаційні виходи: саме на цих виходах мікроконтролер буде генерувати імпульс необхідної довжини із певною частотою, котрий буде оброблюватись як сервоприводом, так і логічним аналізатором.

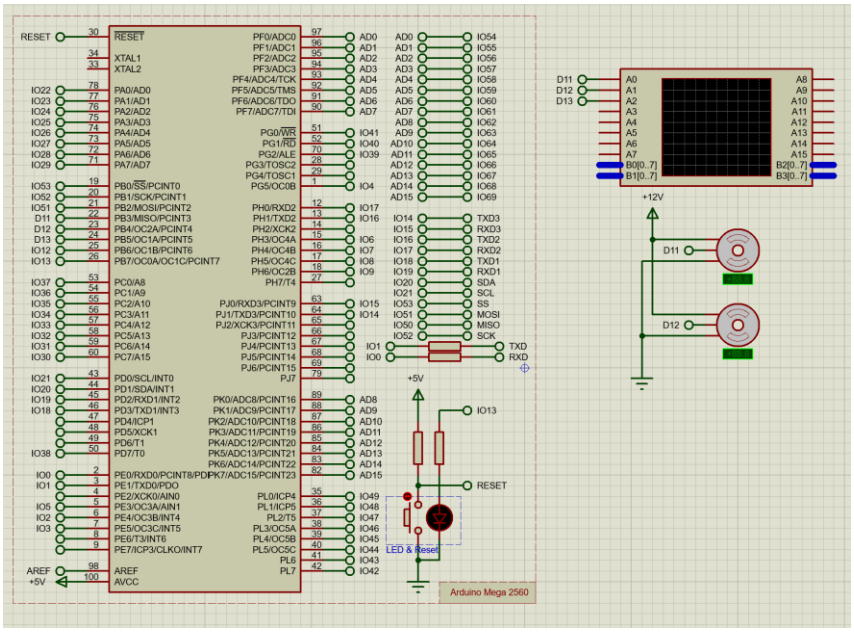

Рисунок 2.5 – Схема під’єднань для керування двома сервоприводами та підключенням логічного анаалізатору

Самі сервоприводи під’єднані до двох виходів мікроконтролера, а саме до OC1A(або OC1C) та OC1B. У документації ці виходи представлені портом B з 4 по 6 біт, відповідно [18]. На платі розробника це виходи D11-D13, див. рис. 2.7.

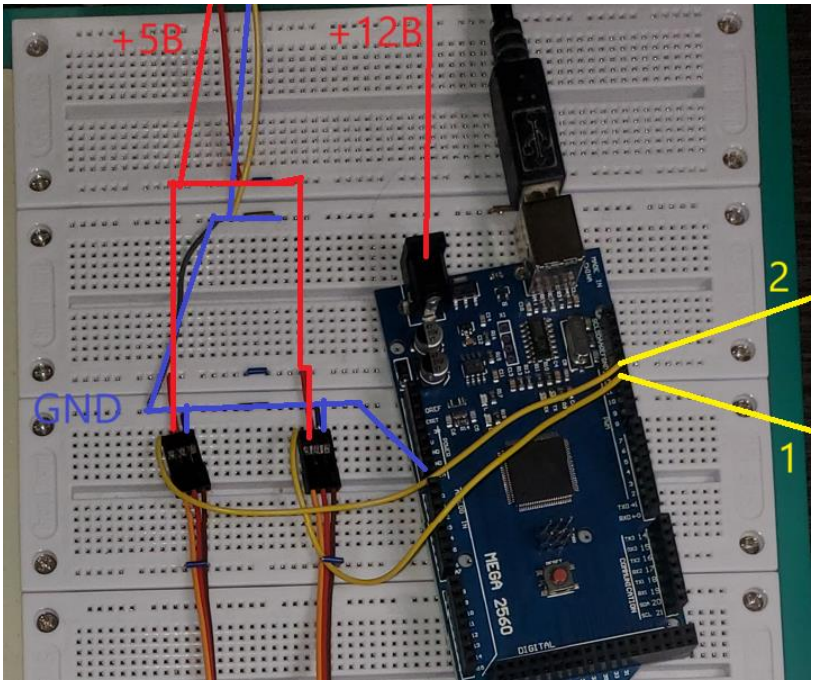

Рисунок 2.6 – Схема підключення: червоний – живлення, синій – спільна земля, жовтий – інформаційні виходи

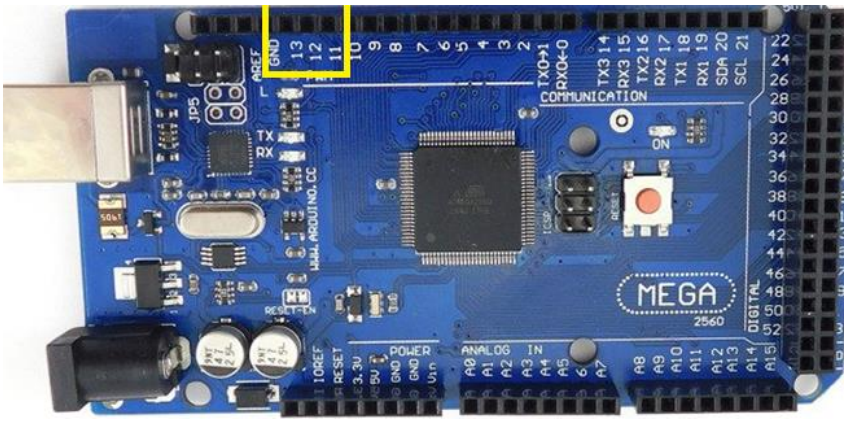

Рисунок 2.7 – Плата розробника із виділеними виходами першого таймера

2.3 Апаратна реалізація

Апаратна реалізація схеми підключень зображена на рис. 2.8. А два сервоприводи, що під’єднані до OC1A та OC1B пінів, представлені у вигляді двох ланок плаского маніпулятора, див. рис. 2.9.

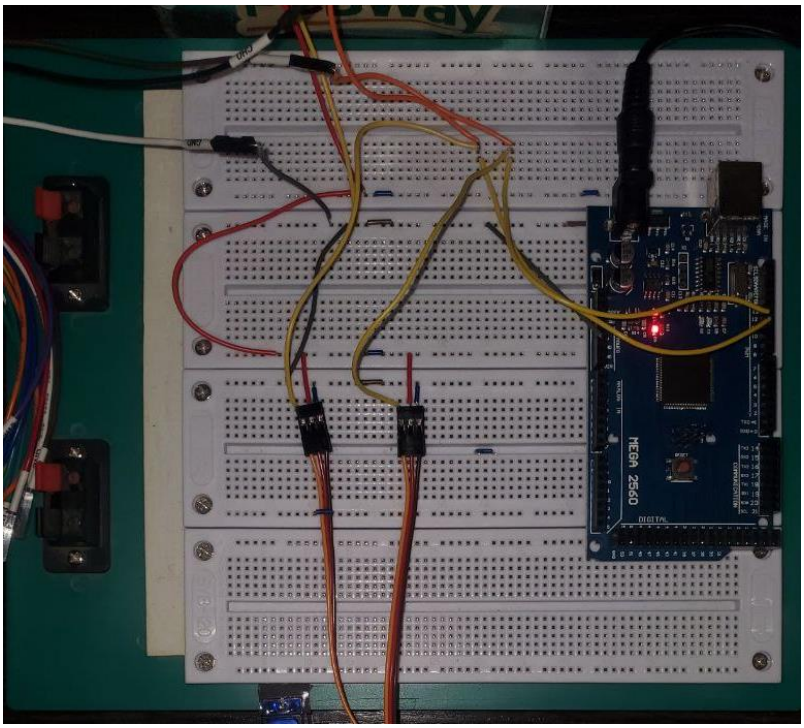

Рисунок 2.8 – Підключення усіх елементів за схемою

Рисунок 2.9 – Фізична модель плаского маніпулятора

Висновки за розділом 2

Апаратна система керування сервоприводами на базі таймера TC1 мікроконтролера ATmega2560 була успішно реалізована. Система забезпечує стабільну роботу двох сервоприводів, точне керування положенням маніпулятора.

РОЗДІЛ 3

РОЗРОБКА НАВЧАЛЬНОЇ ПЛАТФОРМИ ДЛЯ ВІЗУАЛІЗАЦІЇ ПРОЦЕСУ МІКРОКОНТРОЛЕРНОГО КЕРУВАННЯ ПЛАСКИМ МАНІПУЛЯТОРОМ

3.1 Вибір інстурментів розробки

Створення навчальної платформи для візуалізації процесу мікроконтролерного керування вимагає використання сучасних технологій, які забезпечують не тільки високу продуктивність, але й простоту розуміння та інтерактивність для користувачів. Важливим аспектом було обрання інструментів, що дозволяють ефективно розробити веб-застосунок, який симулює роботу таймера-лічильника мікроконтролера ATmega2560 і наочно демонструє принципи керування сервоприводами.

При виборі інструментів враховувалися такі критерії:

Доступність і популярність: Інструменти повинні мати широкий набір навчальних матеріалів і документацію.

Простота інтеграції: Забезпечення сумісності компонентів між собою.

Зручність розробки: Наявність функцій для зменшення витрат часу на налагодження та тестування.

Інтерактивність: Можливість створення зручного інтерфейсу для користувача.

На основі цих критеріїв було обрано середовище розробки WebStorm, мову програмування JavaScript, фреймворк React та бібліотеку react-chartjs-2. Далі описано їх переваги та обґрунтовано вибір кожного інструмента.

1.3.1 Інтегроване середовище розробки WebStorm

WebStorm від компанії JetBrains є потужним інструментом для створення JavaScript-застосунків [21]. Його обрання обумовлене такими перевагами:

- Підтримка JavaScript і React: Завдяки вбудованим функціям, WebStorm забезпечує високу швидкість розробки інтерфейсів.
- Автоматизація рутинних задач: Інструмент має функції автоматичного форматування коду, підсвічування синтаксичних помилок і швидкого навігації по проектах.
- Інтеграція з системами контролю версій: Підтримка Git і GitHub спрощує роботу над проєктом у команді або збереження історії змін.
- Дебагінг: Вбудований відлагоджувач дозволяє швидко знаходити та виправляти помилки.

1.3.2 Мова програмування JavaScript

JavaScript (JS) є основною мовою для створення веб-застосунків, і його вибір обґрунтований такими причинами [22]:

- Універсальність: JS дозволяє створювати як фронтенд (інтерфейс користувача), так і серверну частину.
- Популярність: Мова має багатий екосистему бібліотек і фреймворків для швидкої реалізації складних функцій.

Сумісність: JS підтримується всіма сучасними веб-браузерами, що забезпечує зручність тестування та доступність застосунку.

1.3.3 JavaScript-фреймворк React

React – це популярний JavaScript-фреймворк для створення інтерактивних веб-додатків [23]. Його використання в проєкті забезпечує:

- Компонентна архітектура: Код інтерфейсу розбивається на окремі модулі (компоненти), які можна повторно використовувати.
- Динамічність: React дозволяє оновлювати тільки ті частини інтерфейсу, які змінюються, що підвищує продуктивність.

- Широкий вибір додаткових бібліотек: React легко інтегрується з іншими інструментами, такими як react-chartjs-2.

1.3.4 Бібліотека для візуалізації react-chartjs-2

React-chartjs-2 є обгорткою над бібліотекою Chart.js для роботи з графіками у React-застосунках [24]. У контексті даного проєкту вона дозволяє:

- Відображати PWM-сигнали: Графіки демонструють поведінку сигналів у реальному часі, що полегшує аналіз роботи мікроконтролера.
- Адаптація графіків до потреб користувача: react-chartjs-2 підтримує широкий спектр кастомізації, що дозволяє створити графіки із зрозумілим дизайном.
- Плавна анімація: Анімація переходу між станами допомагає краще зрозуміти зміни в параметрах системи.

3.2 Реалізація WEB-застосунку, що моделює перший таймер-лічильник ATMega2560

Для створення навчальної платформи було реалізовано WEB-застосунок, який моделює роботу таймера-лічильника ATMega2560 та демонструє його взаємодію із сервоприводами. WEB-застосунок включає інтерфейс для налаштування параметрів регістрів таймера та візуалізацію результатів.

3.2.1 Мета реалізації WEB-застосунку

Основною метою реалізації було створення зручного та інтуїтивного інтерфейсу, який:

1. Дозволяє моделювати процес налаштування регістрів мікроконтролера;
2. Візуалізує сигнали PWM та положення маніпулятора у вигляді графіків.

3. Забезпечує навчальну спрямованість шляхом інтерактивної взаємодії з параметрами системи.

3.2.2 Основний інтерфейс програми

WEB-застосунок складається з чотирьох основних областей, див. рис. 3.1:

1. Графік положення маніпулятора – ламана лінія, що в реальному часі показує позицію дволанкового маніпулятора на основі сигналів PWM. Користувач може змінювати параметри регістрів і бачити, як це впливає на положення ланок;
2. Список регістрів – таблиця налаштувань регістрів TIMER1, зокрема TCCR1A, TCCR1B, OCR1A, OCR1B, ICR1. Користувач може змінювати значення бітів (наприклад, вибір режиму PWM, встановлення дільника, поведінку виходів);
3. Графік сигналів – графічне відображення сигналів на двох виходах таймера (OC1A, OC1B). Динамічне оновлення залежно від змін у налаштуваннях;
4. Вивід інформації – текстовий блок із поясненнями для користувача

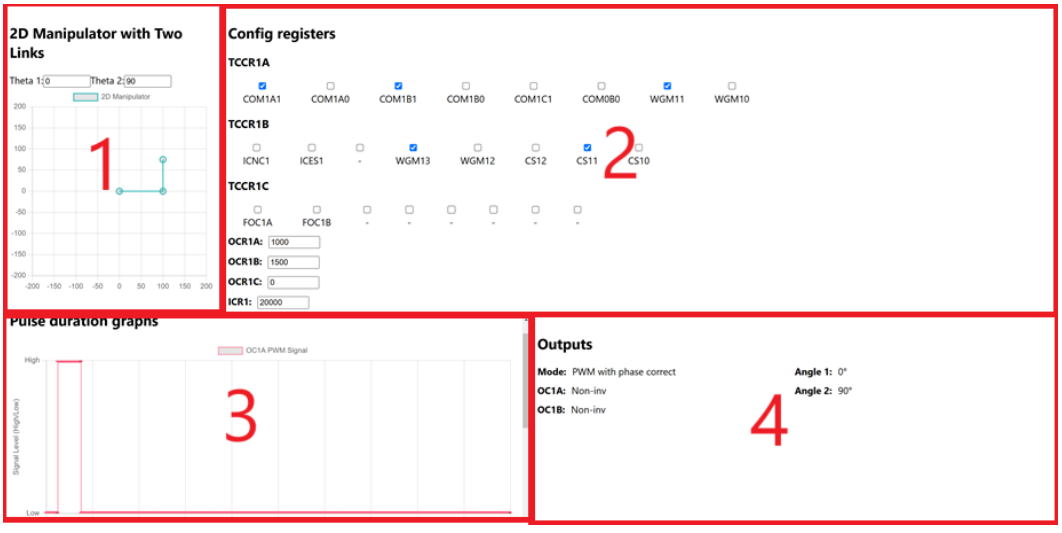

Рисунок 3.1 – Чотири області WEB-застосунку

3.2.3 Функціонал WEB-застосунку

1. Моделювання налаштувань регістрів:
    - Вибір режиму роботи (Fast PWM, Phase Correct PWM, …);
    - Встановлення поведінки виходів (OCR1B, OCR1C);
    - Налаштування верхнього значення для певних режимів (ICR1).
2. Візуалізація PWM-сигналів:
    - Побудова графіків ширини імпульсу для кожного виходу;
    - Відображення частоти та періоду сигналу у зрозумілому вигляді.
3. Інтерактивна взаємодія:
    - Зміна параметрів у режимі реального часу з миттєвим оновленням графіків та положення маніпулятора;
    - Підказки для користувача щодо значень параметрів.
4. Зворотний зв’язок:
    - Виведення помилок у разі некоректного введення даних (наприклад, встановлення ширини імпульсу, що перевищує період сигналу).

3.3 Алгоритми роботи програми

Алгоритм програми, див. рис. 3.2, є ключовим компонентом навчальної платформи, наньому забезпечується взаємодія між введеними користувачем параметрами, їх інтерпретацією та візуалізацією результатів. Реалізація алгоритму враховує:

1. Вибір і налаштування режимів роботи таймера-лічильника;
2. Обробку введених значень регістрів для формування сигналів PWM;
3. Обчислення параметрів положення маніпулятора та вивід результатів у зрозумілому вигляді;.
4. Тестування програми для перевірки її коректності.

Рисунок 3.2 – Алгоритм роботи WEB-застосунку

Алгоритм створений для забезпечення точності, зручності використання та інтерактивності, що дозволяє користувачам отримувати наочні результати та вивчати принципи мікроконтрольного керування. Далі у цьому підрозділі детально описано основні етапи роботи програми.

3.3.1 Опис режимів роботи таймера та поведінки виходів

Мікроконтролер ATmega2560 має кілька режимів роботи таймера-лічильника, див. рис. 3.3, які визначають спосіб формування сигналів на його виходах [25]. Ці режими задаються в регістрах TCCR1A і TCCR1B, див. рис. 3.4, (TCCR1C також існує, але його функціонал виходить за межі лише одноготаймера), де встановлюються біти конфігурації.

№	WGMn3	WGMn2	WGMn1	WGMn0	Режим	TOP
0	0	0	0	0	Звичайний режим	0xFFFF
1	0	0	0	1	Корекція фази ШІМ, 8-біт	0x00FF
2	0	0	1	0	Корекція фази ШІМ, 9-біт	0x01FF
3	0	0	1	1	Корекція фази ШІМ, 10-біт	0x03FF
4	0	1	0	0	Скидання при збігу, CTC	OCR*n*A
5	0	1	0	1	Швидкий ШІМ, 8-біт	0x00FF
6	0	1	1	0	Швидкий ШІМ, 9-біт	0x01FF
7	0	1	1	1	Швидкий ШІМ, 10-біт	0x03FF
8	1	0	0	0	ШІМ, корекція фази та частоти	ICR*n*
9	1	0	0	1	ШІМ, корекція фази та частоти	OCR*n*A
10	1	0	1	0	ШІМ, корекція фази	ICR*n*
11	1	0	1	1	ШІМ, корекція фази	OCR*n*A
12	1	1	0	0	Скидання при збігу, CTC	ICR*n*
13	1	1	0	1	Не використовується	–
14	1	1	1	0	Швидкий ШІМ	ICR*n*
15	1	1	1	1	Швидкий ШІМ	OCR*n*A

Рисунок 3.3 – Усі можливі режими роботи TC1

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
Bit	7	6	5	4	3	2	1	0	
(0x82)	FOC1A	FOC1B	FOC1C	–	–	–	–	–	TCCR1C
Read/Write	W	W	W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

Рисунок 3.4 – Регістри налаштування поведінки таймеру та його виходів (також є біти для роботи із мікроконтролером)

Основні режими роботи таймера TIMER1, див. рис. :

1. Normal Mode (Режим нормального відліку) – таймер просто рахує імпульси до досягнення максимального значення (0xFFFF для 16-бітного таймера). Жодного сигналу на виходах OC1A і OC1B не генерується;

2. CTC Mode (Clear Timer on Compare Match) – таймер скидається при збігу значення лічильника з пороговим значенням у регістрі OCR1A; Використовується для генерування періодичних сигналів на виходах;
3. Fast PWM Mode (Швидка ШІМ) – таймер рахує від 0 до значення в регістрі ICR1 (верхня межа). Виходи OC1A і OC1B генерують широтно-імпульсні сигнали з високою частотою, яка залежить від налаштувань регістрів;
4. Phase Correct PWM Mode (Фазово-коректна ШІМ) – лічильник збільшується від 0 до верхньої межі (ICR1) і потім зменшується до 0. Генерація сигналів відбувається з мінімальним спотворенням завдяки симетричній формі імпульсів.

Дільник частоти. За них відповідає три біти CS, див. рис. 3.5. Він ділить вхідну частоту мікроконтролера, у ATmega2560 це 16 МГц, на певне значення, або підключає зовнішній таймер.

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Рисунок 3.5 – Біти дільника частоти для TC1

Поведінка виходів (OC1A, OC1B, OC1C):

Поведінка виходів таймера визначається встановленням бітів COMnA, COMnB, COMnC, де n – номер виходу, у регістрі TCCR1A. Виходи можуть працювати у таких режимах, див. рис. 3.6:

1. Normal Port Operation – Вихідний сигнал залишається постійним і не змінюється;
2. Toggle on Compare Match – Вихід змінює свій стан при кожному збігу значення лічильника з регістром порівняння (OCR1A або OCR1B);
3. Clear on Compare Match – Вихід встановлюється в "0" при досягненні порогового значення;
4. Set on Compare Match – Вихід встановлюється в "1" при збігу значень.

COM*n*A1 COM*n*B1 COM*n*C1	COM*n*A0 COM*n*B0 COM*n*C0	Опис
0	0	Виводи OC*n*A/OC*n*B/OC*n*C відключені від таймера.
0	1	Стан OC*n*A/OC*n*B/OC*n*C виводів змінюється на протилежний
1	0	Стан OC*n*A/OC*n*B/OC*n*C скидається в «0»
1	1	Стан OC*n*A/OC*n*B/OC*n*C встановлюються в «1»

Рисунок 3.6 – Біти встановлення поведінки виходів для OCR1A(B,C)

Роль регістрів у формуванні сигналів:

1. TCCR1A, TCCR1B: задають режим роботи таймера, поведінку виходів і джерело тактового сигналу;
2. OCR1A, OCR1B, OCR1C: визначають порогові значення для формування ширини імпульсів PWM;
3. ICR1 (OCR1A): використовується як верхня межа лічильника в режимах Fast PWM і Phase Correct PWM для регулювання частоти сигналу.

В цілому ці регістри визначають вигляд сигналу на виході, а саме тривалості високого чи низького рівня сигналу.

3.3.2 Формули для інтерпретації значень регістрів

Регістри мікроконтролера ATmega2560 є ключовим елементом для налаштування таймера-лічильника та формування сигналів PWM. Введені користувачем значення бітів цих регістрів повинні бути перетворені у параметри, які програма використовує для обчислення частоти, ширини імпульсів та інших характеристик. У цьому підрозділі представлено формули, що забезпечують інтерпретацію даних регістрів.

Перш за все має сформуватись значення частоти PWM, оскільки від нього залежить максимальна можлива тривалість імпульсу. Для кожного режиму є своя формула розрахунку частоти.

CTC – Clear to Compare, цей режим можна одразу виключити із проєкту, оскільки на практиці його не використовують для генерування PWM-сигналів. За нього відповідають 4 та 12 режими згідно рис. 3.3. Причина його непопулярності полягає у тому, що він генерує сигнал зі скважністю 50%. Тобто для більшості SD занадто швидкий, при цьому нестабільний і постійно змінюючий свою частоту сигнал.

Fast PWM – стандартний режим для генерації ШІМ. Режими 5, 6, 7, 14, 15. У цьому режимі таймер рахує до верхнього значення, а після скидається. При цьому частота постійно одна і та ж, а тривалість імпульсу на виходах залежить лише від обраної роздільної здатності, користувача та тривалістю імпульсу. Формула 3.1 [25] для розрахунку частоти, у ній видно, що усі значення залежать від користувача, окрім частоти мікроконтролера:

$$f_{PWM} = \frac{f_{clk}}{N(1+TOP)} \tag{3.1}$$

де f_{clk} – частота, на котрій працює мікроконтролер у Гц;

f_{PWM} – частота на виходах;

N – дільник частоти;

TOP – верхнє значення, до котрої потрібно рахувати.

Phase correct PWM – це режими 1, 2, 3, 10, 11. Цей режим, на відміну від швидкого ШІМ, рахує від 0 до верхнього значення, а потім у зворотньому напряму. Формула 3.2 [25] для розрахунку частоти виглядає:

$$f_{PCPWM} = \frac{f_{clk}}{2NTOP} \tag{3.2}$$

де f_{PCPWM} – частота генерації сигналу на виходах.

Phase and Frequency – це режими 8 та 9. Цей режим має усі переваги попереднього, але при цьому він на один такт точніший, оскільки він не витрачає час на встановлення верхнього значення. Для розрахунку частоти використовується формула 3.3 [25]:

$$f_{PFCPWM} = \frac{f_{clk}}{2NTOP} \tag{3.3}$$

Знаючи, що для кожного режиму своя формула, для встановлення частоти виходів буде використано switch-case блок попередньо діставши значення режиму (wave_generation_mode) із обох регістрів контролю за допомогою бітових операцій, див. рис. 3.7.

```
wave_generation_mode = (tccr1b_bitmask<<4)*8 + (tccr1b_bitmask<<3)*4 + (tccr1a_bitmask<<1)*2 + (tccr1a_bitmask<<0);


switch (wave_generation_mode) {
    case 0:
        //top_value = 0xFFFF;
        console.log('Does not support normal mode')
        break;
    case 1:
        top_value = 0x00FF;
        wgm_formula = 2;
        break;
    case 2:
        top_value = 0x01FF;
        wgm_formula = 2;
        break;
    case 3:
        top_value = 0x03FF;
        wgm_formula = 2;
        break;
    case 4:
        //top_value = ocr1a;
        console.log('Does not support CTC mode')
        break;
```

Рисунок 3.7 – Частина коду для вибору режиму

Цей блок коду, після ідентифікації режиму встановлює верхнє значення та формулу для обраного режиму. При цьому режими 0, 4, 12, 13 були виключені за причин перерахованих раніше, а 13 режим зарезервований, тобто пустий.

Також у основній формулі фігурує значення дільника (cs_n), його теж необхідно дістати із регістру керування. Оскільки прямої логіки тут також немає, доведеться використати switch-case, див. рис. 3.8:

```
cs_n = (tccr1b_bitmask<<0) + (tccr1b_bitmask<<1)*2 + (tccr1b_bitmask<<2)*4;
switch (cs_n) {
    case 0:
        console.log('Timer off');
        break;
    case 1:
        n = 1;
        break;
    case 2:
        n = 8;
        break;
    case 3:
        n = 64;
        break;
```

Рисунок 3.8 – Фрагмент коду для визначення дільника

Отже, оскільки формули для частоти дві, для ШІМ із фазовою корекцією та ШІМ із корекцією фази та частоти, створимо код для визначення такого основного параметру як частота, див. рис. 3.9.

```
switch (wave_generation_mode) {
    case 0:
        console.log('Not correct mode');
        break;
    case 1:
        fpwm = fclk / (cs_n * (1 + top_value));
        break;
    case 2:
        fpwm = fclk / (2 * cs_n * top_value);
        break;
    default:
        console.log('Not correct mode');
        break;
}
```

Рисунок 3.9 – Блок коду, що використовує визначає формулу для обчислення частоти

3.3.3 Формули для обробки даних та створення інтерфейсу

Отже дані від користувача були отримані та інтерпретовані. Тепер необхідно обробити отримані дані та представити їх у вигляді графіків та зручних для сприйняття значень.

Для побудови областей, в котрих будуть значення та регістри керування, використано React та chart-js-2. Робочу область було поділено на дві, потім кожну область поділено ще на дві, щоб було отримано 4 необхідні області.

У першій області треба створити графік. Положення маніпулятора розраховується за допомогою тригонометричних функцій, використовуючи кути, визначені сигналами PWM для кожного з двох сервоприводів. Прямі лінії між точками створюються за допомогою chasrt-js.

Формули 3.4, 3.5 для обчислення координат кінцевої точки маніпулятора:

$$x = L1 * \cos(\alpha 1) + L2 * \cos(\alpha 1 + \alpha 2 - 90)\pi/180) \quad (3.4)$$

$$y = L1 * \sin(\alpha 1) + L2 * \sin(\alpha 1 + \alpha 2 - 90)\pi/180) \quad (3.5)$$

де α1 – Кут першої ланки (визначається за сигналом на OC1B);

α2 – Кут другої ланки (визначається за сигналом на OC1C);

L1, L2 – Довжини першої та другої ланок.

В цій формулі є невідомими кути нахилу ланок плаского маніпулятора. Для їх визначення використаємо тривалості імпульсів сигналу та створимо формулу 3.6:

$$\alpha = \frac{T_p - T_{max}}{T_{max} - T_{min}} (\alpha_{max} - \alpha_{min}) + \alpha_{min} \quad (3.6)$$

де T_p – тривалість імпульсу;

T_{max}, T_{min} – максимальна та мінімальна тривалості імпульсу відповідно;

α_{max}, α_{min} – максимальний та мінімальний кут нахилу SD відповідно.

Із цієї отриманої формули видно, що останній необхідний елемент це тривалість імпульсу. Слід зауважити, що тривалість імпульсу (тобто високого рівня сигналу) для різних режимів роботу виходу різний. Для неінвертованої поведінки використаю пропорцію [25] 3.7 для ШІМ із фазовою корекцією:

$$\begin{aligned} TOP &\Leftrightarrow T \\ PWM &\Leftrightarrow t \end{aligned} \quad (3.7)$$

Звідси можна отримати формулу для знаходження тривалості імпульсу 3.8:

$$t = \frac{PWM}{TOP} T \quad (3.8)$$

де t – тривалість імпульсу для неінвертованого ШІМ;

PWM – значення при котрому вихід обнулюється (для неінвертованого) встановлюється користувачем;

TOP – значення до котрого рахує таймер лічильник, встанвлюється користувачем;

T – це період коливань, що обернено пропорційний знайденій частоті виходу f_{PWM}.

Для того, що знайти інвертований, необхідно знати повну тривалість сигналу t_{max} та тривалість імпульсу t, формула 3.9:

$$t_{inv} = t_{max} - t \tag{3.9}$$

Візуалізація першого блоку була розроблена. Але перед тим як виводити результат роботи маніпулятора. Необхідно створити другий блок – блок із регістрами керування:

- TCCR1A (B, C) – регістри, що відповідають за налаштування роботи таймера-лічильника;
- OCR1A (B, C) – регістри порівняння, для визначення поведінки виходів, у даній роботі поведінки саме виходів B, C. У той час як A буде використовуватись для певних режимів як верхнє значення;
- ICR1 – спеціальний окремий регістр для порівняння, що використовується певними режимами.

OCR1 та ICR1 – представлені у вигляді чисел, оскільки це дварегістри, що відображають лише одне 16-бітне число.

Отже, програмна реалізація полягає у створенні чек-боксів, див. рис. 3.10, для встановлення користувачем окремих бітів налаштування, із описом – за що який біт відповідає. Для навчальних цілей було виписано усі можливі значення бітів, але частина з них (FOC1A, FOC1B, ICNC1, ICES1) не працює, оскільки їх функціонал виходить за межі даного проєкту.

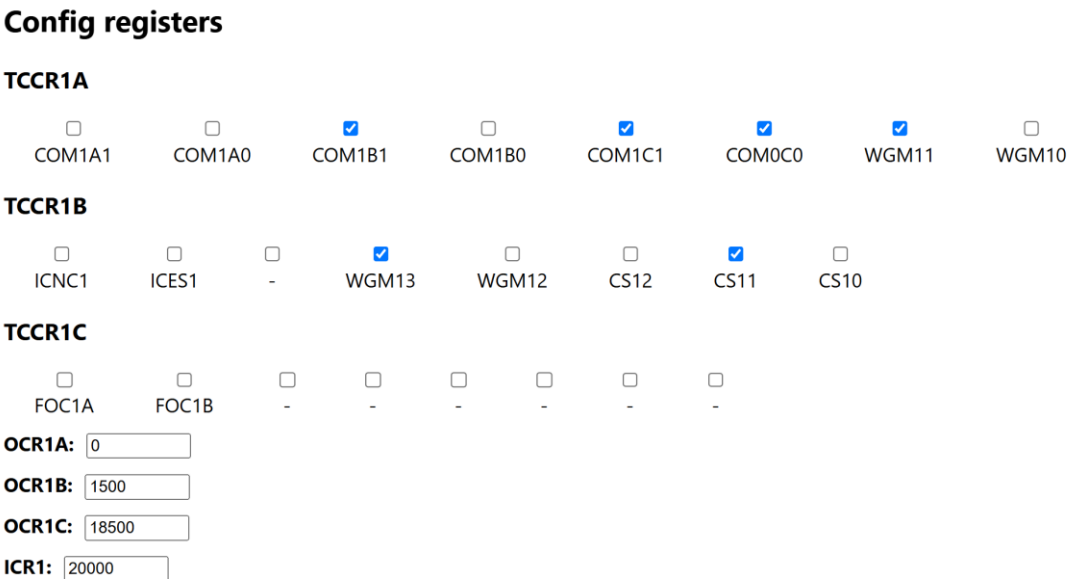

Рисунок 3.10 – Друга область для вводу даних у регістри користувачем

Тепер задавши параметри, можна побачити побудований графік у першій області, див. рис. 3.11. Кут 90 градусів на кожному, оскільки тривалості імпульсу на кожному 1500 мс, при цьому обраний SD SG92R має кутове обмеження у 180 градусів, тобто 1500 мс (нейтральна позиція) дорівнює 90 градусів.

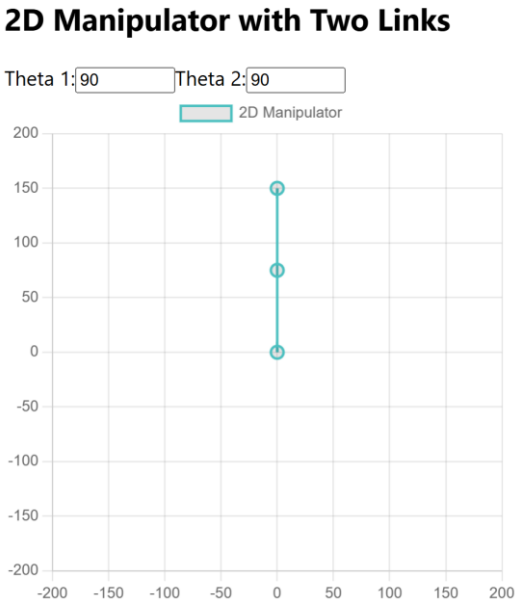

Рисунок 3.11 – Перша область із графічним зображенням плаского дволанкового маніпулятора

Для генерації тривалості імпульсу, використано ті ж бібліотеки, але для побудови сигналів. Оскільки режим, котрий був заданий PWM with phase correct, то сигнал має вигляд для неінвертованого як імпульс на початку та в кінці, для інвертованого навпаки посередині. При цьому, задана частота для ШІМ 50 Гц, тобто тривалість сигналу 20 мс, див. рис. 3.12.

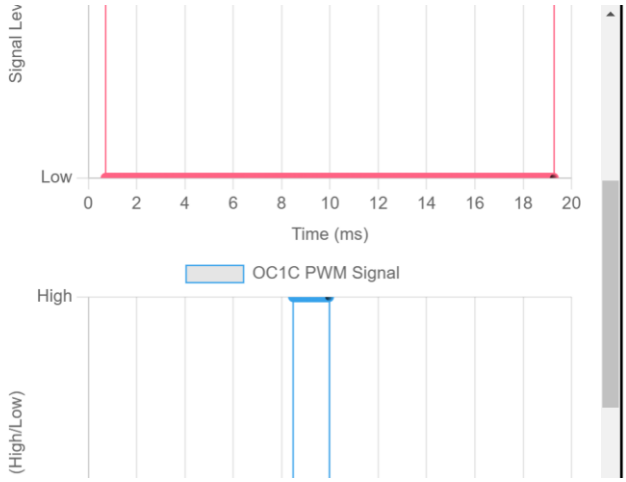

Рисунок 3.12 – Третя область із демонстрацією поведінки виходів

Залишилася остання область, в котрій будуть представлені введені дані у зручному для сприйняття вигляді, див. рис. 3.13.

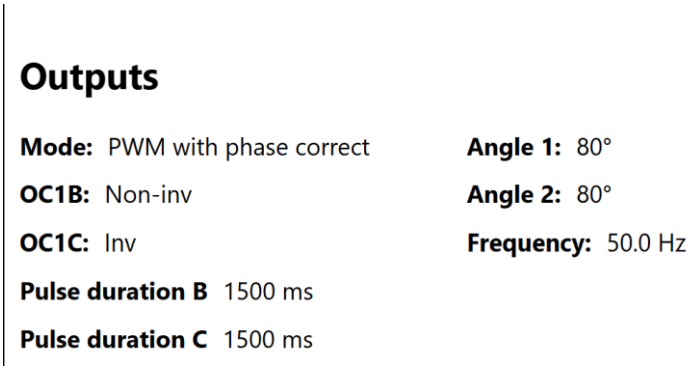

Рисунок 3.13 – Четверта область із зручним представленням даних

Отже, після отримання формул для інтерпретації даних та її подальшої обробки, було створено зручний та інтуїтивно зрозумілий інтерфейс WEB-застосунку, див. рис. 3.14.

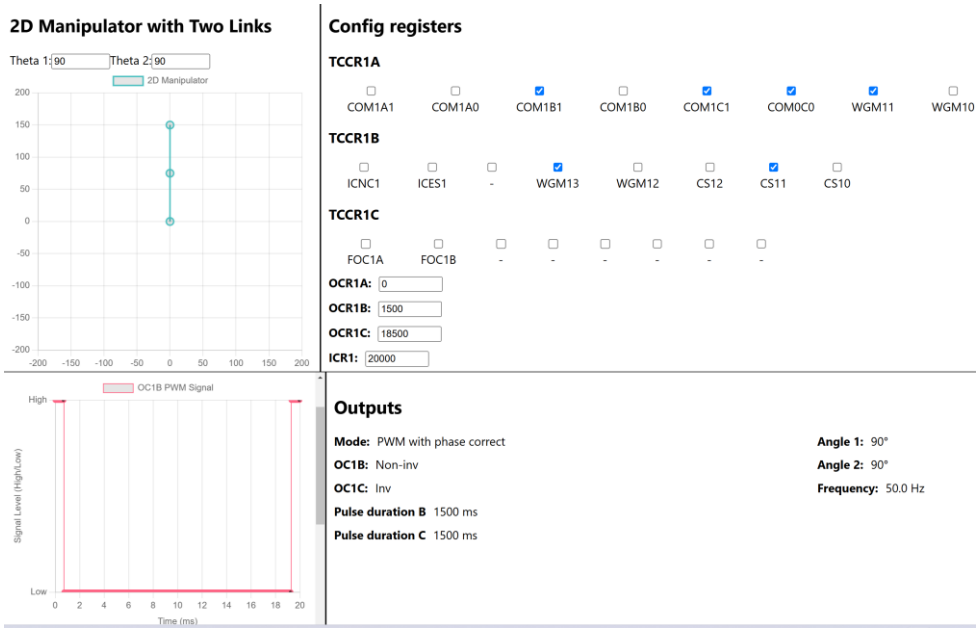

Рисунок 3.14 – Кінцевий інтерфейс застосунку

3.4 Тестування програми

Для перевірки коректності роботи програми проведено тестування, яке включає симуляцію у WEB-застосунку та порівняння результатів із даними реального макета. Тестування виконувалося за такими етапами:

Для тестування було створено два тести. Кожен відповідає за певний режим роботи таймера-лічильника.

3.4.1 Тестування основних функцій програми

Перший тест полягає у налаштуванні таймера в режим 10, PWM with phase correct. Використавши формули отримані у даному проєкті отримаємо:

- Дільник частоти – 1;
- Частота виходів – f_{PWM} =50 Гц, період 20 мс;
- Верхня межа – ICR1 = 20000.

Тест проходитиме у два етапи:

1. Встановлення плаского маніпулятора у положення 1 – з початкового положення повертається на мінімальне значення, а другий на максимальне;
2. Встановлення маніпулятора у положення 2 – другий залишається на місці, а перший встановлюється у середнє положення, після чого повертається назад.

Перший етап. Початкове положення кожної ланки нейтральне, із формул 3.8 та 3.9 отримаємо OC1A = 1500 та OC1B = 18500. Вводимо необхідні дані у регістри контролю, див. рис. 3.15.

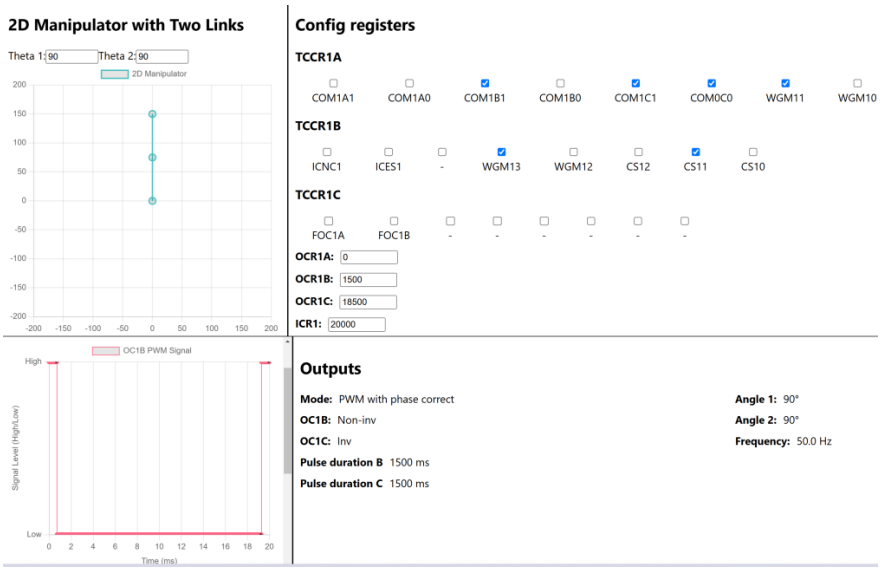

Рисунок 3.15 – Початкове положення Тест 1

Порівняємо графіки отримані застосунком та логічним аналізатором, див. рис. 3.16, а також положення макету плаского маніпулятора та його візуалізації, див. рис. 3.17. З графіків видно, що дані майже співпадають, відхилення спричинене спроможністю логічного аналізатора вчасно піймати сигнал. При цьому положення маніпулятора співпадають.

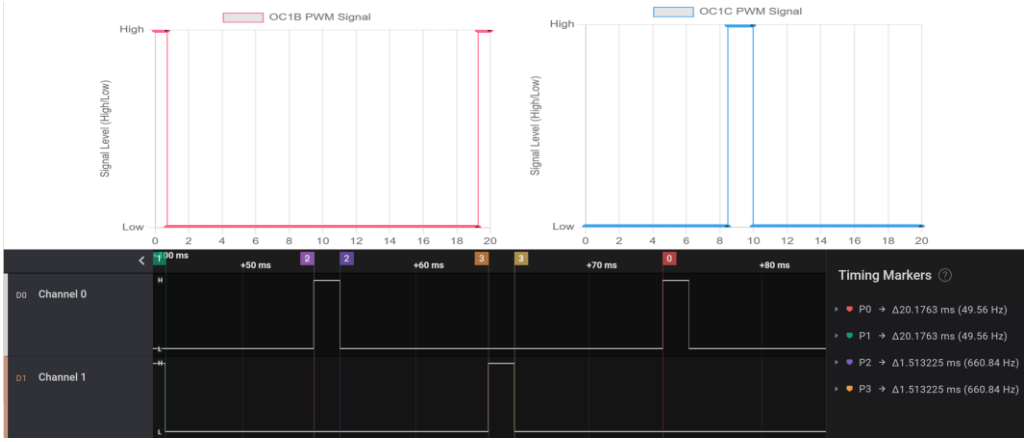

Рисунок 3.16 – Порівняння змодельованого сигналу зі справжнім

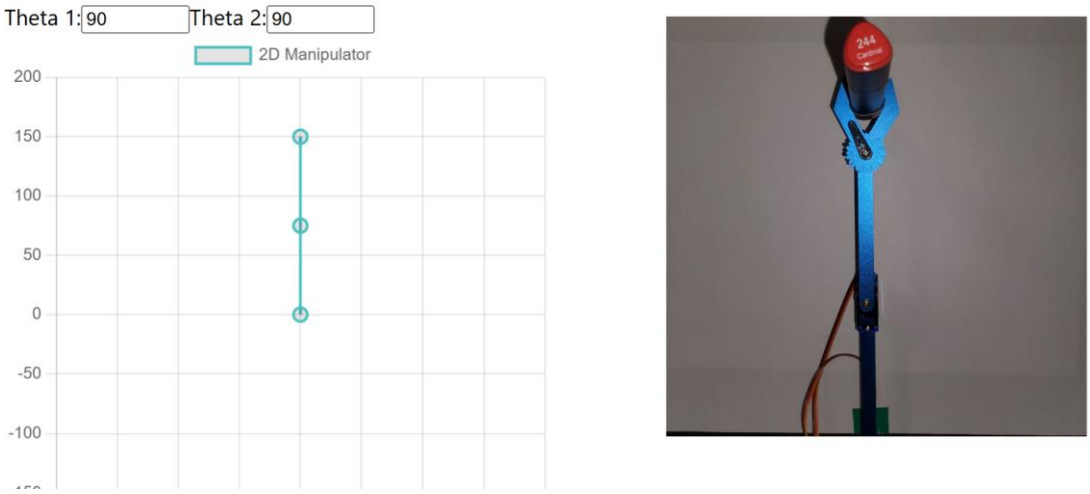

Рисунок 3.17 – Порівняння початкового положення для Тесту 1

Встановимо значення для першого SD у крайнє положення, а другого у протилежне. Для цього треба ввести такі параметри OC1A = 600 та OC1B = 17500, оскільки діапазон SG92R від 600 до 2500 мс. Результат виконання заданих умов видно на рис. та рис. 3.18-3.19. Також перевіримо чи встановилися значення в Outputs, див. рис. 3.20.

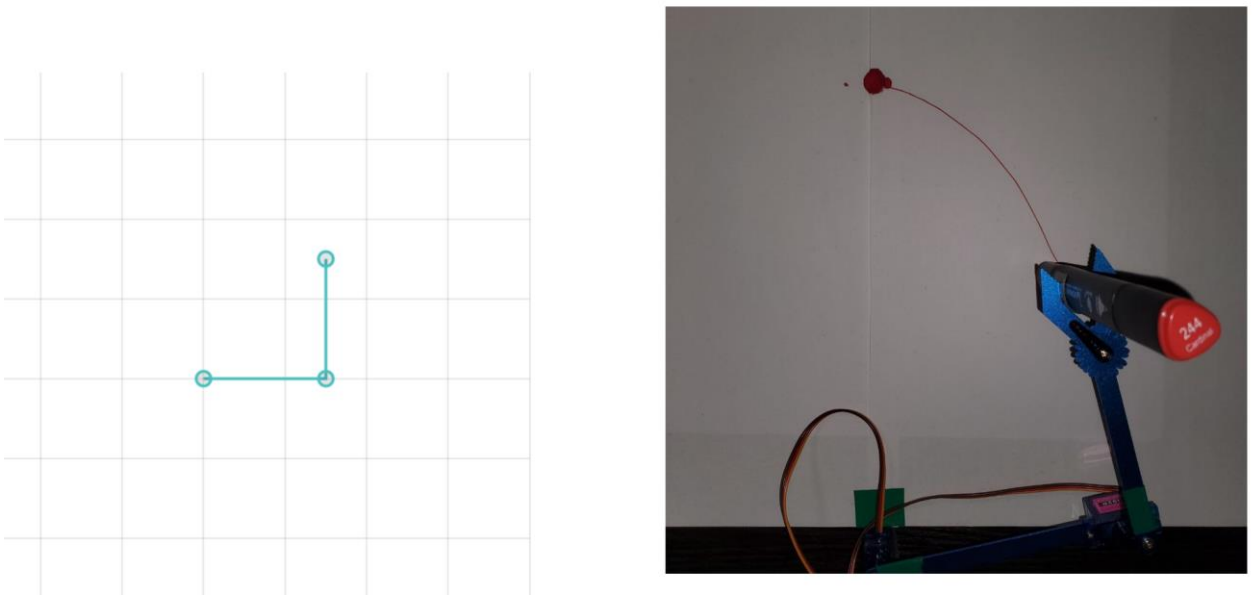

Рисунок 3.18 – Перший етап Тесту 1

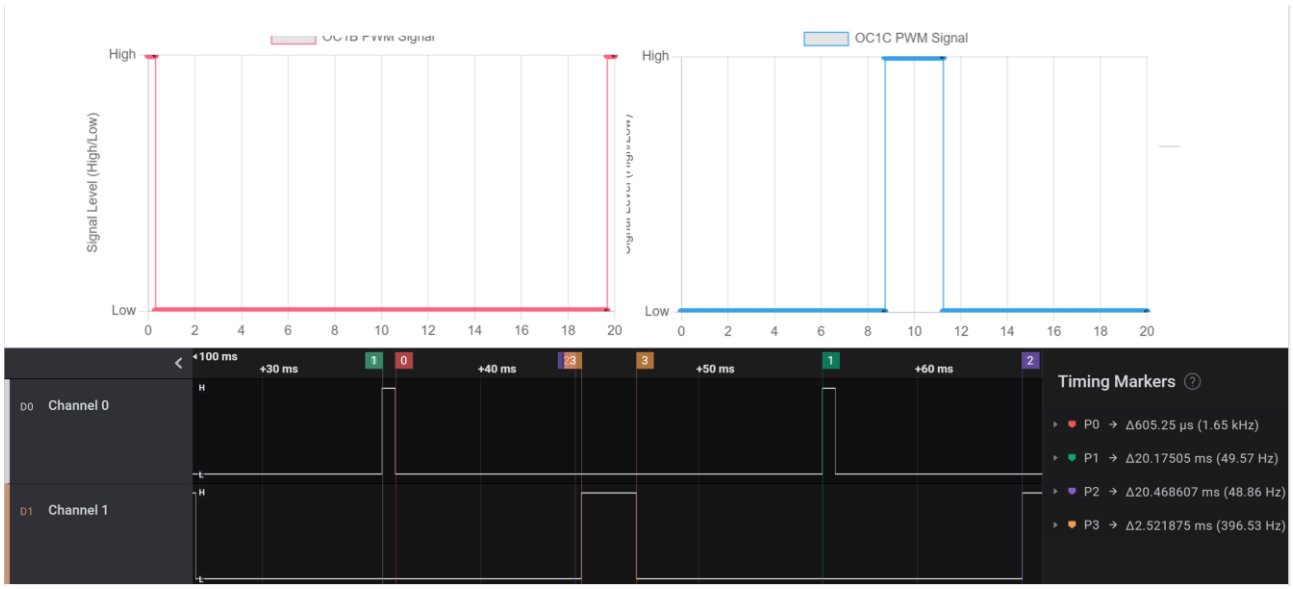

Рисунок 3.19 – Порівняння змодельованих та реальних сигналів

Outputs

Mode: PWM with phase correct
OC1B: Non-inv
OC1C: Inv
Pulse duration B 600 ms
Pulse duration C 2500 ms

Angle 1: 0°
Angle 2: 180°
Frequency: 50.0 Hz

Рисунок 3.20 – Значення Outputs для першого етапу Тесту 1

Другий етап. Треба повернути лише один SD у початкове положення, для цього треба ввести значення OC1A = 1500 та OC1B = 17500. Результат руху видно на рис. 3.21-3.22, декілька ліній утворилося, оскільки цей тест був проведений декілька разів, при цьому відхилення на реальному обладнанні було отвореневипаково при перезапуску експерименту (Зажавши RST випадково утворив сигнал певної довжини, через що SD повів себе інакше, адже від живлення я його не відключив)

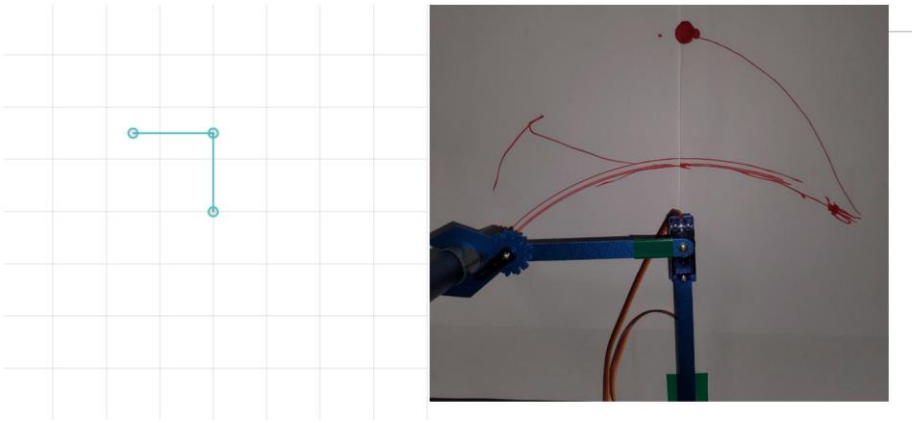

Рисунок 3.21 – Другий етап Тесту 1

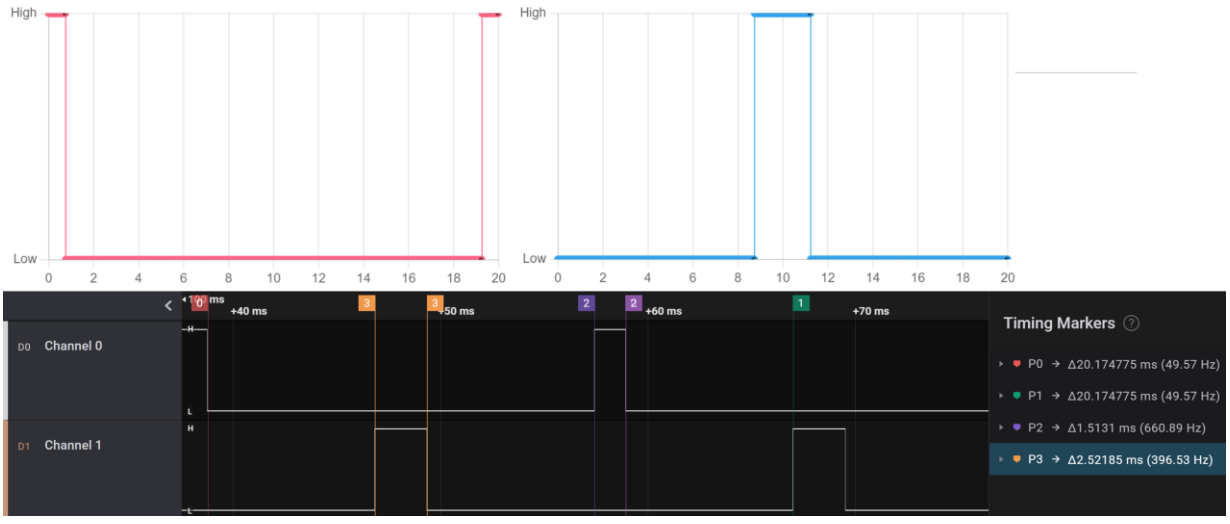

Рисунок 3.22 – Порівняння сигналів для другого етапу Тесту 1

Отже, перший тест продемонстрував роботу WEB-застосунку для візуалізації процесу мікроконтрольного керування пласким маніпулятором у

режимі PWM with phase correct. Та показав, що отримані значення відповідають дійсності.

3.4.2 Тестування альтернативного режиму роботи програми

Тест 2 полягає у перевірці режиму Fast PWM із серією перевірок. Перед початком, необхідно ввести відповідні налаштування, див. рис. 3.23.

1. Дільник – 64
2. Верхнє значення 4999
3. Частота сигналу – 50 Гц

Рисунок 3.23 – Початкові значення для Тесту 2

При заданих

Значення для t 600-2500 лежить для неінвертованого у рамках значення OC1A лежить у 149-599. Для OC1B (інвертованого) лежить у 4549-4999. Серія із зняття покажчиків з графічної візуалізації маніпулятора, реальної моделі маніпулятора, значень візуалізованих сигналів та сигналів з логічного аналізатору представлені на рис. 3.24-3.28, де:

1. Положення першої ланки 45 градусів другої 0;
2. Обидві ланки у 90 градусів;

3. Положення першої 135, другої 180

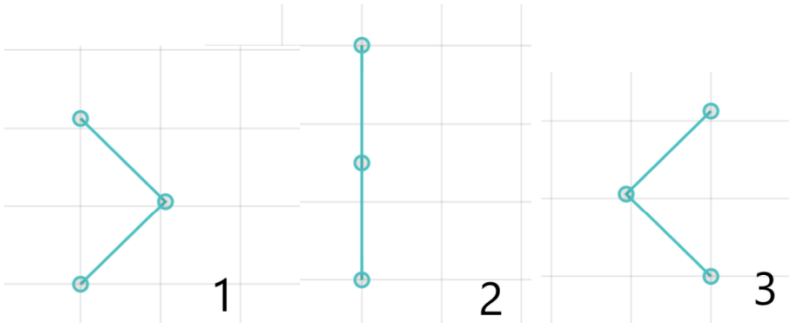

Рисунок 3.24 – Три положення плаского маніпулятора для Тесту 2

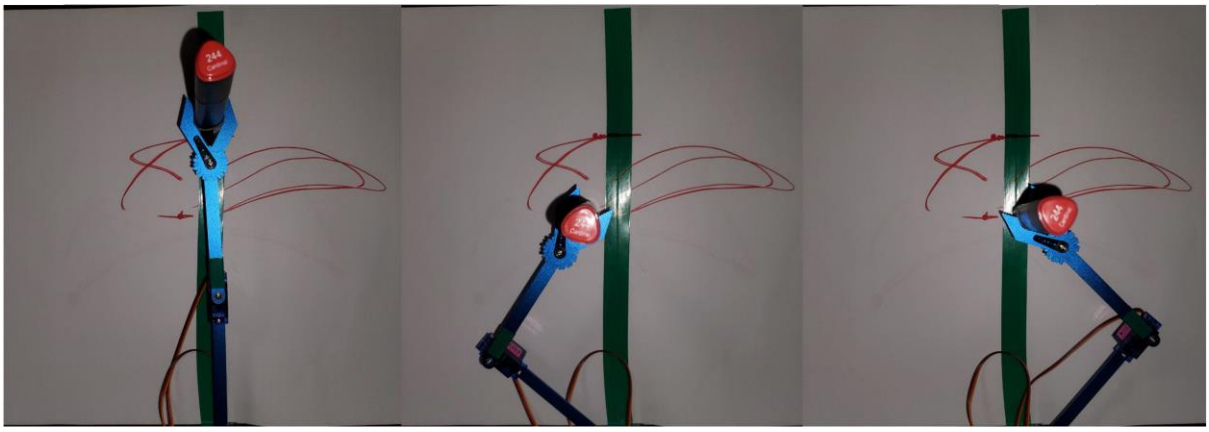

Рисунок 3.25 – Три положення макету плаского маніпулятора для Тесту 2

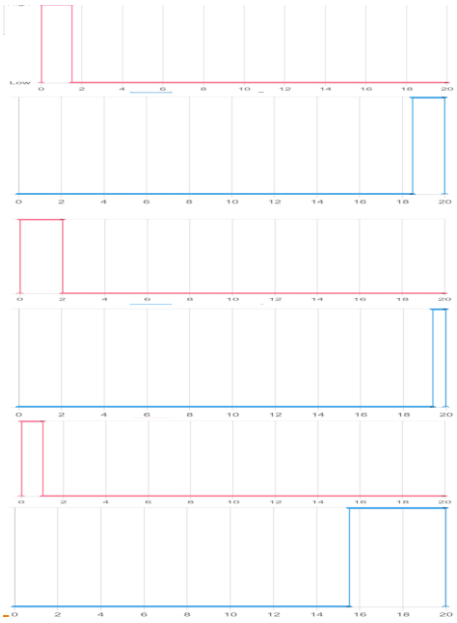

Рисунок 3.26 – Графіки тривалостей імпульсів з візуалізації

Рисунок 3.27– Графіки тривалостей імпульсів з логічного аналізатора

З рисунків видно, що візуалізація процесу мікроконтролерного керування майже відповідає дійсності, не точно оскільки прилади у реальності мають невеликі відхилення. До речі, цікавий вигляд кривої під макетом маніпулятора з’явився, оскільки неінвертований канал спрацьовує близько на 18 мс швидше за інвертований.

Висновки за розділом 3

WEB-застосунок успішно візуалізує роботу першого таймера-лічильника ATMega2560 та його взаємодію із сервоприводами. Він забезпечує наочну візуалізацію процесу налаштування регістрів і демонструє поведінку маніпулятора у залежності від введених параметрів.

ВИСНОВКИ

У межах виконаної роботи було досягнуто поставлену мету – створено навчальну платформу, що візуалізує процес мікроконтролерного керування дволанковим маніпулятором.

Основними результатами є:

Розробка апаратної частини – Реалізовано систему керування сервоприводами на базі мікроконтролера ATmega2560. Використано 16-бітний таймер-лічильник TIMER1 для формування сигналів PWM, які забезпечують точне керування ланками маніпулятора.

Розроблено WEB-застосунок – створено інтуїтивний інтерфейс для налаштування регістрів таймера та візуалізації роботи сервоприводів. Забезпечено інтерактивність та зрозумілість роботи програми завдяки використанню сучасних технологій, таких як React і react-chartjs-2.

Навчальна спрямованість платформи – платформа дозволяє користувачеві вивчити основи мікроконтрольного керування шляхом зміни параметрів регістрів та аналізу результатів у реальному часі.

Завдяки комплексному підходу, платформа не лише демонструє принципи роботи мікроконтролера з сервоприводами, а й виступає інструментом для навчання та дослідження мікроконтрольних систем. Отримані результати можуть бути корисними у навчальних закладах, для самостійного вивчення студентами або як основа для подальшої розробки складніших систем автоматизації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ACTUATORS - SERVOS [Електронний ресурс]. – режим доступу: URL: https://www.societyofrobots.com/actuators_servos.shtml#digitalanalogservos(дата звернення – 20.09.2024).
2. The Purpose of Servo Motors [Елекстронний ресурс] – режим ljступу: URL: https://www.fujielectric.com/about/column/detail/servo_01.html(дата звернення – 20.09.2024).
3. Technical Standards and Key Parameters of Servo Control Systems [Електронний ресурс]. – режим доступу: URL: https://faradyi.com/technical-standards-and-key-parameters-of-servo-control-systems/(дата звернення – 20.09.2024).
4. Servo Motor Micro SG92R [Електронний ресурс] – режим доступу: URL: https://protosupplies.com/product/servo-motor-micro-sg92r/(дата звернення – 10.10.2024).
5. Working with Servo Motors [Електронний ресурс]. – режим доступу: URL: https://xod.io/docs/guide/servo/(дата звернення – 22.09.2024).
6. The Difference between Analog and Digital RC Servos [Електронний ресурс]. – режим доступу: URL: https://www.radiocontrolinfo.com/the-difference-between-analog-and-digital-rc-servos/(дата звернення – 22.09.2024).
7. Servos Explained [Електронний ресурс]. – режим доступу: URL: https://www.sparkfun.com/servos/(дата звернення – 22.09.2024).
8. Robot Manipulator Control: Theory and Practice (Automation and Control Engineering) 2nd Edition

9. PROTEUS DESIGN SUITE Getting Started Guide [Електронний ресурс] – режим доступу: URL: https://www.labcenter.com/downloads/(дата звернення – 11.10.2024).
10. WOKWI [Електронний ресурс] – режим доступу: URL: https://wokwi.com/(дата звернення – 11.10.2024).
11. AUTODESK [Електронний ресурс] – режим доступу: URL: Tinkerad https://www.tinkercad.com/(дата звернення – 11.10.2024).
12. MathWorks Simulink [Електронний ресурс] – режим доступу: URL: https://www.mathworks.com/products/simulink.html(дата звернення – 11.10.2024).
13. Top Microcontrollers for Embedded Systems [Електронний ресурс] – режим доступу: URL: https://octopart.com/pulse/p/top-microcontrollers-embedded-systems(дата звернення – 11.10.2024).
14. ST [Електронний ресурс] – режим доступу: URL: https://www.st.com/content/st_com/en.html(дата звернення – 11.10.2024).
15. ESPRESSIF [Електронний ресурс] – режим доступу: URL: https://www.espressif.com/en/products/socs/esp32(дата звернення – 11.10.2024).
16. MICROCHIP [Електронний ресурс] – режим доступу: URL: https://www.microchip.com/(дата звернення – 11.10.2024).
17. Introduction to Microcontroller Timers: Periodic Timers [Електронний ресурс]. – режим доступу: URL: https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-periodic-timers/(дата звернення – 03.10.2024).
18. ATMEGA2560 Datasheet (PDF) - ATMEL Corporation [Електронний ресурс]. – режим доступу: URL: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-

avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf (дата звернення – 30.09.2024).

19. Programmers and Debuggers AVRISP mkII [Електронний ресурс]. – режим доступу: URL: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42093-AVR-ISP-mkII_UserGuide.pdf(дата звернення – 10.10.2024).
20. Saleae Logic Pro 16 USB Logic Analyzer [Електронний ресурс] – режим доступу: URL: http://downloads.saleae.com/specs/Logic+Pro+16+Data+Sheet.pdf(дата звернення – 10.10.2024).
21. WebStorm [Електронний ресурс] – режим доступу: URL: https://www.jetbrains.com /webstorm/(дата звернення – 10.10.2024).
22. JS Tutorial [Електронний ресурс] – режим доступу: URL: https://www.w3schools.com/js/(дата звернення – 10.10.2024).
23. React [Електронний ресурс] – режим доступу: URL: https://legacy.reactjs.org/(дата звернення – 10.10.2024).
24. ChartJS [Електронний ресурс] – режим доступу: URL: https://www.chartjs.org/docs/latest/(дата звернення – 10.10.2024).
25. INTELLEC*TUAL* CAPITAL IS THE FOUNDATION OF INNOVATIVE DEVELOPMENT: Robotic systems. ChartJS [Електронний ресурс] – режим доступу: URL: https://desymp.promonograph.org/index.php/sge/issue/view/sge28-01(дата звернення – 10.10.2024).

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Навчально-науковий інститут комп'ютерних наук та штучного інтелекту
Кафедра комп'ютерних наук та штучного інтелекту
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність: 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма «Комп'ютеризовані системи управління та автоматика»

ЗАТВЕРДЖУЮ
в.о. завідувача комп'ютерних
систем та робототехніки
к. ф.-м. н., доц. ХРУСЛОВ М. М.
«12» вересня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Соляника Віктора Андрійовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи **Модель мікроконтрольного керування сервоприводами плаского маніпулятора**

керівник роботи Котвицький Альберт Тадеушевич, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету ***№ 4101-5/3657 від 12 листопада 2024 року***

2. Строк подання студентом роботи ***30 листопада 2024 року***

3. Перелік питань, які потрібно розробити)
1) Аналіз технічної літератури, що стосується конструкції та методів керування сервоприводами, пласкими маніпуляторами.
2) Аналіз та вибір мікроконтролерів для керування сервоприводами.
3) Моделювання роботи сервоприводу.
4) Розробка програмної моделі мікроконтролерного керування сервоприводами плаского маніпулятора.
5) Застосування моделі для керування пласким дволанковим маніпулятором із використанням двох сервоприводів.
6) Тестування моделі.
7) Аналіз результатів тестування та ідентифікація виявлених недоліків.
8) Скласти звіт про знайдені в процесі тестування помилки.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Огляд технічної літератури щодо конструкції та керування сервоприводами	05.09.2024 — 15.09.2024
2	Огляд та вибір мікроконтролерів для керування сервоприводами	16.09.2024 — 30.09.2024
3	Вибір засобів розробки для створення програмної моделі керування	01.10.2024 — 10.10.2024
4	Моделювання роботи сервоприводів	11.10.2024 — 31.10.2024
5	Розробка програмної моделі мікроконтрольного керування пласким маніпулятором	01.11.2024 — 10.11.2024
6	Тестування розробленої моделі	11.11.2024 — 18.11.2024
7	Підготовка до попереднього захисту кваліфікаційної роботи	19.09.2024 — 23.10.2024
8	Представлення кваліфікаційної роботи та моделі керівнику дипломного проєкту та рецензенту	24.11.2024 - 30.11.2024

5. Дата видачі завдання ***05 вересня 2024року.***

Студент В. А. Соляник 
ініціали, прізвище підпис

Керівник роботи А. Т. Котвицький 
ініціали, прізвище підпис

Додаток Б

Затверджую

_____

«_____»_____2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

«Модель мікроконтрольного керування сервоприводами плаского маніпулятора»

Назва розділу	Назва та зміст підрозділу
1. Вступ	1.1. Назва програмного виробу: МОДЕЛЬ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ СЕРВОПРИВОДАМИ ПЛАСКОГО МАНІПУЛЯТОРА 1.2. Галузь застосування: робототехніка, промисловість
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 151 – Автоматизація та комп’ютерно-інтегровані технології 2.2. Завдання на кваліфікаційну роботу магістра № **4101-5/3657 від 12 листопада 2024 року** (представити як Додаток А до пояснювальної записки до кваліфікаційної роботи).
3. Призначення розробки	3.1. Мета розробки програмного виробу: візуалізувати процес мікроконтрольного керування дволанковим маніпулятором та зробити апаратну реалізації для перевірки теоретично отриманих значень 3.2. Призначення програмного виробу: Програмний виріб призначений для навчання основам мікроконтролерного керування. Він дозволяє моделювати

	роботу таймера-лічильника ATmega2560, формувати PWM-сигнали для сервоприводів і візуалізувати положення дволанкового маніпулятора. 3.3. Вихідні дані для розробки: значення регістрів керування таймером-лічильником мікроконтролера
4. Технічні вимоги до програмного виробу	4.1. Вимоги до функціональних характеристик: немає 4.2. Вимоги до надійності: забезпечити якісну візуалізацію процесу керування. 4.3. Вимоги до умов експлуатації :немає. 4.4. Вимоги до складу і параметрів технічних засобів: звичайне обчислювальне обладнання; ПК; встановлена програма. 4.5. Вимоги до інформаційної та програмної сумісності : немає 4.6. Вимоги до маркування та упаковки: немає 4.7. Вимоги до транспортування та зберігання: немає 4.8. Спеціальні вимоги : немає
5. Вимоги до програмної документації.	Програмною документацією до виробу «Модель мікроконтрольного керування сервоприводами плаского маніпулятора» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Методику розрахунку інформативності змінних стану (у вигляді глав 3.3 та 3.4 пояснювальної записки до кваліфікаційної роботи).

<table>
<tr><td></td><td colspan="2">3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)</td></tr>
<tr><td>6. Вимоги до техніко-економічних показників</td><td colspan="2">Програмною документацією до виробу «Метод аналізу інформативності змінних стану при діагностиці систем з використанням інформаційних критеріїв» вважати:
1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Методику розрахунку інформативності змінних стану (у вигляді глав 3.3 та 3.4 пояснювальної записки до кваліфікаційної роботи).
3) Опис виробу (представити в розділі 3 пояснювальної записки до кваліфікаційної роботи)</td></tr>
<tr><td rowspan="8">7. Стадії та етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>Вересень 2024</td><td>Огляд технічної літератури щодо конструкції та керування сервоприводами</td></tr>
<tr><td>Вересень 2024</td><td>Огляд та вибір мікроконтролерів для керування сервоприводами</td></tr>
<tr><td>Вересень 2024</td><td>Вибір засобів розробки для створення програмної моделі керування</td></tr>
<tr><td>Жовтень 2024</td><td>Моделювання роботи сервоприводів</td></tr>
<tr><td>Жовтень 2024 – Листопад 2024</td><td>Розробка програмної моделі мікроконтрольного керування пласким маніпулятором</td></tr>
<tr><td>Листопад 2024</td><td>Тестування розробленої моделі</td></tr>
</table>

	Листопад 2024	Підготовка до попереднього захисту дипломної роботи
	Листопад 2024	Представлення дипломного проєкту та моделі керівнику дипломного проєкту та рецензенту
8. Порядок контролю та приймання	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати на паперових носіях в 1 примірнику і в електронному вигляді в 1 примірнику.	

Виконавець
студент групи КУ- 61
Соляник В. А. _____

Замовник
к.ф.-м.н., доцент кафедри КСР
Котвицький А. Т. _____

Додаток В

ПРОГРАМНА РЕАЛІЗАЦІЯ

Програма і методика випробувань

програмного виробу

«МОДЕЛЬ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ СЕРВОПРИВОДАМИ ПЛАСКОГО МАНІПУЛЯТОРА»

1 Об’єкт випробувань

1. Назва програмного виробу : «МОДЕЛЬ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ СЕРВОПРИВОДАМИ ПЛАСКОГО МАНІПУЛЯТОРА»

2. Галузь застосування : Автоматизація та приладобудування

3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації системи заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

1) Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

2) Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі Комп’ютерного класу кафедри в період роботи атестаційної комісії.

3) Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

4) Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередоднізасідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

1) працювати на основних операційних системах: Windows, Linux, MacOS;

2) вимоги до надійності;

3) передбачити захист від некоректних дій користувача;

4) сумісність з іншими програмними продуктами;

5) зменшити об’єм програмного коду необхідного для створення веб-додатків;

6) бути легко розширюваною;

7) елементи програми повинні бути ізольовані одне від одного длязменшення їх впливу на роботи програми під час редагування програмного коду;

8) вимоги до складу і параметрів технічних засобів;

9) вимоги до маркування та упаковки (не висуваються);

10) вимоги до транспортування і зберігання (не висуваються).

Спеціальні вимоги (не пред’являються).

5. Вимоги до програмної документації

Програмною документацією до виробу «Модель мікроконтрольного керування сервоприводами плаского маніпулятора е» вважати:

1. Справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);

2. Програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);

3. Рекомендацій щодо застосування створеної програмної стандартизації у проектах (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи).

4. Текст програми(представити як Додаток Г до пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Для проведення випробувань необхідний проєкт для розробки веб-додатку на мові програмування Java Script з використання бібліотек React та Chart.js, а також для верифікації пропонується узяти апаратну реалізацію з Розділу 2.

6.2 Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

-ознайомчий (1-й етап);

-випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку комплектності програмної документації.

2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.

3. Перевірку комплектності складу технічних і програмних засобів.

4. Методику проведення перевірок на 1 етапі випробувань.

5. Якість програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. перевірку відповідності технічних характеристик програми вимогам технічного завдання;

2. перевірку ступеня виконання функціональних вимог до програми;

3. методику проведення перевірок, що входять до переліку по 2 етапу випробувань.

1. Програма працює відповідно до умов експлуатації операційних систем MS Windows, Linux та MacOS.

2. Для роботи необхідний Node.js, WebStorm.

3. Порядок проведення випробувань:

3.1. Запуск програми здійснюється за допомогою запуску веб-серверу на мові програмування JS. Для цього необхідно в директорії із файлом “manipulator2D.js” викликати консольну команду “npm start ”;

3.2. Після запуску програми необхідно у браузері комп’ютеру перейти за посиланням: http://localhost:3000/;

3.3.У вікні з’явиться чотири області. У другій області необхідно внести початнові дані;

3.4. Після чого програма обробить значення та виведе їх у три інші області у різних форматах.

Для проведення випробувань пропонується тест 1 та тест 2.

Тест 1

1. Перевірка виконання програми;

2. Введення даних для режиму PWM with phase correct у програму та макет.

3. Отримання результату обробки даних у вигляді руху плаского маніпулятора, виведення графіків сигналів та зручну для сприйняття інтерпретацію.

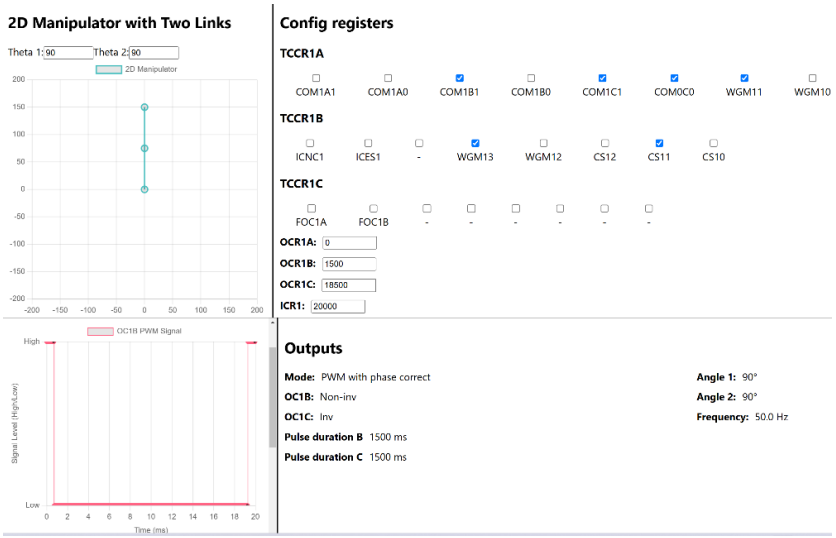

Рис. В.1 Тест 1

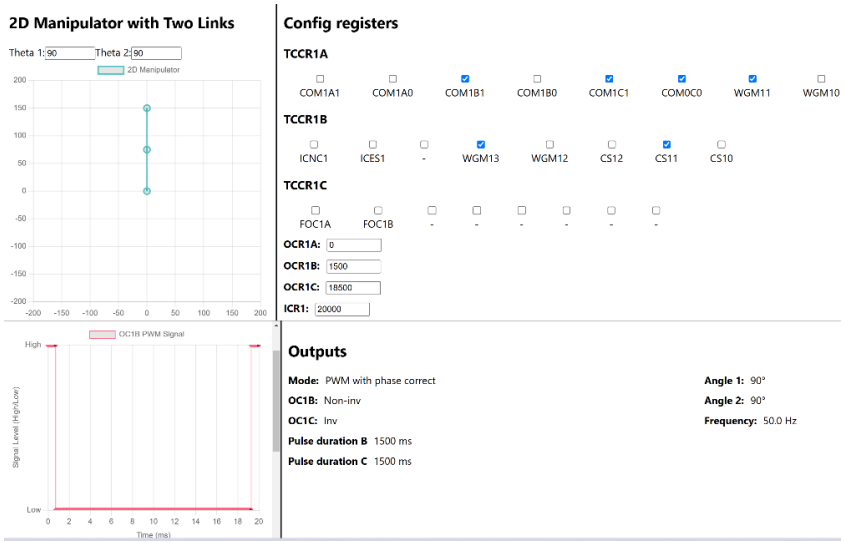

Рис. В.2 Тест 1

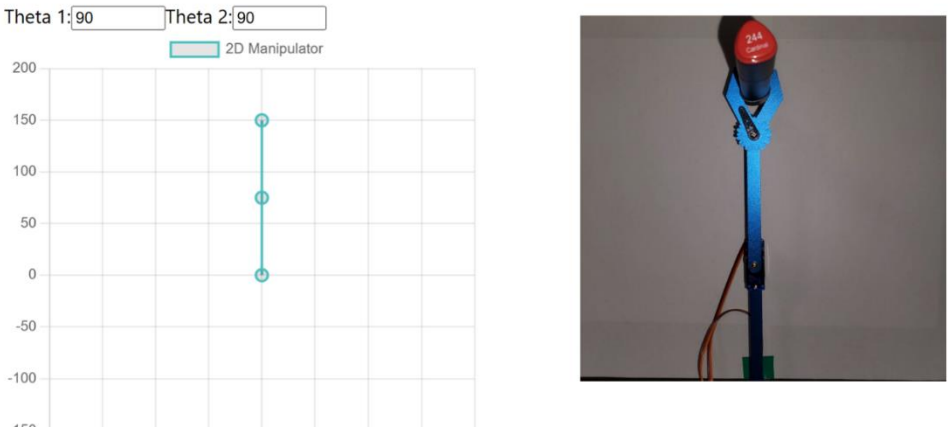

Рис. В.3 Тест 1

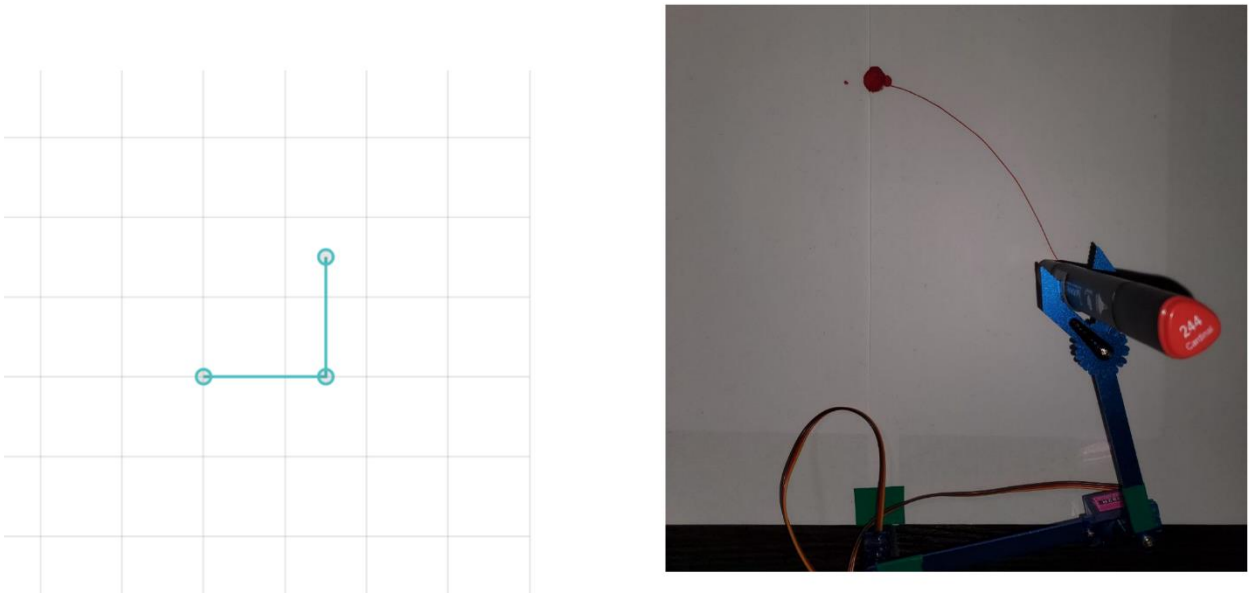

Рис. В.4 Тест 1

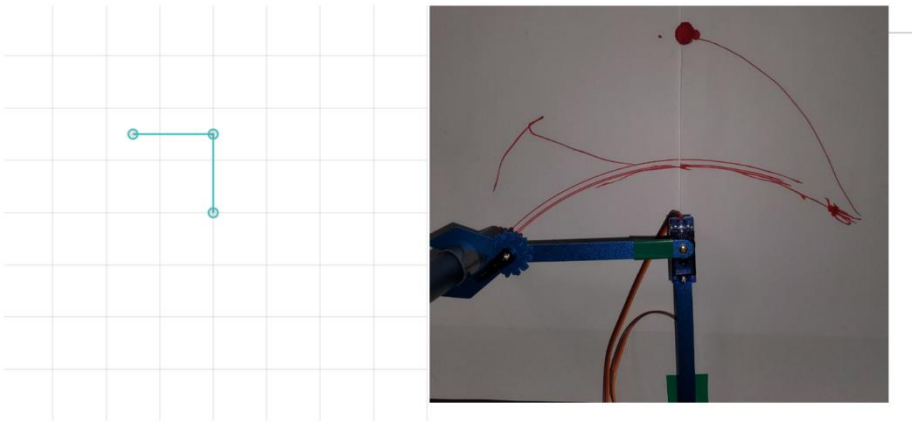

Рис. В.5 Тест 1

Тест 2

1. Перевірка виконання програми;

2. Введення серії нових даних;

3. Отримання результату для інших режимів роботи таймера-лічильника та перевірку сигналів.

Рис. В.6 Тест 2

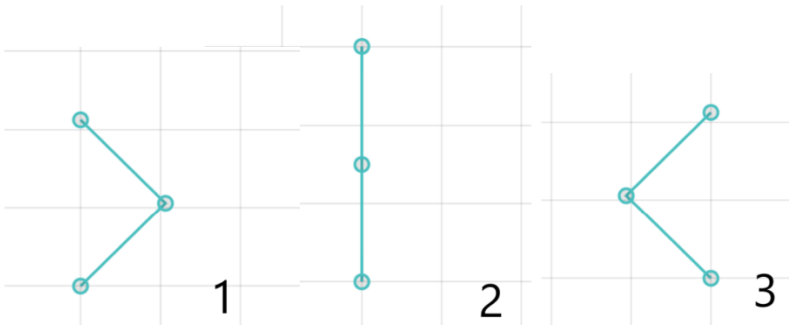

Рис. В.7 Тест 2

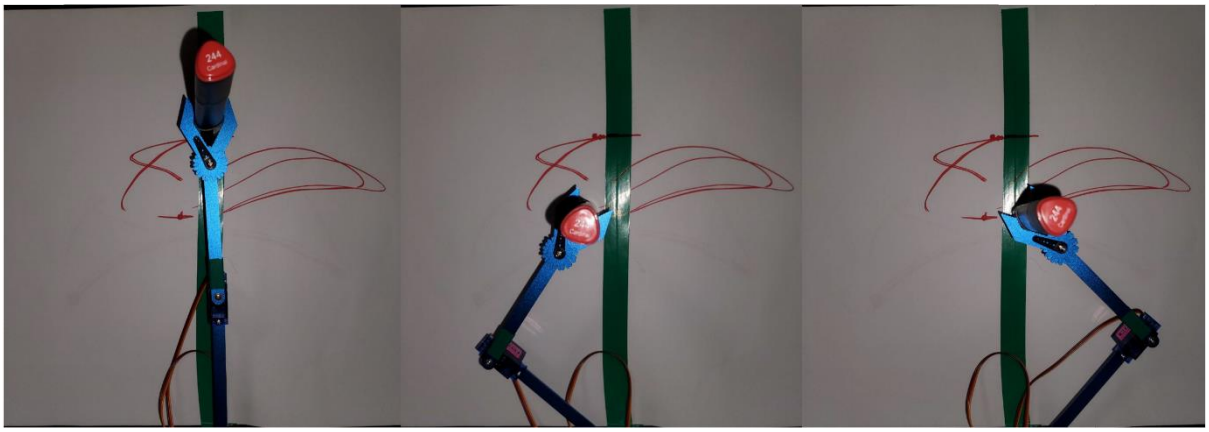

Рис. В.8 Тест 2

Рис. В.9 Тест 2

Висновки: тест 1 успішно пройшов випробування і тест 2 успішно пройшов випробування. Випробування пройшло успішно.

Виконавець: студент групи КУ-61, Соляник В. А. _____

Додаток Г

Лістинг частини виконавчого коду для візуалізації

```
const Manipulator2D = () => {
    const [angles, setAngles] = useState({ theta1: 45, theta2: 90 });
    const [data, setData] = useState(null);

    useEffect(() => {
        const linkLength1 = 75;
        const linkLength2 = 75;

        const x1 = linkLength1 * Math.cos((angles.theta1 * Math.PI) / 180);
        const y1 = linkLength1 * Math.sin((angles.theta1 * Math.PI) / 180);

        const x2 = x1 + linkLength2 * Math.cos(((angles.theta1 + angles.theta2 - 90) * Math.PI) /
180);
        const y2 = y1 + linkLength2 * Math.sin(((angles.theta1 + angles.theta2 - 90) * Math.PI) /
180);

        setData({
            labels: ['Base', 'Joint 1', 'End Effector'],
            datasets: [
                {
                    label: '2D Manipulator',
                    data: [
                        { x: 0, y: 0 },
                        { x: x1, y: y1 },
                        { x: x2, y: y2 },
                    ],
                    borderColor: 'rgba(75,192,192,1)',
                    borderWidth: 2,
                    showLine: true,
                    fill: false,
                    pointRadius: 5,
                    parsing: false,
                },
            ],
        });
    }, [angles]);

    const handleChange = (e) => {
        const { name, value } = e.target;
        const newValue = Math.max(-180, Math.min(180, Number(value)));
        setAngles((prevAngles) => ({ ...prevAngles, [name]: newValue }));
    };


    return (
        <PanelGroup direction="vertical" style={{ height: '100vh' }}>
            <Panel defaultSize={50} minSize={10}>
                <PanelGroup direction="horizontal">
                    <Panel defaultSize={50} minSize={10}>
                        <div style={{ height: '100%', border: '1px solid black', padding: '10px',
overflow: 'hidden' }}>
                            <h2>2D Manipulator with Two Links</h2>
                            <div>
                                <label>
                                    Theta 1:
                                    <input
                                        type="number"
                                        name="theta1"
                                        value={angles.theta1}
                                        onChange={handleChange}
                                        min={-180}
                                        max={180}
                                    />
                                </label>
                                <label>
                                    Theta 2:
                                    <input
                                        type="number"
                                        name="theta2"
                                        value={angles.theta2}
                                        onChange={handleChange}
                                        min={-180}
```

```
                                    max={180}
                                />
                            </label>
                        </div>
                        {data && (
                            <div style={{ position: 'relative', width: '100%', height:
'100%', overflow: 'auto' }}>
                                <Line
                                    data={data}
                                    options={{
                                        responsive: true,
                                        maintainAspectRatio: true,
                                        aspectRatio: 1,
                                        scales: {
                                            x: {
                                                type: 'linear',
                                                position: 'bottom',
                                                min: -200,
                                                max: 200,
                                                ticks: {
                                                    stepSize: 50,
                                                },
                                            },
                                            y: {
                                                min: -200,
                                                max: 200,
                                                ticks: {
                                                    stepSize: 50,
                                                },
                                            },
                                        },
                                        animation: {
                                            duration: 0,
                                        },
                                    }}
                                />
                            </div>
                        )}
                    </div>
                </Panel>
                <PanelResizeHandle />
                <Panel defaultSize={50} minSize={10}>
                    <div style={{ height: '100%', border: '1px solid black', padding: '10px',
overflow: 'auto' }}>
                        <h2>Config registers</h2>
                        <div>
                            <div>
                                <h3>TCCR1A</h3>
                                <div style={{ display: 'flex' }}>
                                    {[...Array(8)].map((_, index) => (
                                        <div key={`tccr1a-${index}`} style={{ display:
'inline-flex', flexDirection: 'column', alignItems: 'center', margin: '0 25px' }}>
                                            <input type="checkbox" onChange={(e) =>
console.log(`TCCR1A Bit ${index + 1}:`, e.target.checked)} />
                                            <span>{['COM1A1', 'COM1A0', 'COM1B1', 'COM1B0',
'COM1C1', 'COM0C0', 'WGM11', 'WGM10'][index]}</span>
                                        </div>
                                    ))}
                                </div>
                            </div>
                            <div>
                                <h3>TCCR1B</h3>
                                <div style={{ display: 'flex' }}>
                                    {[...Array(8)].map((_, index) => (
                                        <div key={`tccr1b-${index}`} style={{ display:
'inline-flex', flexDirection: 'column', alignItems: 'center', margin: '0 25px' }}>
                                            <input type="checkbox" onChange={(e) =>
console.log(`TCCR1B Bit ${index + 1}:`, e.target.checked)} />
                                            <span>{['ICNC1','ICES1','-
','WGM13','WGM12','CS12','CS11','CS10'][index]}</span>
                                        </div>
                                    ))}
                                </div>
                            </div>
                            <div>
                                <h3>TCCR1C</h3>
                                <div style={{ display: 'flex' }}>
                                    {[...Array(8)].map((_, index) => (
```

```
                                        <div key={`tccr1c-${index}`} style={{ display:
'inline-flex', flexDirection: 'column', alignItems: 'center', margin: '0 25px' }}>
                                            <input type="checkbox" onChange={(e) =>
console.log(`TCCR1C Bit ${index + 1}:`, e.target.checked)} />
                                            <span>{['FOC1A','FOC1B','-','-','-','-','-','-
'][index]}</span>
                                        </div>
                                    ))}
                                </div>
                            </div>
                            <div>
                                <label style={{ display: 'block', margin: '10px 0',
fontWeight: 'bold' }}>
                                    OCR1A:
                                    <input type="number" max={65535} min={0} step={1}
style={{ marginLeft: '10px' }} onChange={(e) => { const value = Math.max(0, Math.min(65535,
Number(e.target.value))); e.target.value = value; console.log(`OCR1A:`, value); }} />
                                </label>
                            </div>
                            <div>
                                <label style={{ display: 'block', margin: '10px 0',
fontWeight: 'bold' }}>
                                    OCR1B:
                                    <input type="number" max={65535} min={0} step={1}
style={{ marginLeft: '10px' }} onChange={(e) => { const value = Math.max(0, Math.min(65535,
Number(e.target.value))); e.target.value = value; console.log(`OCR1B:`, value); }} />
                                </label>
                            </div>
                            <div>
                                <label style={{ display: 'block', margin: '10px 0',
fontWeight: 'bold' }}>
                                    OCR1C:
                                    <input type="number" max={65535} min={0} step={1}
style={{ marginLeft: '10px' }} onChange={(e) => { const value = Math.max(0, Math.min(65535,
Number(e.target.value))); e.target.value = value; console.log(`OCR1C:`, value); }} />
                                </label>
                            </div>
                            <div>
                                <label style={{ display: 'block', margin: '10px 0',
fontWeight: 'bold' }}>
                                    ICR1:
                                    <input type="number" max={65535} min={0} step={1}
style={{ marginLeft: '10px' }} onChange={(e) => { const value = Math.max(0, Math.min(65535,
Number(e.target.value))); e.target.value = value; console.log(`ICR1:`, value); }} />
                                </label>
                            </div>
                        </div>
                    </div>
                </Panel>
            </PanelGroup>
        </Panel>
        <PanelResizeHandle />
        <Panel defaultSize={50} minSize={10}>
            <PanelGroup direction="horizontal">
                <Panel defaultSize={50} minSize={10}>
                    <div style={{ height: '100%', border: '1px solid black', padding: '10px',
overflow: 'auto' }}>
                        <h2>Pulse duration graphs</h2>
                        <div style={{ height: '100%' }}>
                            <Line
                                data={{
                                    labels: Array.from({ length: 1000 }, (_, i) => i * 0.02),
                                    datasets: [
                                        {
                                            label: 'OC1B PWM Signal',
                                            data: Array.from({ length: 1000 }, (_, i) => (i
>= 1 && i < 51  ? 1 : 0) ),
                                            borderColor: 'rgba(255, 99, 132, 1)',
                                            borderWidth: 1,
                                            stepped: true,
                                            fill: false,
                                        }
                                    ],
                                }}
                                options={{
                                    responsive: true,
                                    maintainAspectRatio: false,
                                    scales: {
```

```
                                        x: {
                                            type: 'linear',
                                            title: {
                                                display: true,
                                                text: 'Time (ms)'
                                            },
                                            ticks: {
                                                stepSize: 2,
                                            },
                                        },
                                        y: {
                                            min: 0,
                                            max: 1,
                                            title: {
                                                display: true,
                                                text: 'Signal Level (High/Low)'
                                            },
                                            ticks: {
                                                stepSize: 1,
                                                callback: (value) => (value === 1 ? 'High' :
'Low'),
                                            },

                                        },
                                    },
                                    animation: {
                                        duration: 0,
                                    },
                                }}
                            />
                            <Line
                                data={{
                                    labels: Array.from({ length: 1000 }, (_, i) => i * 0.02),
                                    datasets: [
                                        {
                                            label: 'OC1C PWM Signal',
                                            data: Array.from({ length: 1000 }, (_, i) => (i
>= 776 && i < 999 ? 1 : 0)),
                                            borderColor: 'rgba(54, 162, 235, 1)',
                                            borderWidth: 1,
                                            stepped: true,
                                            fill: false,
                                        }
                                    ],
                                }}
                                options={{
                                    responsive: true,
                                    maintainAspectRatio: false,
                                    scales: {
                                        x: {
                                            type: 'linear',
                                            title: {
                                                display: true,
                                                text: 'Time (ms)'
                                            },
                                            ticks: {
                                                stepSize: 2,
                                            },
                                        },
                                        y: {
                                            min: 0,
                                            max: 1,
                                            title: {
                                                display: true,
                                                text: 'Signal Level (High/Low)'
                                            },
                                            ticks: {
                                                stepSize: 1,
                                                callback: (value) => (value === 1 ? 'High' :
'Low'),
                                            },
                                        },
                                    },
                                    animation: {
                                        duration: 0,
                                    },
                                }}
                            />
```

```
                            </div>
                        </div>
                    </Panel>
                    <PanelResizeHandle />
                    <Panel defaultSize={50} minSize={10}>
                        <div style={{ height: '100%', border: '1px solid black', padding: '10px',
overflow: 'auto' }}>
                            <h2>Outputs</h2>
                            <div style={{display: 'flex', flexDirection: 'row', justifyContent:
'space-between'}}>
                                <div style={{flex: 1}}>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>Mode:</label>
                                        <span style={{marginLeft: '10px'}}>Fast PWM</span>
                                    </div>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>OC1B:</label>
                                        <span style={{marginLeft: '10px'}}>Non-inv</span>
                                    </div>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>OC1C:</label>
                                        <span style={{marginLeft: '10px'}}>Inv</span>
                                    </div>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>Pulse duration
B</label>
                                        <span style={{marginLeft: '10px'}}>1500 ms</span>
                                    </div>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>Pulse duration
C</label>
                                        <span style={{marginLeft: '10px'}}>1500 ms</span>
                                    </div>
                                </div>
                                <div style={{flex: 1}}>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>Angle 1:</label>
                                        <span style={{marginLeft: '10px'}}>90°</span>
                                    </div>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>Angle 2:</label>
                                        <span style={{marginLeft: '10px'}}>90°</span>
                                    </div>
                                    <div style={{marginBottom: '10px'}}>
                                        <label style={{fontWeight: 'bold'}}>Frequency:</label>
                                        <span style={{marginLeft: '10px'}}>50.0 Hz</span>
                                    </div>

                                </div>
                            </div>
                        </div>
                    </Panel>
                </PanelGroup>
            </Panel>
        </PanelGroup>
    );
};
```