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 КРОКОВИМИ ДВИГУНАМИ ДЛЯ ПОВОРОТУ КОЛІС РУХОМОЇ ПЛАТФОРМИ»

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
КИРИЧЕНКО Олександр Сергійович
_____
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 76 сторінок, із яких 50 сторінок основної частини з 16 рисунками, списку використаних джерел із 20 найменувань та двома додатками.

Метою кваліфікаційної роботи є розробка навчальної платформи для візуалізації процесу формування послідовностей імпульсів з використанням 8-бітного таймера-лічильника мікроконтролера, призначених для керування кроковими двигунами.

Об'єкт дослідження – процес керування кроковими двигунами для повороту коліс рухомої платформи за допомогою 8-бітного таймера-лічильника мікроконтролера.

Предмет дослідження – методи та засоби програмного управління кроковими двигунами на базі мікроконтролерів з використанням 8-бітних таймерів-лічильників для генерації послідовностей керуючих імпульсів.

Проблема, яка вирішується в кваліфікаційній роботі, полягає у відсутності доступних та інтерактивних засобів для вивчення процесів мікроконтролерного керування кроковими двигунами, зокрема використання 8-бітних таймерів-лічильників для генерації керуючих імпульсів.

Область застосування— освітні заклади та програми самостійного навчання, де платформа може бути використана як ефективний засіб для вивчення принципів мікроконтролерного керування кроковими двигунами та розробки вбудованих систем.

Ключові слова: мікроконтролерне керування, навчальна платформа, інтерактивна візуалізація, кроковий двигун, 8-бітний таймер-лічильник, веб-додаток, JavaScript, React, моделювання, генерація імпульсів, алгоритми керування, рухома платформа

ABSTRACT

The explanatory note to 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 76 pages, of which 50 pages are the main part with 16 figures, and includes a list of 20 sources and two appendices.

The aim of the qualification work is to create an educational platform for visualizing the process of controlling a stepper motor, with a special emphasis on using an 8-bit timer-counter to generate pulses with the correct frequency.

The object of research is the process of controlling stepper motors for turning the wheels of a moving platform using an 8-bit timer-counter of a microcontroller.

The subject of research is the methods and means of software control of stepper motors based on microcontrollers, with a special emphasis on the use of 8-bit timer-counters for generating control pulses.

The problem addressed in the qualification work lies in the absence of accessible and interactive means for studying the processes of microcontroller-based control of stepper motors, particularly the use of 8-bit timer-counters for generating control pulses.

The application area includes educational institutions and self-study programs, where the platform can be used as an effective means for studying the principles of microcontroller control of stepper motors and the development of embedded systems.

Keywords: microcontroller control, educational platform, interactive visualization, stepper motor, 8-bit timer-counter, web application, JavaScript, React, modeling, pulse generation, control algorithms, moving platform.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

CTC – Clear To Compare;

IDE – Integral Development Environment (інтегроване середовище розробки);

TC – Timer/Counter (таймер-лічильник);

DIR Direction (напрямок)

ВСТУП

Актуальність роботи. У сучасній робототехніці та автоматизації ефективне та точне керування кроковими двигунами є критично важливим для досягнення високої продуктивності та надійності систем. Особливе значення має управління за допомогою мікроконтролерів, які дозволяють точно контролювати рух крокового двигуна шляхом генерації імпульсів з необхідною частотою. Використання 8-бітного таймера-лічильника мікроконтролера для формування цих імпульсів є ефективним методом досягнення потрібної швидкості та точності руху. У цьому контексті актуальним є створення навчальної платформи для візуалізації процесу керування крковим двигуном, що дозволить студентам та інженерам краще зрозуміти принципи налаштування таймерів та управління двигунами через драйвери технології STEP/DIR.

З розвитком сучасних технологій автоматизації та робототехніки виникає потреба у доступних та наочних інструментах для вивчення принципів керування кроковими двигунами. Крокові двигуни широко застосовуються для створення точних та керованих рухів у робототехнічних системах, зокрема для повороту коліс рухомих платформ. Розробка навчальної платформи, яка візуалізує процес управління крковим двигуном за допомогою 8-бітного таймера-лічильника, сприятиме глибшому розумінню налаштування таймерів для генерації імпульсів з правильною частотою, необхідних для точного позиціонування двигуна через драйвери STEP/DIR.

Метою дослідження є створення навчальної платформи для візуалізації процесу керування крковим двигуном, з особливим акцентом на використанні 8-бітного таймера-лічильника для генерації імпульсів з правильною частотою. Ця платформа допоможе спростити розуміння налаштування таймерів та керування кроковими двигунами через драйвери

технології STEP/DIR, забезпечуючи студентам наочність та зрозумілість процесу.

Об’єкт дослідження – процес керування кроковими двигунами для повороту коліс рухомої платформи за допомогою 8-бітного таймера-лічильника мікроконтролера.

Методи дослідження: даному дослідженні використано мову програмування JavaScript для розробки веб-додатку, що слугує навчальною платформою. JavaScript обрано через його гнучкість та можливості інтеграції з веб-інтерфейсами, що спрощує візуалізацію процесів керування. Основний акцент зроблено на моделюванні роботи 8-бітного таймера-лічильника мікроконтролера ATmega2560 для генерації імпульсів з потрібною частотою, необхідних для керування крековим двигуном моделі 17HS4401S через драйвер технології STEP/DIR.

Предмет дослідження – методи та засоби програмного управління кроковими двигунами на основі мікроконтролерів, з особливим акцентом на використанні 8-бітних таймерів-лічильників для генерації керуючих імпульсів.

Завдання дослідження

1. Аналіз літератури, присвяченої будові та керуванню кроковими двигунами.
2. Огляд мікроконтролерних платформ для керування кроковими двигунами.
3. Моделювання роботи крокового двигуна.
4. Розробка програмної моделі мікроконтролерного керування кроковими двигунами.
5. Використання моделі для керування рухомою платформою для повороту коліс за допомогою двох двигунів.
6. Тестування моделі.
7. Аналіз результатів тестування та виявлених помилок.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ

У цьому розділі розглядаються основні теоретичні аспекти, пов'язані з керуванням кроковими двигунами за допомогою мікроконтролерів, а також аналізуються сучасні підходи та технології, що використовуються у цій галузі. Основна увага приділяється будові та принципам роботи крокових двигунів, методам їх керування, огляду мікроконтролерних платформ, драйверам технології STEP/DIR, а також використанню 8-бітних таймерів-лічильників у мікроконтролерах.

1.1 Крокові двигуни: будова та принцип роботи

Кроковий двигун — це двигун, який повертається на певний кут, що задається конструкцією при виготовленні. Цей кут називають кроком, тому двигун називається кроковим. Основна відмінність крокових двигунів від звичайних двигунів постійного струму полягає в тому, що при подачі живлення на двигун постійного струму його вал починає обертатися, а при подачі живлення на кроковий двигун він просто повертається на один крок і зупиняється. Щоб повернути його на наступний крок, потрібно підключити обмотки в іншій послідовності, після чого він знову зробить один крок і так далі.

а) б)

Рисунок 1.1 Позначення уніполярного та біполярного крокових двигунів.

Біполярний крокoвий двигун має дві обмотки та, відповідно, чотири виводи (див. рис. 1.1 б). Уніполярний крокoвий двигун має чотири обмотки і може мати вісім (див. рис. 1.1 а), шість або п'ять виводів, якщо обмотки з’єднані відповідними кінцями. Слово «біполярний» означає, що на обмотки подається живлення з подвійною полярністю, тобто в один момент на одному виводі знаходиться «+», а на іншому «-». Уніполярний двигун, навпаки, потребує лише однієї полярності живлення, що робить схему управління простішою і дешевшою [1].

1.1.1 Принцип роботи уніполярних крокових двигунів

Розглянемо, як працює уніполярний двигун (див. рис. 1.2) в режимі повного кроку з однією фазою на крок. Підключаючи по черзі «-» живлення до клем A-, B-, C-, D-, ми отримаємо обертання за годинниковою стрілкою, оскільки ротор двигуна є постійним магнітом, а обмотки при проходженні через них струму перетворюються на електромагніти. Оскільки в кожен момент часу увімкнена лише одна обмотка, цей режим називається повним кроком з однією ифазою на крок.

\

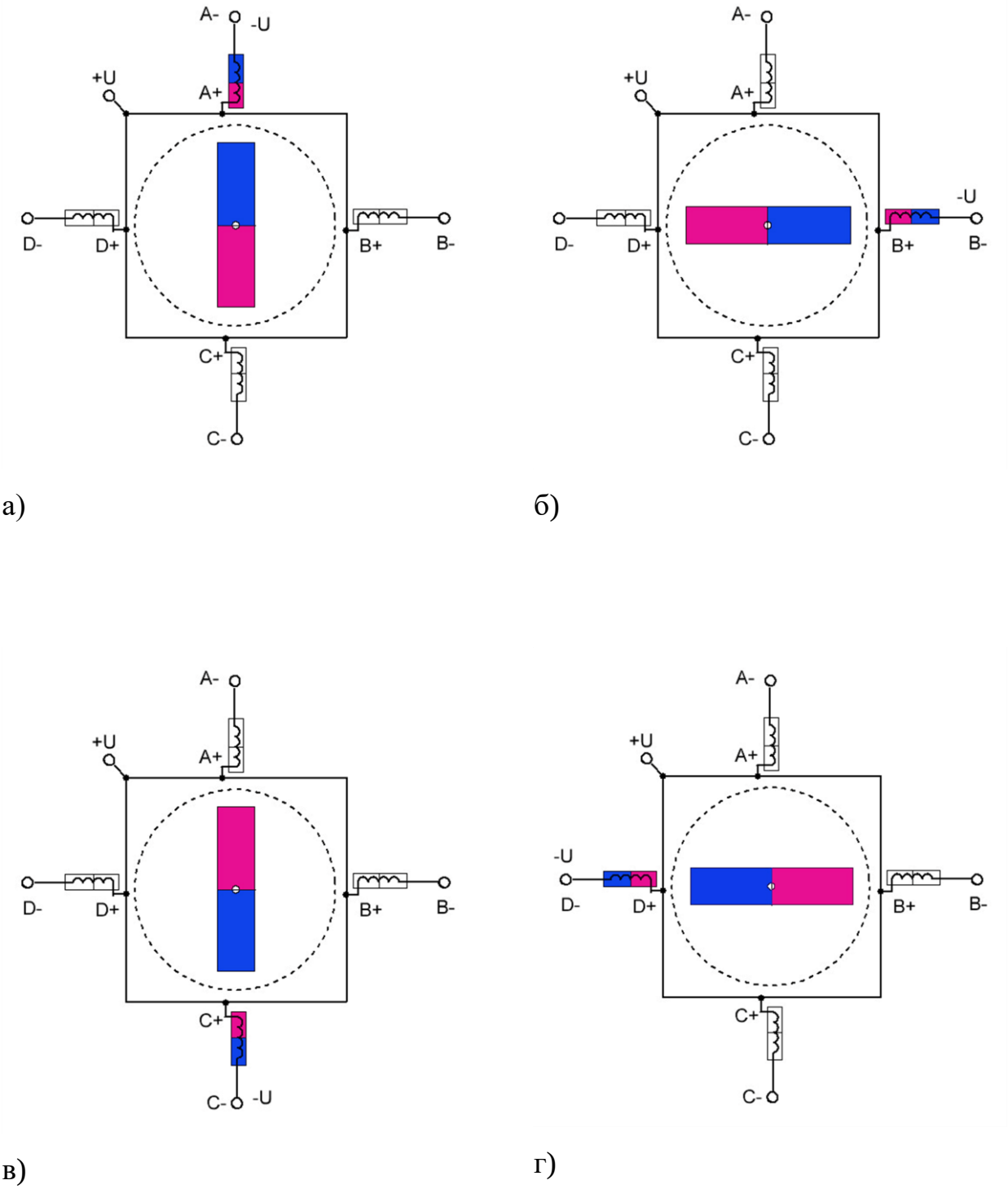
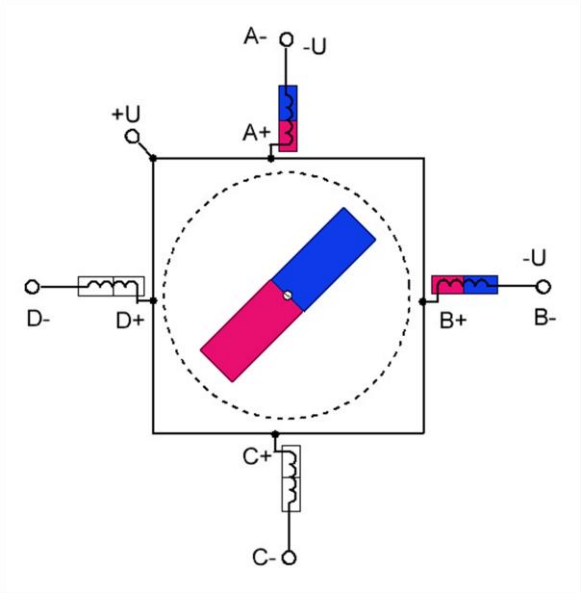
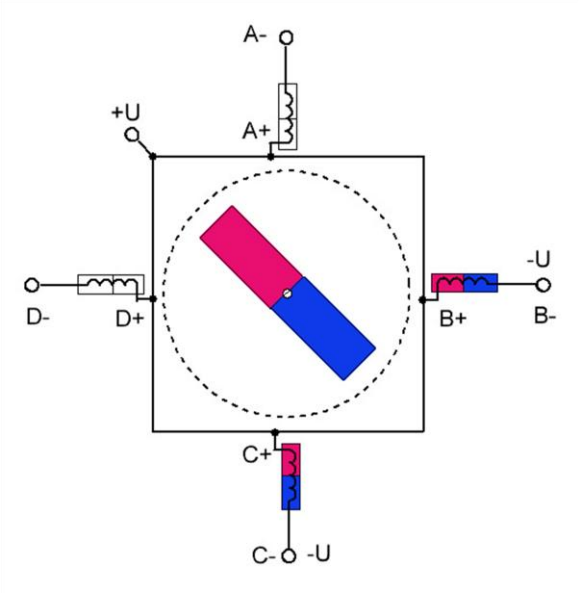


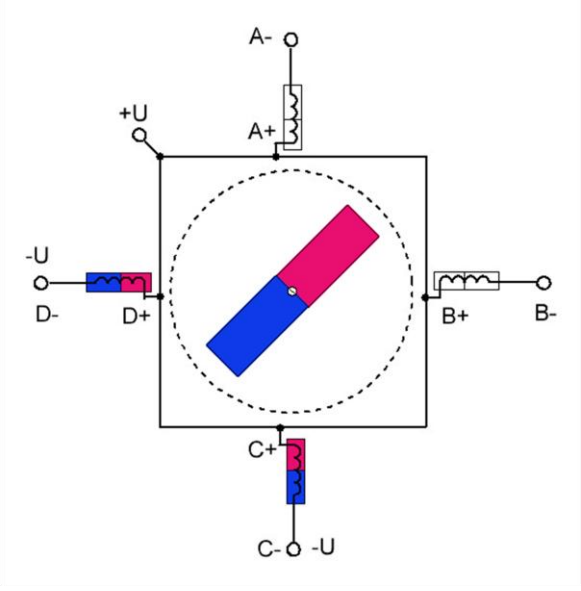
Рисунок 1.2 Принцип роботи уніполярного крокового двигуна в режимі повного кроку з однією фазою на крок.



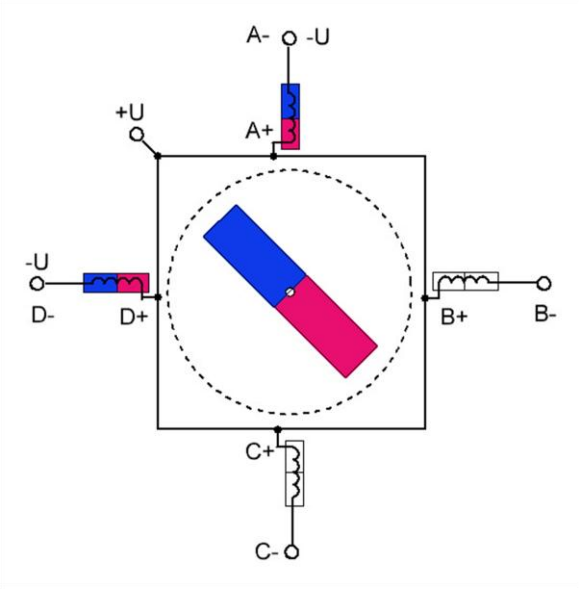
а)



б)



в)



г)

Рисунок 1.3 Принцип роботи уніполярного крокового двигуна в режимі повного кроку з двома фазами на крок.

На рисунку 1.3 зображений принцип роботи уніполярного крокового двигуна в режимі повного кроку з двома фазами на крок. Якщо подати негативний потенціал на дві клеми, наприклад, A- і B- (див. рис. 1.3.а), відповідні обмотки створять магнітне поле однакового напрямку, що призведе до того, що вони обидві притягнуть постійний магніт ротора з однаковою силою, і ротор стане точно посередині між ними. Змінюючи на наступному кроці клеми на B-, C- (див. рис. 1.3.б), ми досягнемо повороту ротора на 90 електричних градусів. У реальному двигуні поворот ротора на один крок означає обертання на кут від часток градуса до кількох десятків градусів, який точно визначається конструкцією двигуна. Режим повного кроку з двома фазами є більш переважним порівняно з режимом з однією фазою (див. рис. 1.2), оскільки при цьому збільшується момент сили, що розвивається валом двигуна. При цьому, звісно, збільшується споживаний струм системи.

Для точнішого позиціонування вала двигуна може застосовуватися напівкроковий режим. Це режим, у якому чергуються два методи управління: з однією фазою на крок і з двома фазами на крок. Наприклад, спочатку підключена клема A-, потім підключаємо клеми A- і B-, тобто ротор повернеться на 45 електричних градусів. Потім підключаємо лише B-. Продовжуючи таким чином перемикати обмотки двигуна, ми збільшимо кількість кроків на один оберт у два рази, тим самим підвищивши точність позиціонування (при цьому, звісно, зменшиться момент сили).

Для ще точнішого позиціонування вала двигуна можна створити драйвер, який підтримуватиме мікрокрокові режими. Це режими, в яких повний крок може бути поділений на 4, 8, 16, або 32 частини, що збільшує точність повороту вала у відповідне число разів (проте і в цьому випадку зменшиться момент сили, що розвивається на валу двигуна).

Оскільки в уніполярному двигуні немає необхідності змінювати полярність живлення, такі двигуни стали дуже популярними, оскільки схема

управління в уніполярному випадку виявляється простішою та дешевшою, ніж у біполярному.

1.1.2 Принцип роботи біполярних крокових двигунів

Біполярний кроковий двигун дозволяє розвинути більший момент сили, ніж уніполярний, за однакових умов. Тому біполярні крокові двигуни також отримали дуже широке розповсюдження.

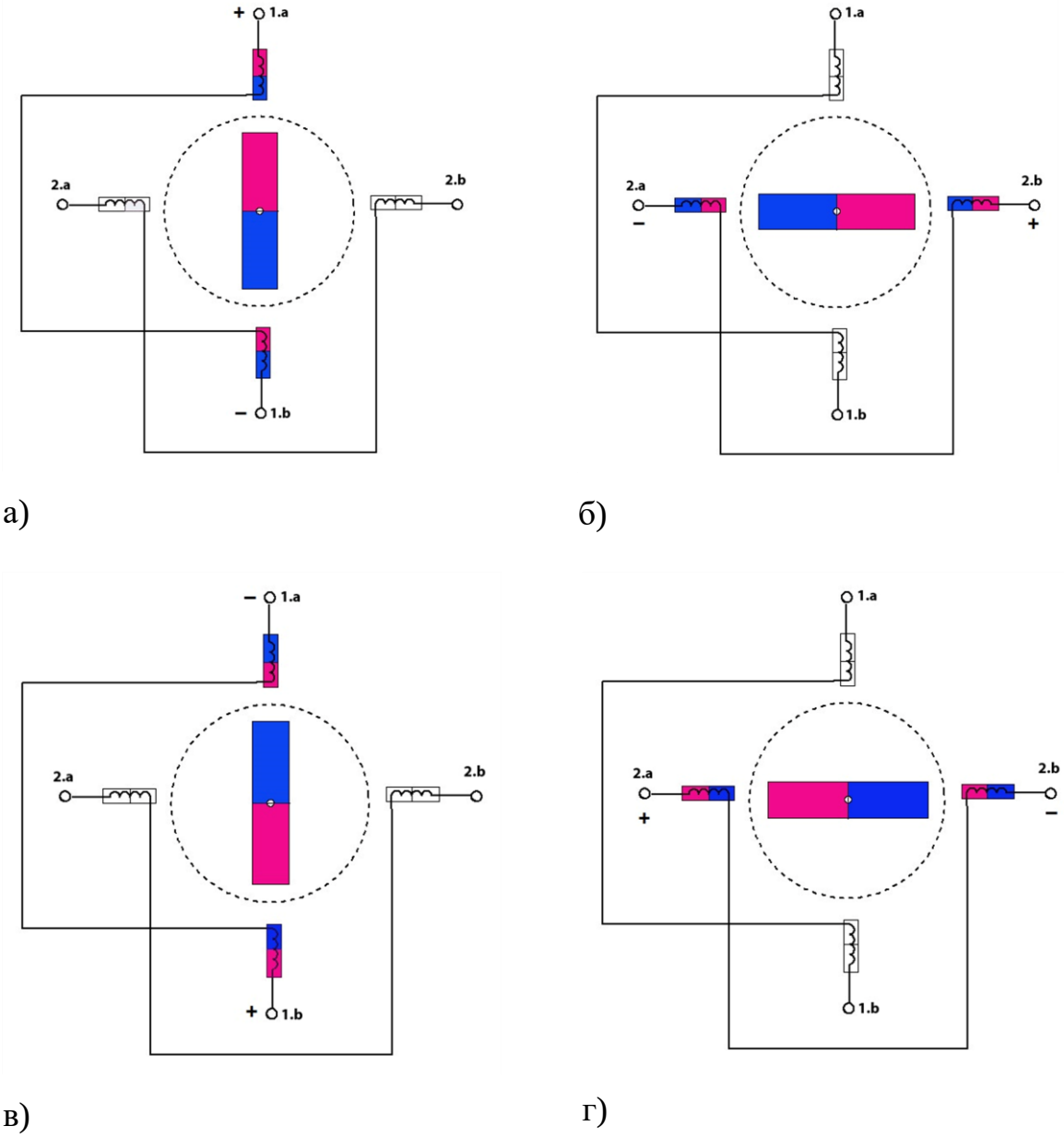


Рисунок 1.4 Принцип роботи біполярного крокового двигуна в режимі повного кроку з однією фазою на крок.

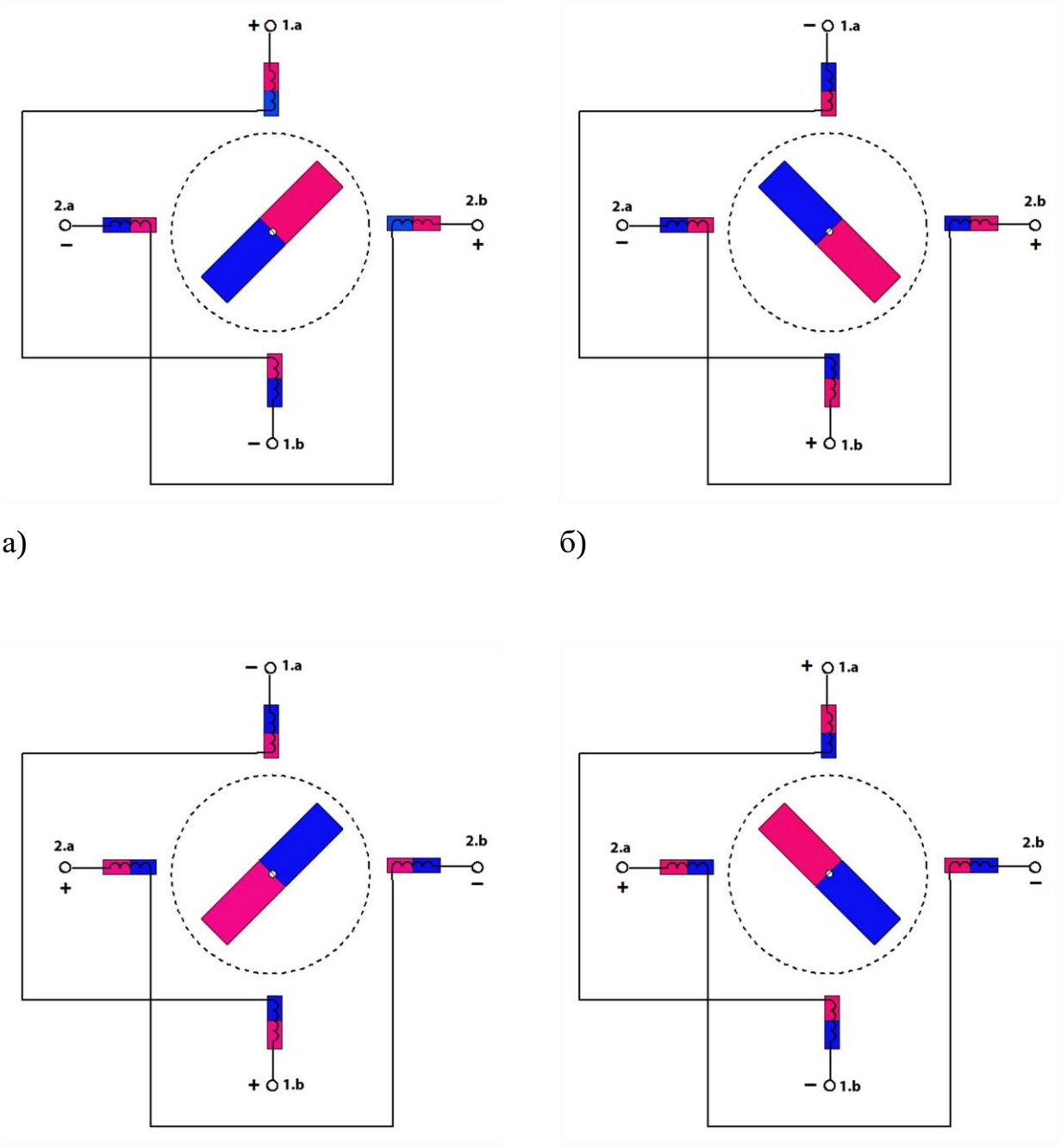


Рисунок 1.5 Принцип роботи біполярного крокового двигуна в режимі повного кроку з двома фазами на крок..

На рисунку 1.4 зображено принцип роботи біполярного крокового двигуна в режимі повного кроку з однією фазою на крок. Якщо в уніполярному крововому двигуні є чотири обмотки, то в біполярному їх дві, але вони складаються з кількох частин (на рисунку – з двох частин). Вони розташовані

навпроти одна одної, що призводить до того, що коли через обмотку проходить струм, одна частина притягує північний полюс магніту ротора, а інша – південний, таким чином досягається більший крутний момент. Наступна важлива відмінність біполярного крокового двигуна від уніполярного полягає в тому, що на клемах обмоток змінюється полярність джерела живлення. Це і є суть поняття «біполярний». Так, на рисунку 1.4.а до клеми 1а підключено «плюс», а до клеми 1б – «мінус», тоді як на рисунку 1.4.в навпаки: до клеми 1а підключено «мінус», а до клеми 1б – «плюс». Те ж саме стосується і рисунків 1.4.б і 1.4.г. З урахуванням цих відмінностей можна сказати, що логіка роботи уніполярного та біполярного крокових двигунів подібна – необхідно по черзі включати відповідну обмотку, щоб магніт ротора повернувся в її бік. Аналогічно на рисунку 1.5 показано, що, включаючи щоразу дві обмотки з різною полярністю, ми досягаємо обертання вала крокового двигуна. Як і в уніполярному двигуні, в біполярному можливі режими напівкроку та мікрокроку, які підвищують точність кута повороту, але при цьому знижують загальний крутний момент.

Уважний аналіз рисунків 1.4 і 1.5 дозволяє зробити висновок, що при використанні уніполярного двигуна достатньо мати ключі одного типу, наприклад, нижнього. А при використанні біполярного двигуна потрібні ключі обох типів, як верхнього, так і нижнього, і найкраще використовувати Н-міст [2].

1.2. Огляд мікроконтролерних платформ для керування кроковими двигунами

Мікроконтролери є ключовим компонентом у системах керування кроковими двигунами, оскільки вони забезпечують необхідну обчислювальну потужність, засоби введення/виведення та можливість програмування для реалізації алгоритмів керування[3]. Існує широкий спектр мікроконтролерних

платформ, які відрізняються за архітектурою, продуктивністю та функціональними можливостями. Розглянемо найбільш популярні з них:

1.2.1. Arduino (ATmega328, ATmega2560)

Arduino — це відкрита апаратна та програмна платформа, яка базується на мікроконтролерах сімейства AVR від компанії Atmel (нині Microchip Technology). Платформи Arduino широко використовуються в освіті, прототипуванні та для створення любительських проектів завдяки своїй доступності, простоті використання та великій спільноті користувачів.

- **Arduino Uno (ATmega328P)**: має 14 цифрових входів/виходів, з яких 6 можуть використовуватися як виходи ШІМ, та 6 аналогових входів. Частота роботи мікроконтролера становить 16 МГц, обсяг флеш-пам'яті — 32 КБ.
- **Arduino Mega (ATmega2560)**: оснащена більш потужним мікроконтролером ATmega2560, має 54 цифрових входи/виходи, з яких 15 можуть використовуватися як виходи ШІМ, та 16 аналогових входів. Частота роботи також 16 МГц, обсяг флеш-пам'яті — 256 КБ [4].

Arduino Mega з ATmega2560, використаний у даному дослідженні, забезпечує достатню кількість портів введення/виведення та пам'яті для реалізації складних алгоритмів керування кроковими двигунами та взаємодії з іншими компонентами системи. Завдяки простоті програмування на мові, схожій на C/C++, та великій кількості бібліотек, ця платформа є ідеальною для навчальних та дослідницьких проектів.

1.2.2. STM32

STM32 — це серія 32-бітних мікроконтролерів на базі архітектури ARM Cortex-M від компанії STMicroelectronics. Вони відрізняються високою продуктивністю, широким набором периферійних пристроїв та енергетичною

ефективністю, що робить їх підходящими для складних та вимогливих до ресурсів застосувань.

- **STM32F103**: один з найпопулярніших представників серії, працює на частоті до 72 МГц, має до 512 КБ флеш-пам'яті та 64 КБ SRAM, оснащений численними таймерами, АЦП, ШІМ та іншими периферійними модулями.
- **STM32F4** та **STM32F7**: більш потужні серії з тактовою частотою до 216 МГц, підтримують розширені можливості, такі як графічні інтерфейси, Ethernet, USB OTG та інші.

STM32 мікроконтролери використовуються у складніших системах, де потрібна висока швидкодія та обчислювальна потужність, наприклад, у промисловій автоматизації, робототехніці, обробці сигналів та зображень [5].

1.2.3. PIC Microcontrollers

PIC — серія мікроконтролерів від компанії Microchip Technology, яка включає широкий спектр моделей від 8-бітних до 32-бітних пристроїв. PIC мікроконтролери відомі своєю надійністю, різноманітністю та можливістю використання в промислових застосуваннях.

- **PIC16F**: 8-бітні мікроконтролери з базовим набором функцій, підходять для простих завдань керування та автоматизації [6].
- **PIC18F**: більш просунуті 8-бітні мікроконтролери з розширеними можливостями, такими як більший обсяг пам'яті, додаткові периферійні модулі.
- **PIC32**: 32-бітні мікроконтролери на базі архітектури MIPS, призначені для високопродуктивних застосувань.

PIC мікроконтролери широко використовуються в промислових системах керування, медичних приладах, автомобільній електроніці та інших галузях, де потрібна надійність та довгострокова підтримка.

1.2.4. Порівняння та вибір платформи

Вибір мікроконтролерної платформи залежить від вимог конкретного проекту:

- **Простота використання та навчання**: Arduino (ATmega328, ATmega2560) є найкращим вибором для початківців та навчальних проектів завдяки простому середовищу розробки та великій кількості навчальних матеріалів.
- **Висока продуктивність та розширені можливості**: STM32 мікроконтролери підходять для проектів, де потрібна висока швидкодія, складні алгоритми обробки даних та взаємодія з різноманітними периферійними пристроями.
- **Промислові застосування та надійність**: PIC мікроконтролери є відмінним вибором для промислових систем, де важливими є надійність, довгострокова підтримка та широкий спектр моделей з різними характеристиками.

У даному дослідженні використано мікроконтролер **ATmega2560**, який поєднує в собі достатню продуктивність, розширені можливості та простоту використання, що робить його оптимальним для розробки навчальної платформи з керування кроковими двигунами.

1.2.5. Використання крокових двигунів у сучасних технологіях

Крокові двигуни є невід'ємною частиною багатьох сучасних технологій завдяки своїм перевагам у точності, надійності та простоті керування. Розглянемо детальніше їхнє застосування в різних галузях.

Промисловість

- **Верстати з числовим програмним керуванням (ЧПК)**: крокові двигуни використовуються для точного переміщення інструментів та заготовок по різних осях, забезпечуючи високу точність обробки матеріалів.
- **3D-принтери**: керування рухом друкуючої головки та платформи здійснюється за допомогою крокових двигунів, що дозволяє створювати об'єкти з високою роздільною здатністю.
- **Роботизовані маніпулятори**: забезпечують точне позиціонування та рух у виробничих процесах, таких як зварювання, пакування, монтаж компонентів.

Робототехніка

- **Мобільні роботи**: крокові двигуни використовуються для керування колесами або гусеницями, забезпечуючи точний контроль швидкості та напрямку руху.
- **Маніпулятори та роботизовані руки**: дозволяють виконувати складні операції з високою точністю, такі як захоплення, переміщення та складання дрібних деталей.
- **Дрони та безпілотні літальні апарати**: в окремих випадках крокові двигуни застосовуються для керування камерою або іншими механізмами.

Медицина

- **Хірургічні роботи**: забезпечують точність та стабільність під час проведення операцій, особливо в мікрохірургії.

- **Дозатори ліків**: крокові двигуни контролюють точний об'єм та швидкість подачі медикаментів, що критично у фармакології та анестезії.
- **Діагностичне обладнання**: у томографах, рентгенівських апаратах та інших пристроях для точного позиціонування компонентів.

Побутова техніка

- **Принтери та сканери**: забезпечують переміщення каретки з друкуючою головкою або скануючим елементом з високою точністю.
- **Автоматизація будинку**: системи керування жалюзі, автоматичні двері, розумні меблі.
- **Аудіо та відео обладнання**: у механізмах приводів CD/DVD, проекторів та камер.

Автомобільна промисловість

- **Регулювання сидінь та дзеркал**: забезпечують комфорт та точне налаштування положення.
- **Системи кондиціонування та вентиляції**: керування заслінками та клапанами для регулювання потоків повітря.
- **Приладова панель**: у деяких випадках для керування стрілками аналогових приладів.

Переваги крокових двигунів у цих застосуваннях:

- **Висока точність позиціонування**: можливість точно контролювати положення без необхідності у зворотному зв'язку.
- **Простота керування**: використання цифрових сигналів спрощує інтеграцію з мікроконтролерами та системами керування.

- **Високий крутний момент на низьких швидкостях**: дозволяє ефективно працювати в режимах, де потрібна повільна та стабільна робота.
- **Надійність та довговічність**: відсутність щіток та колекторів зменшує знос та підвищує термін служби.

Виклики та обмеження:

- **Резонанс та вібрації**: крокові двигуни можуть створювати вібрації на певних швидкостях, що може вимагати додаткового демпфірування або налаштування.
- **Втрати моменту на високих швидкостях**: крутний момент може знижуватися з підвищенням швидкості обертання.
- **Споживання енергії**: двигуни можуть споживати струм навіть у стані спокою, щоб утримувати позицію [7].

Висновки за розділом 1

Проведений аналіз літератури та технологічних рішень свідчить про важливу роль крокових двигунів та мікроконтролерів у сучасних системах автоматизації та робототехніки. Крокові двигуни, завдяки своїм унікальним властивостям точного та надійного керування, знаходять широке застосування в різних галузях, від промисловості до побутової техніки.

Мікроконтролери, оснащені таймерами-лічильниками та іншими периферійними модулями, надають гнучкі можливості для реалізації складних алгоритмів керування. Використання драйверів технології STEP/DIR спрощує апаратну частину системи та знижує вимоги до обчислювальних ресурсів мікроконтролера, що особливо важливо при розробці компактних та енергоефективних пристроїв.

Різноманітність мікроконтролерних платформ дозволяє обрати оптимальне рішення для конкретного проекту, враховуючи вимоги до продуктивності, складності та вартості. У даному дослідженні вибір мікроконтролера ATmega2560 обумовлений його достатніми ресурсами та простотою використання для навчальних цілей.

Таким чином, поєднання крокових двигунів, сучасних мікроконтролерів та драйверів STEP/DIR створює потужний інструментарій для розробки високоточних та надійних систем керування, що відкриває широкі перспективи для подальшого розвитку автоматизації та робототехніки.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ КРОКОВИМИ ДВИГУНАМИ ДЛЯ ПОВОРОТУ КОЛІС РУХОМОЇ ПЛАТФОРМИ

2.1. Вибір та обґрунтування компонентів системи

Для реалізації системи мікроконтролерного керування кроковими двигунами необхідно обрати оптимальні компоненти, які забезпечать надійну та ефективну роботу системи. Правильний вибір кожного елемента впливає на загальну продуктивність, точність та стабільність роботи.

2.1.1. Вибір крокового двигуна

Для реалізації системи керування обрано біполярний кроковий двигун моделі 17HS4401S (див. рис 2.1). Цей двигун є популярним у сферах автоматизації та робототехніки завдяки своїм технічним характеристикам та доступності [8].

Рисунок 2.1 Зовнішній вигляд крокового двигуна 17HS4401

Електричні параметри:

- Номінальна напруга живлення: 2,5 В.
- Струм фази: 1,7 А.
- Опір обмотки: 1,5 Ом.
- Індуктивність обмотки: 3,0 мГн.

Механічні параметри:

- Кут кроку: 1,8° (200 кроків на повний оберт).
- Утримуючий крутний момент: 0,4 Н·м.
- Момент інерції ротора: 54 г·см².
- Розміри:
  - Довжина корпусу: 40 мм.
  - Діаметр валу: 5 мм.
  - Стандарт фланця кріплення: NEMA17.

Особливості роботи та застосування

Переваги:

- Висока точність позиціонування: Кут кроку 1,8° дозволяє забезпечити точне переміщення, що важливо для прецизійних задач.
- Надійність та довговічність: Відсутність рухомих електричних контактів зменшує знос компонентів.
- Простота керування: Можливість прямого цифрового керування спрощує інтеграцію з мікроконтролерами.

Недоліки:

- Обмежена швидкість обертання: На високих швидкостях крутний момент знижується.

- Споживання енергії: Двигун споживає струм навіть у стані утримання позиції.
- Відсутність зворотного зв'язку: Без додаткових сенсорів неможливо визначити пропуск кроків.

Сфери застосування:

- 3D-принтери та ЧПК-верстати
- Роботизовані маніпулятори
- Автоматизовані системи подачі та позиціонування

2.1.2. Вибір драйвера крокового двигуна

Для керування біполярними кроковими двигунами широко використовуються драйвери з інтерфейсом STEP/DIR, такі як A4988 (див. рис 2.2) та DRV8825.

Рисунок 2.2 Зовнішній вигляд драйвера A4988

Драйвер A4988:

- Електричні характеристики:
  - Максимальний струм фази: до 2 А (з охолодженням).
  - Робоча напруга живлення двигуна (VMOT): 8–35 В.
- Функціональні можливості:
  - Режими мікрокрокування: повний крок, 1/2, 1/4, 1/8, 1/16 кроку.

- Захисні функції: захист від перегріву, перевантаження по струму, короткого замикання.
- Інтерфейс керування: простий двосигнальний інтерфейс (STEP, DIR).

Обґрунтування вибору драйвера A4988

Сумісність з двигуном та мікроконтролером:

- Струм фази: A4988 може забезпечити до 2 А, що відповідає вимогам двигуна 17HS4401S (1,7 А).
- Напруга живлення: Драйвер підтримує напругу VMOT до 35 В, що достатньо для оптимальної роботи двигуна.
- Інтерфейс STEP/DIR: Легко інтегрується з мікроконтролером ATmega2560.

Підтримка режимів мікрокрокування:

- Мікрокрокування до 1/16 кроку: Це дозволяє підвищити точність та плавність руху.

Доступність та вартість:

- Популярність: A4988 є доступним та широко використовуваним, що спрощує його придбання та інтеграцію.
- Вартість: Бюджетне рішення для навчальних та прототипних проектів.

Драйвер A4988 має наступні основні пінові виходи (див. рис 2.3):

- VMOT та GND: Живлення двигуна.
- VDD та GND: Живлення логіки драйвера (3,3 В або 5 В).
- 1A, 1B, 2A, 2B: Підключаються до обмоток двигуна.
- STEP та DIR: Сигнали керування від мікроконтролера.

- MS1, MS2, MS3: Встановлюють режим мікрокрокування.
- EN (Enable): Активує або деактивує драйвер.

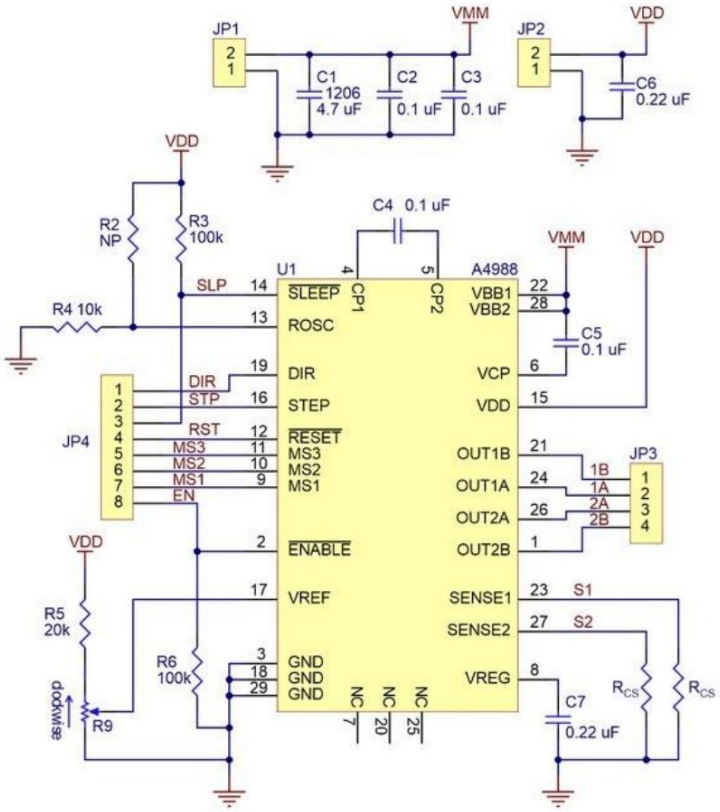


Рисунок 2.3 Схема драйвера A4988

Режими мікрокрокування вибираються шляхом встановлення логічних рівнів на входах MS1, MS2 та MS3:

- Повний крок: MS1=0, MS2=0, MS3=0.
- 1/2 кроку: MS1=1, MS2=0, MS3=0.
- 1/4 кроку: MS1=0, MS2=1, MS3=0.
- 1/8 кроку: MS1=1, MS2=1, MS3=0.
- 1/16 кроку: MS1=1, MS2=1, MS3=1.

2.1.3. Вибір мікроконтролера

У процесі розробки системи керування кроковим двигуном виникла необхідність у виборі відповідного мікроконтролера, який би забезпечив

необхідну функціональність, продуктивність та гнучкість для реалізації поставлених завдань. Після детального аналізу доступних на ринку мікроконтролерів було обрано ATmega2560 (див. рис 2.4), що належить до сімейства AVR 8-бітних мікроконтролерів, розроблених компанією Microchip Technology [9].

Рисунок 2.4 Зовнішній вигляд мікроконтролера ATmega2560

ATmega2560 —має такі характеристики:

- Пам'ять:
  - Флеш-пам'ять: 256 КБ для зберігання програм.
  - SRAM: 8 КБ для даних.
  - EEPROM: 4 КБ для постійних даних.
- Периферійні модулі:
  - Таймери-лічильники:
    - 2 × 8-бітних таймери (Timer0, Timer2).
    - 4 × 16-бітних таймери (Timer1, Timer3, Timer4, Timer5).
  - Аналого-цифровий перетворювач (АЦП): 16 каналів, 10-бітний.

- Комунікаційні інтерфейси:
  - UART: 4 канали.
  - SPI: 5 модулів.
  - I2C (TWI): 1 модуль.
- Інші модулі:
  - Watchdog Timer.
  - JTAG для налагодження.

Причини вибору ATmega2560

- Достатня кількість портів введення/виведення:
  - 54 цифрових входи/виходи, з яких 15 можуть використовуватися для ШІМ.
- Підтримка необхідних периферійних модулів:
  - Наявність кількох таймерів-лічильників дозволяє реалізувати точне керування сигналами.
- Великий обсяг пам'яті:
  - 256 КБ флеш-пам'яті дозволяють розробляти складні програми.
- Широка підтримка спільноти та наявність бібліотек:
  - Arduino Mega 2560 базується на цьому мікроконтролері, що спрощує розробку та тестування.

Опис таймерів-лічильників та їх режимів роботи

- 8-бітні таймери (Timer0, Timer2):
  - Режими роботи:
    - Normal Mode.
    - CTC Mode (Clear Timer on Compare Match).
    - Fast PWM Mode.
    - Phase Correct PWM Mode.
- 16-бітні таймери (Timer1, Timer3, Timer4, Timer5):

- Режими роботи:
  - Normal Mode.
  - CTC Mode.
  - Fast PWM Mode.
  - Phase and Frequency Correct PWM Mode.
  - Input Capture Mode.

2.2. Схема підключення крокового двигуна та апаратна реалізація

Мікроконтролер ATmega2560 (Arduino Mega 2560):

- Виконує функції керування кроковим двигуном.
- Підключається до комп'ютера через USB для живлення та програмування.
- Забезпечує генерування сигналів керування для драйвера.

Драйвер крокового двигуна A4988:

- Призначений для перетворення сигналів від мікроконтролера у команди для крокового двигуна.
- Контакти:
  - VMOT і GND — живлення двигуна.
  - VDD і GND — живлення логічної частини драйвера.
  - STEP і DIR — входи для сигналів, що визначають кількість кроків та напрямок обертання.
  - MS1, MS2, MS3 — налаштування режимів мікрокрокування.
  - EN (Enable), RST (Reset) і SLP (Sleep) — додаткові входи для керування режимами роботи.
  - 1A, 1B, 2A, 2B — виходи для підключення обмоток двигуна.

Кроковий двигун 17HS4401S:

- Біполярний двигун з 4-провідним підключенням (дві фази: A і B).
- Забезпечує точне позиціонування завдяки кроку 1,8°.

Макетна плата (Breadboard):

- Використовується для зручного з'єднання компонентів без пайки.
- Розміщено компоненти та з'єднання між ними.

Зовнішнє джерело живлення:

- Подає напругу на драйвер для живлення двигуна через контакти VMOT і GND.
- Забезпечує стабільну роботу двигуна, оскільки живлення від мікроконтролера може бути недостатнім.

Підключення драйвера A4988 до мікроконтролера ATmega2560 (див. рис 2.5 – 2.6):

- STEP (A4988) → Digital Pin 2 (PD2) на ATmega2560.
- DIR (A4988) → Digital Pin 3 (PD3) на ATmega2560.
- GND (A4988) → GND на ATmega2560.

Підключення драйвера A4988 до крокового двигуна 17HS4401S:

- 1A і 1B (A4988) → обмотка A двигуна.
- 2A і 2B (A4988) → обмотка B двигуна.
- Важливо правильно визначити пари обмоток двигуна за допомогою омметра або технічної документації.

Підключення драйвера A4988 до джерела живлення:

- VMOT (A4988) → позитивний вивід зовнішнього джерела живлення (наприклад, 12 В).
- GND (A4988) → негативний вивід зовнішнього джерела живлення.

- VDD (A4988) → 5 В від мікроконтролера ATmega2560 (живлення логіки драйвера).
- Між контактами VMOT і GND рекомендується встановити конденсатор (наприклад, 100 µF) для фільтрації пульсацій напруги.

Додаткові підключення:

- RST і SLP (A4988) з'єднані між собою та підключені до 5 В для забезпечення нормальної роботи драйвера (запобігає переходу в режими сну або скидання) [10].

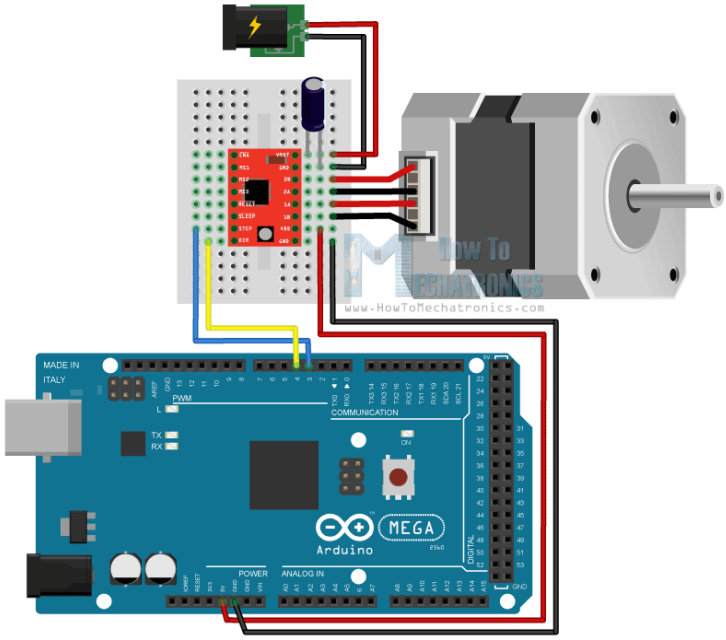


Рисунок 2.5 Схема підключення одного крокового двигуна

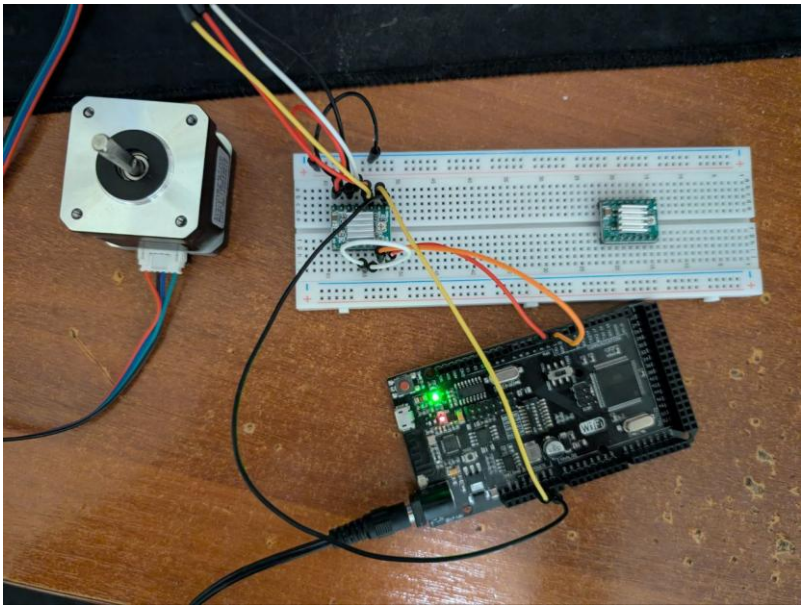


Рисунок 2.6 Апаратна реалізація

2.3. Поворот рухомої платформи за допомогою двох поворотних коліс

Для коректного повороту платформи необхідно забезпечити, щоб всі колеса рухалися по траєкторіях, які перетинаються в одній точці — центрі повороту (див. рис. 2.7). У цій схемі видно, що внутрішнє та зовнішнє колеса повинні повертатися на різні кути, щоб рухатися по концентричних дугах різного радіусу. Це забезпечує плавний та коректний поворот без бокового ковзання.Це запобігає пробуксовуванню та небажаному зносу шин .

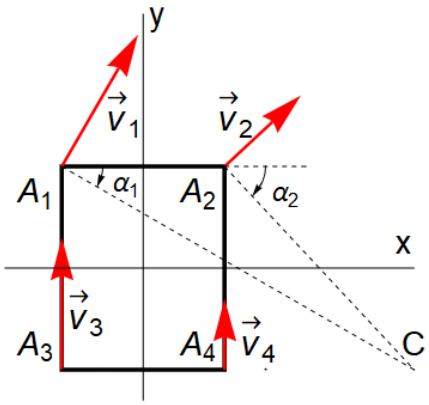


Рисунок 2.7 – Кінематична схема для розрахунку кутів повороту за допомогою двох коліс. Поворот праворуч.

Виведемо формулу для залежності центру повороту та кутів повороту колес. Виберемо систему відліку таким чином, щоб її початок координат збігався з геометричним центром рухомої платформи. При цьому координати точок A_i є

$$A_1(-a,b),\ A_2(a,b),\ A_3(-a,-b),\ A_4(a,-b).$$

Тоді рівняння A_1C и A_2C визначаються виразами

$$\begin{aligned} A_1C: &\qquad y - A_{1y} = (x - A_{1x})tg\alpha_1 \\ A_2C: &\qquad y - A_{2y} = (x - A_{2x})tg\alpha_2. \end{aligned} \tag{2.1}$$

З геометричних міркувань ясно, що ордината центру повороту т.C є

$$C_y = -b.$$

З формули 2.1 для точки C можна написати

$$\begin{aligned} ctg\alpha_1 &= \frac{C_x - A_{1x}}{C_y - A_{1y}} \\ ctg\alpha_2 &= \frac{C_x - A_{2x}}{C_y - A_{2y}} \end{aligned} \tag{2.2}$$

З урахуванням формули 2.1 маємо із формули 2.2

$$ctg\alpha_1 - ctg\alpha_2 = \frac{C_x + a}{-b - b} - \frac{C_x - a}{-b - b}$$

По суті, ми отримали добре відому формулу Акермана, але з урахуванням знаків кутів α_1, α_2 [11].

У випадку з чотирьох колісною рухомою платформою з двома поворотними колесами нам знадобиться 2 крокових двигуна підключених за схемою зображеною на рисунку 2.8.

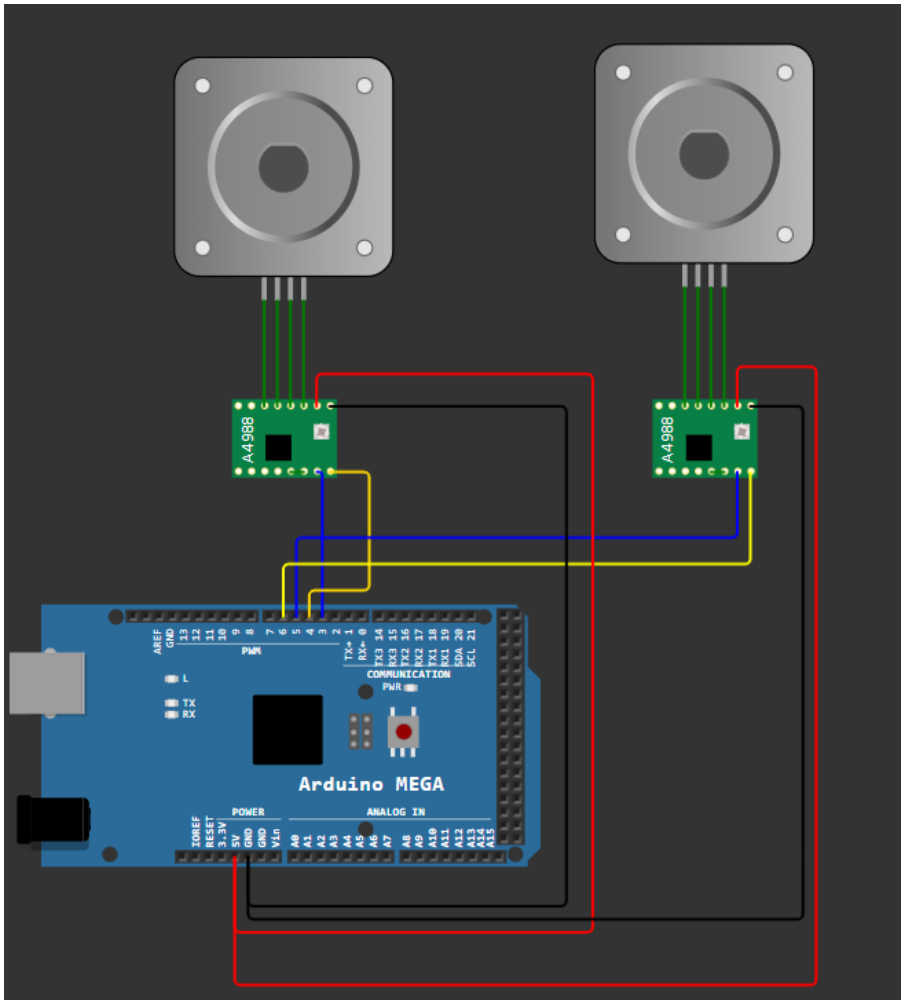


Рисунок 2.8 – Схема підключення двох крокових двигунів

Алгоритм керування поворотом

1. **Введення параметрів**: Користувач або система задає бажаний радіус повороту R або кут повороту платформи.
2. **Обчислення кутів повороту коліс**: за заданими параметрами використовуючи формули 2.2 вираховується кути на які повинні повернути крокові двигуни

3. **Налаштування крокових двигунів**: Генеруємо сигнали керування для встановлення коліс під кутами α1та α2.

Висновки до розділу 2

У цьому розділі здійснено детальний аналіз і обґрунтування вибору компонентів для системи мікроконтролерного керування кроковими двигунами, а також описано їх апаратну реалізацію. Крековий двигун моделі 17HS4401S було обрано завдяки його високій точності, надійності, доступності та оптимальним технічним характеристикам, що робить його придатним для використання в автоматизації, 3D-друці, робототехніці та інших сферах. Драйвер A4988 забезпечує стабільне керування двигуном завдяки підтримці мікрокрокування, захисним функціям і простому інтерфейсу підключення до мікроконтролера. Його технічні можливості дозволяють ефективно керувати двигуном у різних режимах.

Мікроконтролер ATmega2560 обрано завдяки його широким можливостям, достатньому об’єму пам’яті, багатій периферії та підтримці спільноти. Його продуктивності достатньо для реалізації алгоритмів керування двигуном.

Розроблена апаратна схема враховує всі аспекти підключення, включаючи живлення, налаштування режимів роботи драйвера, підключення обмоток двигуна та стабілізацію напруги. Використання макетної плати дозволяє легко здійснювати тестування і внесення змін у схему. У цілому, система є універсальною, надійною і доступною, що робить її придатною як для навчальних, так і для прикладних завдань автоматизації.

РОЗДІЛ 3
РОЗРОБКА НАВЧАЛЬНОЇ ПЛАТФОРМИ ДЛЯ ВІЗУАЛІЗАЦІЇ ПРОЦЕСУ МІКРОКОНТРОЛЕРНОГО КЕРУВАННЯ КРОКОВИМИ ДВИГУНАМИ

3.1. Обґрунтування вибору засобів розробки

При розробці навчальної платформи для візуалізації процесу мікроконтролерного керування кроковими двигунами особливу увагу було приділено вибору відповідних засобів розробки. Це рішення є ключовим, оскільки від нього залежить ефективність розробки, зручність використання платформи та можливість її подальшого розвитку.

3.1.1. Вибір мови програмування JavaScript

Вибір мови програмування є одним із ключових рішень при розробці програмного забезпечення, особливо коли мова йде про створення навчальної платформи з візуалізацією процесів керування. У даному проекті було обрано **JavaScript** як основну мову програмування з наступних причин:

1. **Веб-орієнтованість та кросплатформеність**: JavaScript є основною мовою програмування для веб-браузерів, що дозволяє створювати додатки, які можуть працювати на будь-якому пристрої з доступом до Інтернету без необхідності встановлення додаткового програмного забезпечення.
2. **Інтерактивність та динамічність**: JavaScript дозволяє створювати інтерактивні користувацькі інтерфейси з миттєвим відгуком на дії користувача. Це критично важливо для навчальної платформи, де користувач повинен бачити результати своїх дій у реальному часі.
3. **Багата екосистема бібліотек та фреймворків**: Існує велика кількість відкритих бібліотек і фреймворків, які полегшують розробку складних

додатків. Зокрема, використання **React** та **react-chartjs-2** значно спрощує процес розробки інтерфейсу та візуалізації даних.

4. **Легкість вивчення та популярність**: JavaScript є однією з найбільш популярних мов програмування, що робить її доступною для широкого кола розробників. Це особливо важливо для навчального проекту, який може бути розширений або модифікований іншими студентами або викладачами.
5. **Можливість моделювання складних процесів**: Завдяки асинхронності та потужним засобам обробки подій, JavaScript дозволяє ефективно моделювати роботу мікроконтролерів та таймерів, що необхідно для реалізації емуляції роботи 8-бітного таймера-лічильника.
6. **Інтеграція з веб-технологіями**: JavaScript тісно інтегрується з HTML та CSS, що дозволяє створювати зручні та привабливі інтерфейси користувача [12].

Враховуючи всі ці переваги, JavaScript став оптимальним вибором для реалізації навчальної платформи, яка поєднує в собі інтерактивність, наочність та можливість роботи в веб-середовищі.

3.1.2. Використання бібліотек React та react-chartjs-2

Для створення сучасних веб-додатків з інтерактивним інтерфейсом та динамічною візуалізацією даних було обрано бібліотеки **React** та **react-chartjs-2**.

React — це популярна JavaScript-бібліотека для побудови користувацьких інтерфейсів, розроблена компанією Facebook [13]. Переваги використання React у даному проекті:

1. **Компонентний підхід**: React дозволяє розбити інтерфейс на незалежні компоненти, що спрощує розробку, тестування та повторне використання коду.
2. **Віртуальний DOM**: Завдяки використанню віртуального DOM, React забезпечує високу продуктивність додатку шляхом мінімізації операцій з реальним DOM.
3. **Односторонній потік даних**: Це спрощує розуміння та відстеження змін у стані додатку, що важливо для складних інтерактивних систем.
4. **Велика спільнота та підтримка**: Широка спільнота розробників та велика кількість додаткових бібліотек і інструментів полегшують вирішення різних завдань.

react-chartjs-2 — це бібліотека, яка забезпечує інтеграцію популярної бібліотеки для побудови графіків **Chart.js** з React. Переваги її використання:

1. **Проста інтеграція з React**: Компоненти react-chartjs-2 легко вбудовуються у React-додатки, що спрощує процес розробки.
2. **Потужні можливості візуалізації**: Chart.js підтримує різні типи графіків (лінійні, стовпчикові, кругові тощо), що дозволяє відображати дані у зручному для користувача форматі.
3. **Інтерактивність**: Графіки підтримують інтерактивні елементи, такі як підказки, легенди, масштабування, що покращує взаємодію користувача з даними.
4. **Налаштовуваність**: Велика кількість параметрів дозволяє налаштувати вигляд та поведінку графіків відповідно до потреб додатку.

У контексті навчальної платформи, використання React та react-chartjs-2 дозволило створити інтуїтивно зрозумілий інтерфейс, де користувачі можуть взаємодіяти з системою в реальному часі, змінюючи параметри таймера та спостерігаючи за результатами на графіках [14].

3.1.3. Вибір інтегрованого середовища розробки WebStorm

Для ефективної розробки веб-додатку було обрано інтегроване середовище розробки **WebStorm**, розроблене компанією JetBrains [15]. Вибір WebStorm обумовлений наступними факторами:

1. **Потужні засоби розробки**: WebStorm пропонує розширену підтримку JavaScript, React та інших веб-технологій, включаючи синтаксичний аналіз коду, автодоповнення, рефакторинг та навігацію по коду.
2. **Вбудована підтримка сучасних фреймворків**: IDE має інтеграцію з React, що полегшує роботу з компонентами, JSX-синтаксисом та іншими особливостями фреймворку.
3. **Налагодження та тестування**: WebStorm надає зручні інструменти для налагодження коду, включаючи підтримку breakpoints, огляд значень змінних та стеку викликів.
4. **Інтеграція з системами контролю версій**: Підтримка Git та інших систем контролю версій дозволяє легко керувати змінами у коді та співпрацювати з іншими розробниками.
5. **Гнучкість налаштувань**: Широкий спектр налаштувань дозволяє адаптувати середовище під індивідуальні потреби розробника.
6. **Плагіни та розширення**: Велика кількість доступних плагінів розширює функціональність IDE, додаючи підтримку нових технологій та інструментів.

Використання WebStorm сприяло підвищенню продуктивності розробки, забезпечуючи зручний та ефективний процес написання та налагодження коду. Це особливо важливо при роботі з складними додатками, де необхідно підтримувати високу якість коду та швидко реагувати на можливі помилки.

Таким чином, вибір JavaScript як мови програмування, використання бібліотек React та react-chartjs-2, а також інтегрованого середовища розробки WebStorm,

дозволили створити ефективну та зручну навчальну платформу, яка відповідає вимогам сучасних веб-технологій та забезпечує наочність та інтерактивність процесу навчання.

3.2. Моделювання роботи 8-бітного таймера-лічильника

Моделювання роботи 8-бітного таймера-лічильника є важливим кроком у розробці системи керування кроковими двигунами. Розуміння його принципів роботи та налаштування дозволяє точно генерувати необхідні часові інтервали для керування двигуном, забезпечуючи стабільну та ефективну роботу системи.

3.2.1. Теоретичні основи роботи таймера-лічильника ATmega2560

Таймери-лічильники у мікроконтролерах AVR, зокрема в ATmega2560, є апаратними модулями, які забезпечують генерацію точних часових інтервалів, вимірювання часу та формування сигналів керування. 8-бітний таймер-лічильник Timer2 є одним із таких модулів [16].

Основні характеристики 8-бітного таймера-лічильника Timer2:

- Розрядність: 8 біт (лічильник може приймати значення від 0 до 255).
- Режими роботи:
    - Normal Mode (звичайний режим).
    - Clear Timer on Compare Match (CTC) — обнулення таймера при співпадінні з регістром порівняння.
    - Fast PWM (швидка широтно-імпульсна модуляція).
    - Phase Correct PWM (фазово-коректна широтно-імпульсна модуляція).
- Джерела тактування:
    - Внутрішній генератор з переддільниками (1, 8, 32, 64, 128, 256, 1024).

- Зовнішнє тактування.

Режим CTC (Clear Timer on Compare Match):

- У режимі CTC таймер рахує імпульси від 0 до значення в регістрі OCR2A.
- При досягненні цього значення генерується подія Compare Match, таймер обнуляється, і починається новий цикл.
- Цей режим використовується для генерації періодичних подій з точно визначеним інтервалом.

Формування сигналів керування:

- Використовуючи переривання від таймера або апаратні можливості виводів OC2A та OC2B, можна генерувати сигнали на виходах мікроконтролера.
- У випадку керування кроковим двигуном сигнал STEP формується шляхом зміни стану виходу при кожному співпадінні таймера.

Розрахунок параметрів таймера:

- Частота імпульсів на виході залежить від тактової частоти мікроконтролера, переддільника та значення в регістрі OCR2A.
- Формула для розрахунку:

$$f_{STEP} = \frac{f_{clk}}{2 \times N \times (OCR2A + 1)} \quad (3.1)$$

Де:

- f_{out} — вихідна частота сигналу.
- f_{clk} — тактова частота мікроконтролера (зазвичай 16 МГц).
- N — коефіцієнт переддільника.

- OCR2A — значення регістру порівняння.

Приклад використання формули 3.1 для визначення частоти імпульсів:

- При f_{clk}=16 МГц, N=64, і OCR2A=124:

$$f_{STEP} = \frac{16000}{2 \times 64 \times (124 + 1)} \approx 1000\text{Гц}$$

Це означає, що на виході отримуємо сигнал з частотою 1 кГц.

3.2.2. Реалізація емуляції таймера у веб-додатку

Програма представляє собою інтерактивний веб-додаток, який моделює роботу таймера-лічильника на основі введених користувачем налаштувань. Інтуїтивно зрозумілий інтерфейс (див. рис. 3.1) дозволяє користувачу взаємодіяти з моделлю в реальному часі, спостерігати за змінами параметрів та аналізувати результати.

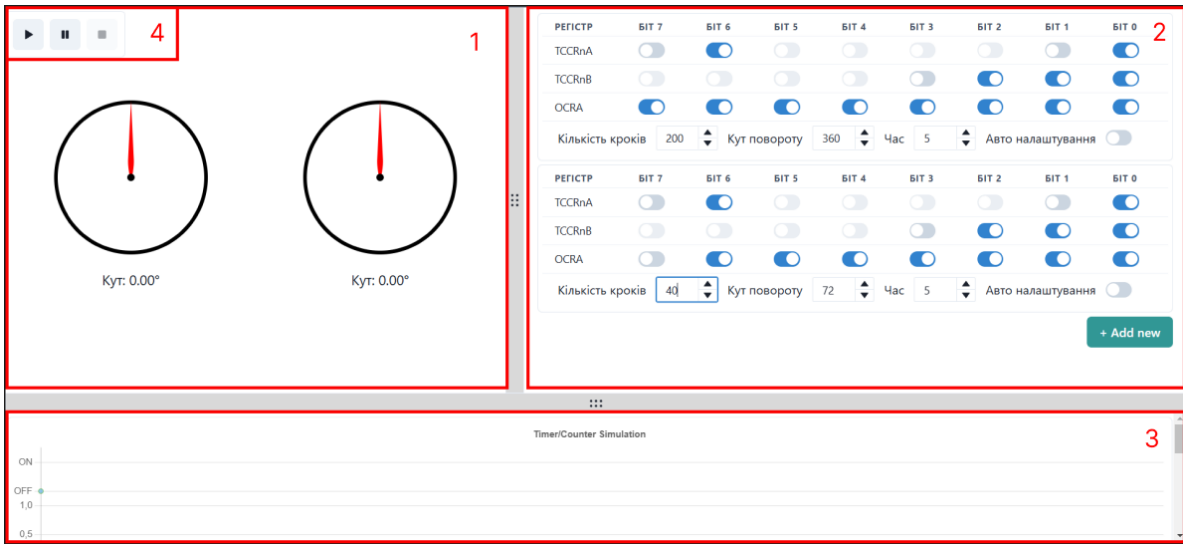


Рисунок 3.1 Інтерфейс користувача

Функціональні можливості програми

1. Налаштування регістрів таймера

   Користувач може вручну встановлювати значення регістрів:

   - TCCRnA та TCCRnB: визначають режими роботи таймера, коефіцієнти переддільника та поведінку виходів.
   - OCRnA: задає значення порівняння для події Compare Match.

2. Моделювання роботи таймера

   Програма виконує математичне моделювання роботи таймера на основі введених налаштувань, враховуючи режими роботи та значення регістрів. Це дозволяє користувачу бачити, як змінюється поведінка таймера при зміні параметрів.

3. Зворотні розрахунки

   Користувач може задати бажаний кут повороту θ. та час виконання руху ttt, а програма автоматично обчислює необхідні налаштування таймера для реалізації цього руху. Це включає розрахунок:

   - Кількості кроків $N_{кроків}$.
   - Необхідної частоти сигналу STEP f_{STEP}.
   - Значень регістрів OCRnA, TCCRnA, TCCRnB.

4. Візуалізація результатів
   - Графічне відображення роботи таймера, включаючи графіки зміни лічильника TCNTn та стану виходу OCnA.
   - Інформаційні панелі, що показують поточні значення параметрів та сигналів.
   - Панель з графічним відображенням руху крокових двигунів

Теоретичні основи, використані в програмі

Програма базується на теоретичних знаннях про роботу таймерів-лічильників у мікроконтролерах AVR, зокрема:

- Режими роботи таймера (Normal, CTC, PWM тощо).
- Розрахунки частоти та періоду таймера, використовуючи формули:

$$\mathrm{f_{таймера}} = \frac{\mathrm{f}_{clk}}{\mathrm{N}} \tag{3.2}$$

$$\mathrm{T_{таймера}} = \frac{\mathrm{N}}{\mathrm{f}_{clk}} \tag{3.3}$$

- Розрахунок періоду та частоти подій Compare Match:

$$\mathrm{T}_{CM} = \frac{(\mathrm{OCRnA} + 1) \times \mathrm{N}}{\mathrm{f}_{clk}} \tag{3.4}$$

$$f_{CM} = \frac{\mathrm{f}_{clk}}{(\mathrm{OCRnA} + 1) \times \mathrm{N}} \tag{3.5}$$

- Зв'язок між частотою сигналу STEP та параметрами таймера:

$$f_{STEP} = \frac{f_{CM}}{2} \tag{3.6}$$

- Зворотні розрахунки для визначення значень регістрів на основі бажаних параметрів руху двигуна.

Приклад використання програми

1. Задання параметрів руху

   Користувач вводить:

   - Кут повороту θ=90°
   - Час виконання руху t=1

- Режим мікрокрокування M=8 (1/8 кроку)

2. Розрахунки, виконані програмою
   - Кількість кроків:

$$N_{\text{кроків}} = \frac{90^\circ}{1.8^\circ \times 8} = \frac{90^\circ}{14.4} = 6{,}25$$

Округлюємо до найближчого цілого числа: $N_{\text{кроків}}$=6.

   - Необхідна частота сигналу STEP:

$$f_{STEP} = \frac{6}{1} = 6 \text{ Гц}$$

   - Розрахунок значення OCRnA:

Обираючи переддільник N=64:

$$\text{OCRnA} = \frac{16000000}{(2 \times 64 \times 6)} - 1 = 128$$

   - Встановлення регістрів:
     - TCCRnA: налаштовується для режиму CTC.
     - TCCRnB: встановлюється переддільник N=64.
     - OCRnA=128.

3. Моделювання та візуалізація
   - Моделювання роботи таймера: програма симулює, як лічильник TCNTn рахує від 0 до OCRnA, після чого обнуляється.
   - Генерація сигналу STEP: зміна стану виходу OCnAO при кожному співпадінні значення лічильника з OCRnA.
   - Графічне відображення: користувач бачить графіки, які демонструють роботу таймера та сигналів у часі.
   - Підтвердження результатів: переконується, що двигун здійснить поворот на 90° за 1 секунду.

Переваги розробленої програми

- Інтерактивність: можливість експериментувати з різними параметрами та одразу бачити результати.
- Навчальний інструмент: допомагає глибше зрозуміти принципи роботи таймерів та керування кроковими двигунами.
- Практичність: автоматизація розрахунків спрощує процес налаштування системи керування для реальних застосувань.

3.3 Порівняння з існуючими аналогами

Для оцінки ефективності та унікальності розробленого веб-додатку з емуляцією таймера-лічильника мікроконтролера **ATmega2560** було проведено аналіз існуючих програмних рішень та веб-сервісів, які надають подібні можливості. Особливу увагу приділено конкретним програмам та сайтам, що використовуються в освітніх та інженерних колах.

3.3.1. Proteus Design Suite (Labcenter Electronics)

Proteus Design Suite — це професійний інструмент для розробки та симуляції електронних схем, який включає в себе модуль **Proteus VSM** (Virtual System Modelling) [17]. Цей модуль дозволяє моделювати роботу мікроконтролерів AVR, включаючи **ATmega2560**, та взаємодію з іншими компонентами схеми.

Можливості:

- Симуляція коду на рівні мікроконтролера.
- Візуалізація сигналів за допомогою осцилографа та логічного аналізатора.
- Можливість створення складних схем з різноманітними компонентами.

- Підтримка написання та відлагодження коду мовою **C/C++** або **Assembler**.

Обмеження:

- **Вартість:** Програмне забезпечення є комерційним і потребує придбання ліцензії.
- **Складність:** Вимагає значних знань у сфері електроніки та програмування мікроконтролерів.
- **Потреба у встановленні:** Необхідно встановлювати програму на комп'ютер, що може бути несумісним з деякими операційними системами.

3.3.2. Tinkercad Circuits (Autodesk)

Tinkercad Circuits — це безкоштовний онлайн-сервіс, який дозволяє моделювати прості електронні схеми та взаємодію з мікроконтролерами **Arduino** на базі **ATmega328P** [18].

Можливості:

- Простий інтерфейс для створення та моделювання схем.
- Підтримка написання коду на мові **Arduino IDE**.
- Візуалізація роботи схеми в режимі реального часу.
- Доступність через веб-браузер без необхідності встановлення.

Обмеження:

- **Обмежена підтримка мікроконтролерів:** Не підтримує моделювання **ATmega2560** та його специфічних модулів.
- **Обмежений функціонал таймерів:** Не надає детальної симуляції таймерів-лічильників та їх налаштувань.

3.3.3. SimulIDE

SimulIDE — це простий в реальному часі симулятор електронних схем та мікроконтролерів, який підтримує сімейства AVR, включаючи деякі моделі **ATmega** [19].

Можливості:

- Візуальна симуляція схем та мікроконтролерів.
- Можливість відлагодження коду в режимі реального часу.
- Інтуїтивний інтерфейс з основними інструментами для моделювання.

Обмеження:

- **Мінімалістичний функціонал:** Менш потужний у порівнянні з професійними інструментами.
- **Потреба у встановленні:** Необхідно завантажити та встановити програму на комп'ютер.

3.3.4. AVR Simulator IDE

AVR Simulator IDE — це середовище для симуляції та відлагодження AVR мікроконтролерів з підтримкою написання коду на **Assembler** та **C** [20].

Можливості:

- Симуляція роботи мікроконтролера на рівні регістрів.
- Підтримка основних периферійних модулів, включаючи таймери.
- Відлагодження коду з можливістю встановлення точок зупину.

Обмеження:

- **Складність інтерфейсу:** Менш зручний для новачків.

- **Відсутність графічної візуалізації:** Немає можливості візуалізувати сигнали чи рух двигуна.
- **Необхідність знання програмування:** Потрібні глибокі знання мікроконтролерів та мов програмування.

3.3.5 Порівняння з розробленим веб-додатком

Переваги розробленого веб-додатку:

1. **Доступність та зручність:**
    - **Веб-інтерфейс:** Не потребує встановлення програмного забезпечення, доступний з будь-якого пристрою з браузером.
    - **Інтуїтивний дизайн:** Призначений для користувачів з різним рівнем підготовки.
2. **Спеціалізація на таймерах та крокових двигунах:**
    - **Глибока опрацювання таймерів-лічильників:** Можливість детального налаштування та моделювання роботи таймера **ATmega2560**.
    - **Зворотні розрахунки:** Автоматичний розрахунок параметрів таймера на основі заданих параметрів руху двигуна.
3. **Інтерактивна візуалізація:**
    - **Графіки сигналів:** Відображення зміни лічильника та стану виходів у часі.
    - **Візуалізація руху двигуна:** Графічне представлення повороту крокового двигуна.
4. **Освітня спрямованість:**

- **Відсутність потреби у програмуванні:** Користувач може зосередитися на принципах роботи таймера без написання коду.

Недоліки в порівнянні з аналогами:

- **Обмежена універсальність:** Додаток зосереджений на специфічному випадку роботи таймера та керуванні кроковим двигуном.
- **Відсутність підтримки інших модулів:** Не підтримує моделювання інших периферійних модулів мікроконтролера.

Висновки до розділу 3

У цьому розділі було представлено розробку навчальної платформи для візуалізації процесу мікроконтролерного керування кроковими двигунами. Вибір мови програмування JavaScript, підтриманий бібліотеками React та react-chartjs-2, а також використання інтегрованого середовища розробки WebStorm, забезпечили створення ефективної та інтуїтивно зрозумілої платформи, яка відповідає сучасним вимогам веб-технологій.

Детальне дослідження теоретичних основ роботи 8-бітного таймера-лічильника ATmega2560 дозволило розробити математичні моделі та реалізувати їх у веб-додатку. Це забезпечило можливість користувачам взаємодіяти з моделлю в реальному часі, змінювати параметри та аналізувати результати, що сприяє глибшому розумінню принципів роботи таймерів та керування кроковими двигунами.

Порівняння з існуючими аналогами, такими як Proteus Design Suite, Tinkercad Circuits, SimulIDE та AVR Simulator IDE, виявило переваги розробленого веб-додатку в контексті доступності, спеціалізації та освітньої спрямованості. Хоча додаток має певні обмеження щодо універсальності, його унікальні можливості роблять його цінним інструментом для навчання та практичного застосування знань.

Загалом, розроблена платформа успішно поєднує теоретичні аспекти з практичною реалізацією, забезпечуючи інтерактивне та наочне середовище

для вивчення мікроконтролерного керування кроковими двигунами. Це сприяє підвищенню ефективності навчального процесу та розвитку навичок у сфері електроніки та програмування вбудованих систем.

ВИСНОВКИ

У рамках роботи досягнуто основної мети – створено навчальну платформу для демонстрації принципів керування кроковими двигунами за допомогою мікроконтролера.

Головні досягнення:

На основі мікроконтролера ATmega2560 реалізовано систему керування кроковими двигунами. Використання 8-бітного таймера-лічильника дозволило забезпечити необхідну точність і стабільність сигналів для управління.

Розроблено інтерактивний інтерфейс, що дозволяє налаштовувати параметри роботи таймера, моделювати поведінку системи та візуалізувати результати. Використання технологій React та react-chartjs-2 забезпечує зрозумілу та зручну взаємодію для користувача.

Платформа допомагає вивчати основи мікроконтролерного управління, дозволяючи змінювати налаштування, спостерігати за моделями та аналізувати дані в реальному часі.

Створена система поєднує в собі демонстрацію технологій керування кроковими двигунами з навчальними можливостями, що робить її корисним інструментом для закладів освіти, самостійного навчання та подальших дослідницьких проєктів у галузі автоматизації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a Stepper Motor : Types & Its Working [Електронний ресурс],URL: https://www.elprocus.com/stepper-motor-types-advantages-applications/ (Дата звернення: 20.10.2024).
2. AVR446: Linear Speed Control of Stepper Motor [Електронний ресурс],URL: https://ww1.microchip.com/downloads/en/Appnotes/doc8017.pdf (Дата звернення: 20.10.2024).
3. Application of Micro-controllers for Stepper Motors Position and Speed Control [Електронний ресурс],URL: https://www.ijser.org/researchpaper/Application-of-Micro-controllers-for-Stepper-Motors-Position-and-Speed-Control-A-review.pdf (Дата звернення: 20.10.2024).
4. ATmega640/1280/1281/2560/2561 - Complete Datasheet. [Електронний ресурс],URL: https://www.microchip.com/en-us/product/atmega2560 (Дата звернення: 20.10.2024).
5. Stm32 datasheet. [Електронний ресурс],URL: https://www.alldatasheet.com/datasheet-pdf/pdf/560960/ETC2/STM32.html (Дата звернення: 20.10.2024).
6. PIC16F84A Data Sheet URL: https://www.alldatasheet.com/datasheet-pdf/pdf/82341/MICROCHIP/PIC16F876A.html Дата звернення: 20.11.2024
7. Крокові двигуни [Електронний ресурс],URL: https://ukrayinska.libretexts.org/Робоча_сила/Технологія_електроніки/Книга%3A_Електричні_ланцюги_II_-_змінний_струм_(Kuphaldt)/13%3A_Двигуни_змінного_струму/13.05%3A_Крокові_двигуни (Дата звернення: 20.10.2024).
8. Stepper Motor Control Using AVR (Atmega) Microcontroller [Електронний ресурс],URL: https://www.elprocus.com/stepper-motor-control-using-avr-microcontroller/ (Дата звернення: 20.10.2024).

9. Design of the Stepper Motor Controller [Електронний ресурс],URL: https://link.springer.com/chapter/10.1007/978-3-642-21697-8_66 (Дата звернення: 20.10.2024).
10. Система керування кроковим двигуном на базі мікроконтролера та персонального комп'ютера [Електронний ресурс],URL: https://epa.kpi.ua/files/conference/fea/2008/04.pdf (Дата звернення: 20.10.2024).
11. Контролер керування кроковими двигунами [Електронний ресурс],URL: https://science.lpnu.ua/sites/default/files/journal-paper/2017/nov/6680/23-7.pdf (Дата звернення: 20.10.2024).
12. JavaScript [Електронний ресурс],URL: https://uk.wikipedia.org/wiki/JavaScript (Дата звернення: 20.10.2024).
13. React [Електронний ресурс],URL: https://legacy.reactjs.org/ (Дата звернення: 20.10.2024).
14. Chart.js [Електронний ресурс],URL: https://www.chartjs.org/docs/latest/ (Дата звернення: 20.10.2024).
15. WebStorm [Електронний ресурс],URL: https://www.jetbrains.com/webstorm/ (Дата звернення: 20.10.2024).
16. Intellectual capital is the foundation of innovative development '2024 [Електронний ресурс],URL: https://desymp.promonograph.org/index.php/sge/issue/view/sge28-01 (Дата звернення: 20.10.2024).
17. Proteus Design Suite [Електронний ресурс],URL: https://www.labcenter.com/ (Дата звернення: 20.10.2024).
18. Tinkercad Circuits [Електронний ресурс],URL: https://www.tinkercad.com/ (Дата звернення: 20.10.2024).
19. SimulIDE [Електронний ресурс],URL: https://simulide.com/p/ (Дата звернення: 20.10.2024).

20.AVR SIMULATOR IDE [Електронний ресурс],URL: https://www.oshonsoft.com/avr.php (Дата звернення: 20.10.2024).

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 систем та робототехніки
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

Кириченка Олександра Сергійовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи: **Модель мікроконтрольного керування кроковими двигунами для повороту коліс рухомої платформи**

керівник роботи Котвицький Альберт Тадеушевич, к.ф-м.н., доцент,
(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 літератури, присвяченої будові та керуванню кроковими двигунами.
2. Огляд мікроконтролерних платформ для керування кроковими двигунами.
3. Моделювання роботи крокового двигуна.
4. Розробка програмної моделі мікроконтролерного керування кроковими двигунами.
5. Використання моделі для керування рухомою платформою для повороту коліс за допомогою двох двигунів.
6. Тестування моделі.
7. Аналіз результатів тестування та виявлених помилок.
8. Складання звіту про виявлені помилки та їх можливі шляхи усунення.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Дослідження методів управління крокovими двигунами	05.09.2024 — 15.09.2024
2	Вивчення основ програмування мікроконтролерів	16.09.2024 — 30.09.2024
3	Вибір інструментів розробки для програмної моделі керування	01.10.2024 — 10.10.2024
4	Моделювання процесу роботи крокових двигунів	11.10.2024 — 31.10.2024
5	Створення програмної моделі для управління двигунами	01.11.2024 — 10.11.2024
6	Проведення тестування розробленої моделі	11.11.2024 — 18.11.2024
7	Підготовка до попереднього захисту дипломної роботи	19.09.2024 — 23.10.2024
8	Представлення кваліфікаційної роботи та моделі керівнику кваліфікаційної роботи та рецензенту	24.11.2024 - 30.11.2024

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

Студент О. С. Кириченко
ініціали, прізвище

підпис

Керівник роботи А. Т. Котвицький
ініціали, прізвище

підпис

Додаток Б

Затверджую

_____

«_____»_____2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

«Модель мікроконтрольного керування кроковими двигунами для повороту коліс рухомої платформи»

Назва розділу	Назва та зміст підрозділу
1. Вступ	1.1. Назва програмного виробу: МОДЕЛЬ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ КРОКОВИМИ ДВИГУНАМИ ДЛЯ ПОВОРОТУ КОЛІС РУХОМОЇ ПЛАТФОРМИ 1.2. Галузь застосування: робототехніка, промисловість
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 174 – 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: створення навчальної платформи для візуалізації процесу керування кроковим двигуном, з особливим акцентом на використанні 8-бітного таймера-лічильника для генерації імпульсів з правильною частотою 3.2. Призначення програмного виробу: Програмний виріб призначений для навчання основам мікроконтролерного керування. Він дозволяє моделювати роботу таймера-

	лічильника ATmega2560, формувати керуючи імпульси для керування кроковим двигуном. 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; 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 кроковими двигунами для повороту коліс рухомої платформи» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи). 2) Програму та методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до кваліфікаційної роботи)
6. Вимоги до техніко-економічних показників	Програмною документацією до виробу «Модель мікроконтрольного керування кроковими двигунами для повороту коліс рухомої платформи» вважати: 1) Справжнє Технічне завдання на розробку виробу (представити у вигляді Додатку Б до пояснювальної

	записки до кваліфікаційної роботи). 2) Програму та методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до кваліфікаційної роботи)	
7. Стадії та етапи розробки	Дата	Назва етапу
	Вересень 2024	Дослідження методів управління крокoвими двигунами
	Вересень 2024	Вивчення основ програмування мікроконтролерів
	Вересень 2024	Вибір інструментів розробки для програмної моделі керування
	Жовтень 2024	Моделювання процесу роботи крокових двигунів
	Жовтень 2024 – Листопад 2024	Створення програмної моделі для управління двигунами
	Листопад 2024	Проведення тестування розробленої моделі
	Листопад 2024	Підготовка до попереднього захисту дипломної роботи
	Листопад 2024	Представлення кваліфікаційної роботи та моделі керівнику кваліфікаційної роботи 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
Кириченко О. С. _____

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

Додаток В

ПРОГРАМНА РЕАЛІЗАЦІЯ
«МОДЕЛЬ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ КРОКОВИМИ ДВИГУНАМИ ДЛЯ ПОВОРОТУ КОЛІС РУХОМОЇ ПЛАТФОРМИ»

1 Об’єкт випробувань

1. Назва програмного виробу : «МОДЕЛЬ МІКРОКОНТРОЛЬНОГО КЕРУВАННЯ КРОКОВИМИ ДВИГУНАМИ ДЛЯ ПОВОРОТУ КОЛІС РУХОМОЇ ПЛАТФОРМИ»

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 для

зменшення 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 кроковими двигунами для повороту коліс рухомої платформи» 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.
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 “package.json” 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 дві 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 CTC (Clear Timer on Compare) у програму та макет.
3. Отримання результату обробки даних у вигляді графіку, графічному зоббрж.

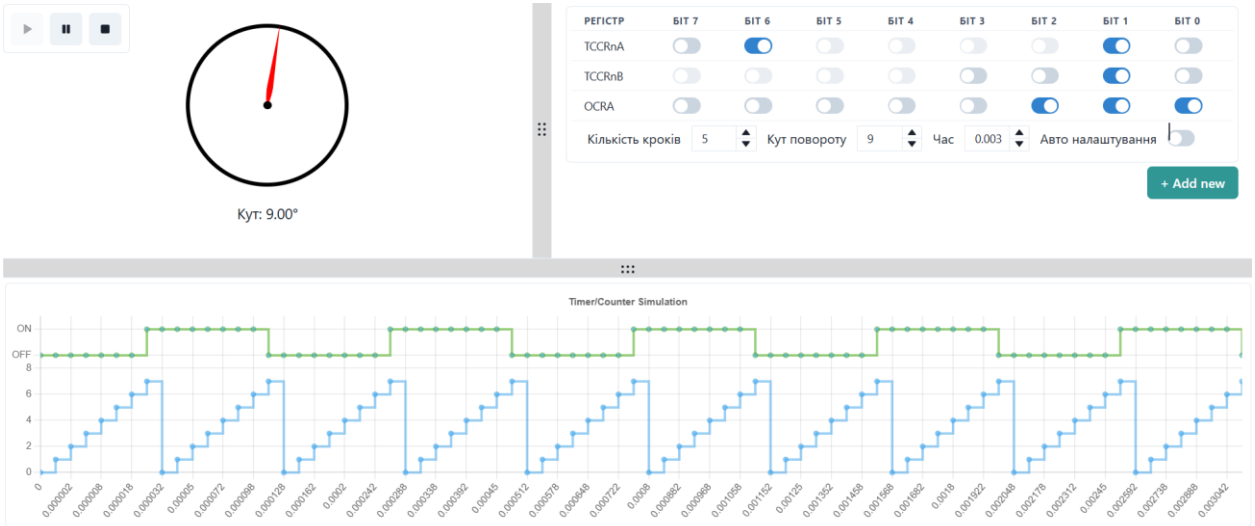


Рис. В.1 Тест 1

Тест 2

1. Перевірка виконання програми;
2. Введення серії нових даних;
3. Отримання результату при заданих куті повороту та часу

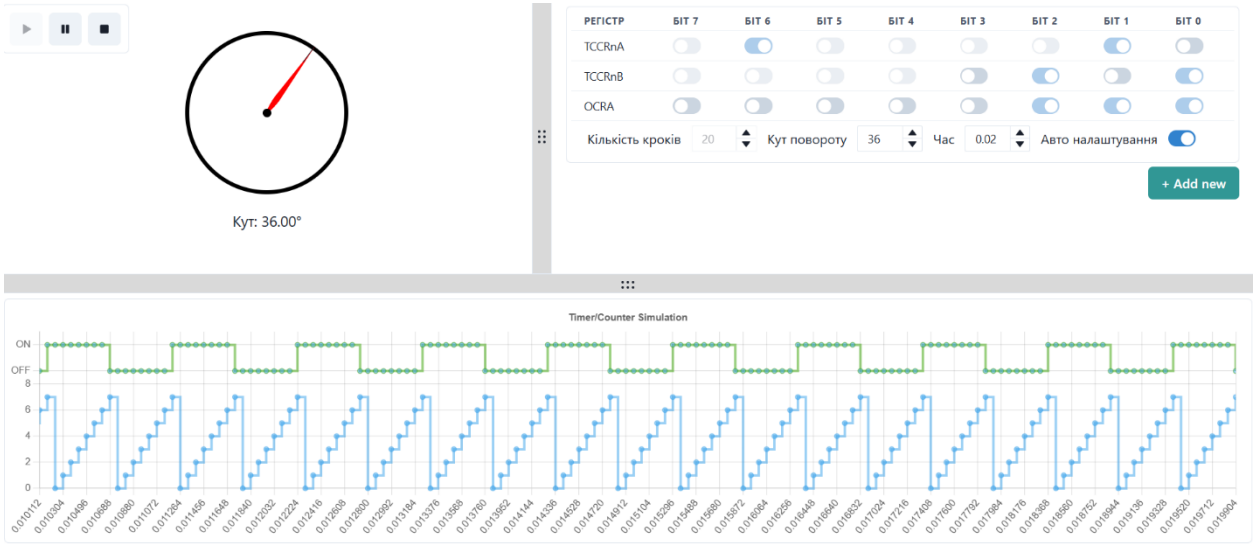


Рис. В.2 Тест 2

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

Виконавець: студент групи КУ-61, Кириченко О.С. _____

Лістинг частини виконавчого коду для візуалізації

```
import './MainPage.css';
import EngineSection from "../EngineSection/EngineSection";
import UserInputSection from
"../UserInputSection/UserInputSection";
import {GraphsSection} from "../GraphsSection/GraphsSection";
import {Panel, PanelGroup, PanelResizeHandle} from "react-
resizable-panels";
import {DragHandleIcon} from '@chakra-ui/icons'
import {Center, Container} from "@chakra-ui/react";
import {Player} from "../Player/Player";
import Draggable from "react-draggable";

function MainPage() {

    return (
        <>
            <Draggable cancel="strong" bounds="parent">
                <div style={{position: 'absolute', zIndex:
100}}>
                    <Player/>
                </div>
            </Draggable>
            <PanelGroup id="mainPage" autoSaveId="example_test"
direction="vertical">
                <Panel>
                    <PanelGroup autoSaveId="example"
direction="horizontal">
                        <Panel>
                            <EngineSection/>
                        </Panel>
                        <PanelResizeHandle>
                            <Container padding="5px" width={6}
height={"full"} minHeight={"full"} borderWidth="1px"
                                       background={"rgba(180,
180, 180, 0.5)"}>
                                <Center height={"full"}>
                                    <DragHandleIcon/>
                                </Center>
                            </Container>
                        </PanelResizeHandle>
                        <Panel>
                            <UserInputSection/>
                        </Panel>
                    </PanelGroup>
                </Panel>
                <PanelResizeHandle>
                    <Container padding="5px" height={6}
minWidth={"full"} borderWidth="1px"
                               background={"rgba(180, 180, 180,
0.5)"}>
```

```
                        <Center width={"full"}>
                            <DragHandleIcon style={{rotate:
"90deg"}}/>
                        </Center>
                    </Container>
                </PanelResizeHandle>
                <Panel>
                    <GraphsSection/>
                </Panel>
            </PanelGroup>
        </>
    );
}

export default MainPage;


import { useEffect, useState } from "react";
import { useSelector } from "react-redux";
import bigDecimal from "js-big-decimal";
import { useDispatch } from "react-redux";
import { stop as stopAction } from '../store/slice/playerSlice';
import {updateItem} from "../store/slice/outputSlice";

export const useChartGeneration = (id) => {
    const userInput = useSelector((state) => state.userInput);
    const output = useSelector((state) => state.output);
    const playerState = useSelector((state) => state.player);
    const dispatch = useDispatch();

    const currentUserInputIndex = userInput.findIndex(item =>
item.id === id)
    const TCCRnA =
userInput[currentUserInputIndex]?.body?.TCCRnA || [false, false,
false, false, false, false, false, false];
    const TCCRnB =
userInput[currentUserInputIndex]?.body?.TCCRnB || [false, false,
false, false, false, false, false, false];

    const currentOutputIndex = output.findIndex(item => item.id
=== id)

    const initialOC0A = output[currentOutputIndex].OC0A ;

    const [data, setData] = useState({
        labels: ["0"], // Початкове значення для осі X
        datasets: [
            {
                label: 'TCNTn',
                data: [initialTCNT0],
                borderColor: "rgba(53, 162, 235, 0.5)",
                backgroundColor: "rgba(53, 162, 235, 0.5)",
```

```
            stepped: true,
        },
        {
            label: 'OCnA',
            data: [initialOC0A ? 'ON' : 'OFF'],
            borderColor: "rgba(53, 162, 0, 0.5)",
            backgroundColor: "rgba(53, 162, 235, 0.5)",
            stepped: true,
            yAxisID: 'y2',
        },
    ]
});

const [options, setOptions] = useState({
    maintainAspectRatio: false,
    responsive: true,
    animation: {
        duration: 100,
        easing: 'linear',
    },
    plugins: {
        title: {
            display: true,
            text: 'Timer/Counter Simulation',
        },
        zoom: {
            zoom: {
                wheel: {
                    enabled: true
                },
                mode: "x",
                speed: 100
            },
            pan: {
                enabled: true,
                mode: "x",
                speed: 100
            }
        }
    },
    scales: {
        y: {
            type: 'linear',
            position: 'left',
            stack: 'demo',
            stackWeight: 2,
        },
        y2: {
            type: 'category',
            labels: ['ON', 'OFF'],
            offset: true,
            position: 'left',
```

```
                stack: 'demo',
                stackWeight: 1,
            }
        }
    });

    const calculateNextValues = (clockFrequency, TCCRnA, TCCRnB)
=> {
        const WGM02 = TCCRnB[4] ? 1 : 0;
        const WGM01 = TCCRnA[6] ? 1 : 0;
        const WGM00 = TCCRnA[7] ? 1 : 0;

        const COM0A1 = TCCRnA[0] ? 1 : 0;
        const COM0A0 = TCCRnA[1] ? 1 : 0;

        const CS00 = TCCRnB[7] ? 1 : 0;
        const CS01 = TCCRnB[6] ? 1 : 0;
        const CS02 = TCCRnB[5] ? 1 : 0;


        const calculatePrescaler = (CS02, CS01, CS00) => {
            let prescaler = 1;

            if (CS02 === 0 && CS01 === 0 && CS00 === 0) {
                return 0;
            } else if (CS02 === 0 && CS01 === 0 && CS00 === 1) {
                prescaler = 1;
            } else if (CS02 === 0 && CS01 === 1 && CS00 === 0) {
                prescaler = 8;
            } else if (CS02 === 0 && CS01 === 1 && CS00 === 1) {
                prescaler = 64;
            } else if (CS02 === 1 && CS01 === 0 && CS00 === 0) {
                prescaler = 256;
            } else if (CS02 === 1 && CS01 === 0 && CS00 === 1) {
                prescaler = 1024;
            }
            return prescaler;
        };

        const waveformMode = getWaveformGenerationMode(WGM02,
WGM01, WGM00);
        const prescaler = calculatePrescaler(CS02, CS01, CS00)
        const timeDelta = 1 / (clockFrequency / prescaler)

        return (currentTime, OC0A, TCNT0, OCRA) => {
            let topValue = waveformMode.top === "OCRA" ? OCRA :
waveformMode.top;

            let newTime = bigDecimal.add(currentTime,
timeDelta);
            let newTCNT0 = (TCNT0 + 1) % (topValue + 1);
```

```
            let newOC0A = OC0A;

            if (waveformMode.mode === "Normal") {
                if (newTCNT0 === OCRA) {
                    if (COM0A1 === 0 && COM0A0 === 1) {
                        newOC0A = OC0A === 0 ? 1 : 0;
                    } else if (COM0A1 === 1 && COM0A0 === 0) {
                        newOC0A = 0;
                    } else if (COM0A1 === 1 && COM0A0 === 1) {
                        newOC0A = 1;
                    }
                }
            } else if (waveformMode.mode === "Fast PWM") {
                if (WGM02 === 1 && COM0A1 === 0 && COM0A0 === 1)
{
                    if (newTCNT0 === OCRA) {
                        newOC0A = OC0A === 0 ? 1 : 0;
                    }
                }
            } else if (waveformMode.mode === "PWM, Phase
Correct") {
                if (WGM02 === 1 && COM0A1 === 0 && COM0A0 === 1)
{
                    if (newTCNT0 === OCRA) {
                        newOC0A = OC0A === 0 ? 1 : 0;
                    }
                }
            } else if (waveformMode.mode === "CTC") {
                if (newTCNT0 === OCRA) {
                    if (COM0A1 === 0 && COM0A0 === 1) {
                        newOC0A = OC0A === 0 ? 1 : 0;
                    } else if (COM0A1 === 1 && COM0A0 === 0) {
                        newOC0A = 0;
                    } else if (COM0A1 === 1 && COM0A0 === 1) {
                        newOC0A = 1;
                    }
                }
            }

            let TOVFlag = false;
            if (waveformMode.TOVFlagSetOn === "MAX" && newTCNT0
=== topValue) {
                TOVFlag = true;
            } else if (waveformMode.TOVFlagSetOn === "BOTTOM" &&
newTCNT0 === 0) {
                TOVFlag = true;
            }


            dispatch(updateItem({ id, OC0A:newOC0A }));
            return {
                newTime: newTime,
```

```
                newOC0A: newOC0A,
                newTCNT0: newTCNT0,
                TOVFlag: TOVFlag
            };
        }

    };

    function bitsToDecimal(bits) {
        if (!Array.isArray(bits) || bits.some(bit => bit !== 0
&& bit !== 1 && bit !== false && bit !== true)) {
            throw new Error("Масив повинен містити лише значення
0 або 1 (false або true)");
        }

        const normalizedBits = bits.map(bit => bit === true ? 1
: bit === false ? 0 : bit);

        const binaryString = normalizedBits.join('');
        return parseInt(binaryString, 2);
    }

    useEffect(() => {
        const clockFrequency = 16000000;
        const OCRA =
bitsToDecimal(userInput[currentUserInputIndex]?.body?.OCRA); //
Приклад значення для OCRA
        let currentTime = 0;
        let OC0A = initialOC0A;
        let TCNT0 = initialTCNT0;

        let interval;

        if (playerState.play) {
            let step = 0;
            const updateValues =
calculateNextValues(clockFrequency, TCCRnA, TCCRnB);
            interval = setInterval(() => {
                const { newTime, newOC0A, newTCNT0 } =
updateValues(currentTime, OC0A, TCNT0, OCRA);
                if(parseFloat(newTime) === 0 || step ===
userInput[currentUserInputIndex]?.body?.stepCount){
                    clearInterval(interval); // При pause
зупиняємо підрахунок
                    return;
                }

                if (OC0A === 1 && newOC0A === 0 &&
!playerState.stop) {
                    step++; // Робимо крок
                }
                currentTime = newTime;
```

```
                OC0A = newOC0A;
                TCNT0 = newTCNT0;

                setData((oldValue) => {
                    const labels = [...oldValue.labels];
                    const newTimerData =
[...oldValue.datasets[0].data];
                    const newOC0AData =
[...oldValue.datasets[1].data];

                    const newLabel = !isNaN(newTime) ? newTime :
0;

                    labels.push(newLabel);
                    newTimerData.push(newTCNT0);
                    newOC0AData.push(newOC0A ? 'ON' : 'OFF');

                    if (labels.length > 100) {
                        setOptions(oldOptions => ({
                            ...oldOptions,
                            scales: {
                                ...oldOptions.scales,
                                x: {
                                    min: labels.length - 100,
                                    max: labels.length,
                                }
                            }
                        }));
                    }

                    return {
                        labels: labels,
                        datasets: [
                            {
                                ...oldValue.datasets[0],
                                data: newTimerData,
                            },
                            {
                                ...oldValue.datasets[1],
                                data: newOC0AData,
                            },
                        ],
                    };
                });
            }, 100);
        }

        if (playerState.pause ) {
            clearInterval(interval);
        }

        if (playerState.stop) {
```

```
            clearInterval(interval);
            setData({
                labels: ["0"],
                datasets: [
                    {
                        label: 'TCNTn',
                        data: [initialTCNT0],
                        borderColor: "rgba(53, 162, 235, 0.5)",
                        backgroundColor: "rgba(53, 162, 235,
0.5)",
                        stepped: true,
                    },
                    {
                        label: 'OCnA',
                        data: [initialOC0A ? 'ON' : 'OFF'],
                        borderColor: "rgba(53, 162, 0, 0.5)",
                        backgroundColor: "rgba(53, 162, 235,
0.5)",
                        stepped: true,
                        yAxisID: 'y2',
                    },
                ]
            });
            currentTime = 0;
            OC0A = initialOC0A;
            TCNT0 = initialTCNT0;
            dispatch(stopAction());
            dispatch(updateItem({ id, OC0A:initialOC0A }));
        }

        return () => {
            clearInterval(interval);
        };
    }, [playerState]);

    return [data, options];
};

function getWaveformGenerationMode(WGM02, WGM01, WGM00) {
    if (WGM02 === 0 && WGM01 === 0 && WGM00 === 0) {
        return {
            mode: "Normal",
            top: 255, // 0xFF
            updateOfOCRxAt: "Immediate",
            TOVFlagSetOn: "MAX",
        };
    } else if (WGM02 === 0 && WGM01 === 0 && WGM00 === 1) {
        return {
            mode: "PWM, Phase Correct",
            top: 255, // 0xFF
            updateOfOCRxAt: "TOP",
            TOVFlagSetOn: "BOTTOM",
```

```
        };
    } else if (WGM02 === 0 && WGM01 === 1 && WGM00 === 0) {
        return {
            mode: "CTC",
            top: "OCRA",
            updateOfOCRxAt: "Immediate",
            TOVFlagSetOn: "MAX",
        };
    } else if (WGM02 === 0 && WGM01 === 1 && WGM00 === 1) {
        return {
            mode: "Fast PWM",
            top: 255, // 0xFF
            updateOfOCRxAt: "TOP",
            TOVFlagSetOn: "MAX",
        };
    } else if (WGM02 === 1 && WGM01 === 0 && WGM00 === 1) {
        return {
            mode: "PWM, Phase Correct",
            top: "OCRA",
            updateOfOCRxAt: "TOP",
            TOVFlagSetOn: "BOTTOM",
        };
    } else if (WGM02 === 1 && WGM01 === 1 && WGM00 === 1) {
        return {
            mode: "Fast PWM",
            top: "OCRA",
            updateOfOCRxAt: "BOTTOM",
            TOVFlagSetOn: "TOP",
        };
    }
}
```