Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету(відділення))

Кафедра інформаційних систем і комп'ютерного моделювання
(повна назва кафедри (предметної циклової комісії))

Пояснювальна записка

до дипломної роботи

перший (бакалаврський)
(рівень вищої освіти)

на тему: «Розроблення алгоритму і програмного забезпечення для роботи автономного мобільного саперного пристрою засобами C++»

Виконав студент 2 курсу, групи ІСТС-21
спеціальності: 126
„Інформаційні системи та технології"
(шифр і назва напрямку підготовки спеціальності)

Ситник В. В.
(прізвище, ініціали)

Керівник: доц. Прусак Ю.В.
(прізвище, ініціали)

Рецензент: Крошний І.М.
(прізвище, ініціали)

Львів-2023

Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну

Кафедра Інформаційних систем і комп'ютерного моделювання

Рівень вищої освіти перший (бакаравський)

Спеціальність 126 *„Інформаційні системи та технології"*

ЗАТВЕРДЖУЮ:

В.о завідувача кафедри ІСКМ

_____ Сторожук О.Л.

„21" 11 2022 року

ЗАВДАННЯ

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Ситник Владислав Володимирович

(прізвище, ім'я, по батькові)

1.Тема магістерської роботи: Розроблення алгоритму і програмного забезпечення для роботи автономного мобільного саперного пристрою засобами С++

керівник роботи доц. Прусак Ю.В.,

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "21" 11 2022 року, №С-521

2.Термін подання студентом проекту(роботи) 10 червня 2023р

3. Вихідні дані до проекту (роботи) Розробки програмного забезпечення для пристрою котрий керує роботою шліфувального верстата. У роботі розроблено програмне забезпечення пристрою засобами середовища Arduino IDE під управління операційної системи Linux мовою програмування С/С++.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Стан проблемної області

Інформаційне забезпечення

Програмне забезпечення

Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Підготовка матеріалу до доповіді .

6. Консультанти розділів проекту (роботи)

7. Дата видачі завдання 23 листопада 2022р.

КАЛЕНДАРНИЙ ПЛАН

№, з/п	Етапи бакалаврської роботи	Термін виконання етапів роботи	Примітка
1.	Огляд літератури згідно досліджуваної теми. Збір необхідних матеріалів.	23.11-20.12	Виконано
2.	Постановка задачі і її формалізація	20.12-13.01	Виконано
3.	Виконання вхідного етапу технології	13.01-14.04	Виконано
4.	Реалізація головних класів проекту	14.04-20.05	Виконано
5.	Виконання етапу відлагодження проєкту	20.05.-25.05.	Виконано
6.	Виконання етапу впровадження та випуску бета-версії.	25.05.-02.06.	Виконано
7.	Оформлення записки до дипломного проекту.	02.06.-10.06.	Виконано

Студент _____ Ситник Владислав Володимирович
(підпис) (прізвище та ініціали)

Керівник роботи _____ доц.Прусак Ю.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 38 сторінок пояснювальної записки, 21 рисунків, 15 джерел, 1 додатку.

Дипломна робота виконана з метою розробки програмного забезпечення для пристрою котрий керує роботою шліфувального верстата. У роботі розроблено програмне забезпечення пристрою засобами середовища Arduino IDE під управління операційної системи Linux. Для створення програмного забезпечення використовуватиметься мова програмування С. Робота з пристроєм інтуїтивно зрозуміла і не потребує додаткової кваліфікації персоналу.

Ключові слова: Програмний інтерфейс, прошивка, програмне забезпечення, модуль мікроконтролера, сенсор.

ABSTRACT

The thesis contains 38 pages of an explanatory note, 21 figures, 15 sources, 1 appendix.

The thesis was completed with the aim of developing software for a device that controls the operation of a grinding machine. In the work, the software of the device was developed using the Arduino IDE environment under the control of the Linux operating system. The C programming language will be used to create the software. Working with the device is intuitive and does not require additional personnel qualifications

Keywords: Software interface, firmware, software, microcontroller module, sensor.

ТЕХНІЧНЕ ЗАВДАННЯ

Для пристрою, створеному на базі мікроконтролера фірми Atmel, створити програмне забезпечення (firmware, прошивку). Програмне забезпечення повинно формувати управляючі команди (сигнали) для блоків управління кроковими двигунами, надавати вивід діагностичної інформації, забезпечувати зчитування значень із сенсорів (датчиків), а також кнопок управління. Пристрій буде встановлюватись на верстат для обробки матеріалів для управління роботою верстата. Програмне забезпечення має бути кросплатформеним, тобто легко переноситись на інші моделі шиирокоросповсюджених мікроконтролерів. А також мати низький поріг входу.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ЧПК – числове програмне керування.

ЕОМ – електронно-обчислювальна машина.

МК – мікроконтролер.

IDE – Integrated Development Environment, інтегроване середовище розробки.

UART – Universal Asynchronous Receiver-Transmitter, універсальний асинхронний приймач-передавач.

STEP/DIR – протокол керування кроковими двигунами (крок/напрям).

BTNS – модуль кнопок керування (Start/Stop).

MCU – Microcontroller Unit, мікроконтролерний модуль. Axis Sensors – датчики кінцевих положень осей.

Main Motor Sensor – сенсор обертів головного двигуна.

Relay Switches – релейні модулі керування навантаженням.

Stepper Driver – драйвер крокового двигуна.

STL – Standard Template Library, стандартна бібліотека шаблонів C++.

ВСТУП

Верстат з числовим програмним керуванням (ЧПУ) представляє собою комп'ютерно керовану машину, яка забезпечує високу точність та ефективність обробки. Завдяки використанню ЧПУ, обробний інструмент та заготовка матеріалу контролюються спеціальною комп'ютерною програмою, що відкриває безліч можливостей для промисловості.

У процесі роботи з верстатом ЧПУ використовуються дві ключові технології - CAD (автоматизоване проектування) та CAM (автоматизоване виробництво). CAD дозволяє створювати точні та складні моделі продукту на комп'ютері, що служить основою для подальшої обробки. CAM використовує дані з CAD, перетворюючи їх у програми керування верстатом, які визначають шлях руху інструменту та необхідні параметри обробки.

Верстати з ЧПУ знаходять своє застосування в різних галузях промисловості, включаючи машинобудування, авіацію, медичну техніку та інші. Вони є незамінним інструментом для виробництва складних деталей, прототипування, нарізання різьби та багатьох інших операцій.

Загалом, верстати з ЧПУ забезпечують високу якість, точність та ефективність обробки. Ці машини є важливим компонентом сучасної промисловості, що допомагають підприємствам досягати високих стандартів і задовольняти вимоги ринку.

Об’єктом дослідження використання Arduino для створення верстату.

Метою роботи розробити шліфувальний верстат за допомогою C/C++ мов програмування.

Предметом дослідження використання нових технологій для покращення точності ЧПУ.

Практичне значення роботи полягає у простоті та дишевизні реалізації систем такого типу.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області

3D-друкарки, лазерні різаки, різаки водним струменем високого тиску, токарні та фрезерні верстати для роботи із деревом і металом, дельта-роботи, механічні маніпулятори - це лише кілька прикладів сучасних машин з числовим програмним управлінням (ЧПК). Вони пропонують надзвичайну точність і можливості для виконання складних операцій.

ЧПК-машини забезпечуються спеціальними контролерами, що дозволяють оператору точно визначати їх рухи за допомогою числових команд. Вони використовують приводи на основі крокових двигунів або сервоприводів для переміщення в заданих координатах. Цей процес передбачає перетворення бажань оператора на інструкції в мові машинного коду, які ЧПК буде виконувати.

Історія числового програмного управління сягає далекого 1801 року, коли Ж.М. Жакаром був винайдений перший ткацький верстат, що керувався двійковим кодом. Цей приклад числового керування показує, що ідея програмного управління існувала задовго до появи сучасних комп'ютерів, коли програми для верстатів записувались на перфоровані картки.

Перший промисловий верстат з числовим керуванням був винайдений американським інженером Джоном Парсонсом. Його верстати застосовувались у авіаційній промисловості для виготовлення складних деталей. З тих пір числове програмне керування перетворилося на комп'ютеризовану систему, яка керує верстатами та іншим технологічним устаткуванням.

Структура ЧПК схожа на структуру електронно-обчислювальних машин. Вона включає запам'ятовуючі пристрої, пульти керування, пристрої виводу і оперативну пам'ять. Контролером може бути як спеціальний промисловий комп'ютер, так і персональний комп'ютер.

Основною характеристикою ЧПК-контролера є його здатність керувати кількома осями. Для цього необхідна потужна обчислювальна система. З

розвитком технологій відмінність у потужності між промисловим і персональним обладнанням значно зменшилась, тому персональний комп'ютер може виступати в ролі контролера ЧПК. Виконавчими механізмами у ЧПК-машин часто виступають сервоприводи, крокові двигуни і лінійні двигуни.

1.2. Актуальність застосування обладнання з числовим програмним керуванням

Протягом останнього десятиліття в українському машинобудуванні спостерігається тенденція оснащення технологічного устаткування обладнанням з числовим програмним керуванням (ЧПК). Це стає основним напрямком розвитку верстатів з використанням ЧПК, де впроваджуються нові технології і функціональні можливості.

Одним з напрямків є створення пристроїв ЧПК на основі мікро-ЕОМ та мікропроцесорів, а також введення діагностичних пристроїв для системи верстату. Широке впровадження автоматизованих адаптивних пристроїв сприяє оптимізації управління та обробки деталей. Також розвиваються пристрої ЧПК, які можуть керувати як окремими верстатами, так і групою верстатів.

Управління комплексом верстатів і роботів, складів, транспортних ліній та контрольних пристроїв, що забезпечують корекцію похибок верстатів, також є важливим аспектом розвитку. Планування та контроль за працею виробничої ділянки стають більш автоматизованими. Впровадження автоматизованих приводів з великим діапазоном безступінчатого регулювання частоти обертання двигунів і використання більш досконалих перетворювачів і двигунів також є актуальним.

Машини з ЧПК мають численні переваги, серед яких можна відмітити вищий рівень автоматизації, точніше виготовлення деталей та гнучкість в роботі. Впровадження такого обладнання дозволяє підприємствам зменшити кількість помилок операторів, скоротити час обробки та виготовлення деталей, а також забезпечити більш повне завантаження та швидке переналагодження

обладнання. Це сприяє підвищенню економічної ефективності підприємства, особливо в сучасних умовах.

Майбутні розробки та удосконалення верстатів з ЧПК спрямовані на більше використання ЕОМ, що дозволить підвищити їх багатофункціональність, енергетичну та економічну ефективність, підвищити продуктивність і поліпшити надійність роботи.

1.3. Можливості й переваги верстатів з ЧПК.

Верстати з числовим програмним керуванням (ЧПК) широко використовуються у різних сферах промисловості, і це не випадково. Ці машини мають численні переваги, які забезпечують високу точність і продуктивність обробки.

По-перше, точність верстатів з ЧПК може перевищувати точність ручного різання до 10 разів, зокрема, 5-осьовий верстат з ЧПК є ідеальним рішенням для проектів з високоякісною обробкою. Крім того, швидкість обробки на ЧПК значно перевершує ручне різання, що дозволяє ефективно знижувати час виробництва.

Економічна ефективність також є однією з переваг верстатів з ЧПК. Вони дозволяють зменшити трудові затрати та підвищити ефективність процесу виробництва. Крім того, завдяки використанню ЧПК, обробка матеріалів стає більш довговічною, з меншою кількістю відходів.

Гнучкість є ще однією вагомою перевагою верстатів з ЧПК. Завдяки можливості обробляти різні типи матеріалів без зміни налаштувань або інструментів, ці машини є універсальними в різних галузях промисловості, включаючи архітектуру, виробництво, інженерію, художній дизайн та будівництво.

Ефективність верстатів з ЧПК вражає своєю високою продуктивністю і швидкістю виробництва. Вони здатні працювати без перерви, 24 години на добу,

що дозволяє підприємствам збільшити обсяг виробництва і скоротити час, необхідний для виготовлення продукції.

Незважаючи на численні переваги верстатів з ЧПК, варто врахувати й деякі недоліки. Вартість машини та її обслуговування можуть бути досить високими, що ставить певні фінансові вимоги перед підприємствами. Крім того, обмеженість матеріалів для обробки на верстаті з ЧПК може становити обмеження для деяких проектів.

Точність обробки також може варіювати залежно від складності заготовки та використовуваних матеріалів. Потребується кваліфікований персонал для роботи з верстатами з ЧПК, і їхні навички можуть бути специфічними для даного типу обробки, що може ускладнити пошук роботи у випадку зміни професії.

Крім того, верстати з ЧПК найбільше підходять для виробництва від малого до середнього обсягу, і не є оптимальним варіантом для масового виробництва.

Враховуючи всі ці переваги та недоліки, верстати з ЧПК все ж залишаються важливими інструментами в сучасній промисловості, що допомагають підприємствам досягати високої точності, ефективності та гнучкості в обробці різних матеріалів.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. C++

Як бачите, незважаючи на старість, C ++ є відносно популярним і сьогодні, що є власним подвигом. Графік від stackoverflow.com.

Чи є C++ найкращою мовою програмування?

Відповідь залежить від перспективи та вимог. Деякі завдання можна виконати на C++, хоча і не дуже швидко. Наприклад, проектування екранів графічного інтерфейсу для додатків.

Інші мови, такі як Visual Basic, Python, мають вбудовані елементи дизайну графічного інтерфейсу. Тому вони краще підходять для задач типу графічного інтерфейсу.

Деякі мови сценаріїв, які забезпечують додаткову можливість програмування програм. Такі як MS Word і навіть photoshop, як правило, є варіантами Basic, а не C ++. C++ все ще широко використовується, а найвідоміше програмне забезпечення має основу на C++.

Хто використовує C++?

Деякі з найбільш помітних систем, що використовуються сьогодні, мають свої критичні частини написані на C++.

- Прикладами є Amadeus (продаж авіаквитків)
- Bloomberg (фінансова формація),
- Amazon (веб-комерція), Google (веб-пошук)
- Facebook (соціальні мережі)

Багато мов програмування залежать від продуктивності та надійності C++ при їх реалізації. Приклади включають:

- Java Віртуальні машини
- Перекладачі JavaScript (наприклад, Google V8)

- Веб-переглядачі (наприклад, Internet Explorer, Mozilla Firefox, Apple Safari та Google Chrome)
- Застосунки та веб-фреймворки (наприклад, фреймворк веб-служб .NET від Microsoft).

Програми, які включають локальні та широкі мережі, взаємодію з користувачем, числовий, графічний та доступ до баз даних, сильно залежать від мови C ++.

П'ять основних концепцій C++

Ось п'ять основних концепцій C++:

C++ змінні:

- Змінні є основою будь-якої мови програмування;
- Змінна - це просто спосіб зберігання деякої інформації для подальшого використання. Ми можемо отримати це значення або дані, звернувшись до "слова", яке описує цю інформацію;
- Після оголошення та визначення вони можуть бути використані багато разів у межах, в яких вони були задекларовані.

Структури управління C++:

- При запуску програми код зчитується компілятором рядок за рядком (зверху вниз, і здебільшого зліва направо). Це відоме як "потік коду";
- Коли код читається зверху вниз, він може зіткнутися з моментом, коли йому потрібно прийняти рішення. На основі рішення програма може перейти до іншої частини коду. Це може навіть змусити компілятор знову запустити певний фрагмент або просто пропустити купу коду;
- Ви можете думати про цей процес так, ніби ви вибираєте з різних курсів від Guru99. Ви вирішуєте, натисніть посилання та пропустіть кілька сторінок. Точно так само комп'ютерна програма має набір строгих правил для вирішення потоку виконання програми.

Пам'ятаючи про популярність та актуальність C++, нижче наведено 10 основних причин дізнатися про це:

1. Популярність C++ та висока зарплата

C++ є однією з найпопулярніших мов у світі. Його використовують близько 4,4 мільйона розробників по всьому світу. Крім того, розробники C++ досить затребувані, і вони займають одні з найбільш високооплачуваних робочих місць у галузі із середньою базовою заробітною платою $ 103,035 на рік.

2. C++ має багату підтримку бібліотеки

C++ має стандартну бібліотеку шаблонів (STL), яка дуже корисна, оскільки допомагає писати код компактно і швидко, як це потрібно. Він містить в основному чотири компоненти, тобто алгоритми, контейнери, функції та ітератори. Алгоритми бувають різних типів, такі як сортування, пошук тощо. Контейнери зберігають класи для реалізації різних структур даних, які зазвичай використовуються, таких як стеки, черги, хеш-таблиці, вектори, множини, списки, карти тощо. Функтори дозволяють налаштовувати роботу пов'язаної функції за допомогою переданих параметрів. Також ітератори використовуються для роботи над послідовністю значень.

C++ STL – Self-Paced сповнений концепцій C ++, таких як використання функцій, циклів, масивів, структури тощо, а також розширених концепцій, таких як алгоритми.

3. C++ має велику спільноту

Існує велика онлайн-спільнота користувачів та експертів C++, яка особливо корисна, якщо потрібна будь-яка підтримка. Існує багато ресурсів, таких як GeeksforGeeks тощо. доступний в Інтернеті щодо C++. Деякі з інших онлайн-ресурсів для C++ включають StackOverflow, cppreference.com, Standard C++ тощо.

4. C++ у базах даних

Існує багато сучасних баз даних, таких як MySQL, MongoDB, MemSQL тощо. які написані на C++. Це тому, що C++ досить сучасний і підтримує такі функції, як винятки, лямбда-вирази тощо. Багато баз даних, написаних на C++, використовуються майже у всіх використовуваних програмах, таких як YouTube, WordPress, Twitter, Facebook тощо.

Хочете зробити крок у світ програмування, тоді C ++ - це мова, яку вам потрібно вивчити. Зареєструйтеся сьогодні в Geeksforgeeks C++ Programming Foundation – самостійний курс і вивчіть такі теми, як введення/виведення на C++, управління потоком, оператори, цикли тощо.

5. C++ в операційних системах

Усі основні операційні системи, такі як Windows, Linux, Android, Ubuntu, iOS тощо. записуються в комбінації C і C++. Програми Windows написані на C ++, тоді як програми Android написані на Java разом із C / C ++ з нестандартним часом виконання підтримки C ++. Крім того, C++ можна використовувати для розробки ядра додатків в iOS. Загалом, C або C ++ використовуються в операційних системах через швидкість і сильно типізовану природу цих мов.

6. C++ в компіляторах

C++ ближче до апаратного рівня і є порівняно низькорівневою мовою. З цієї причини він використовується в багатьох компіляторах як бекенд мова програмування. Прикладом цього є GNU Compiler Collection (GCC), яка в даний час написана в основному на C++ разом з C.

7. C++ у веб-браузерах

Багато веб-браузерів розроблені з використанням C ++, таких як Chrome, Firefox, Safari тощо. Chrome містить C++ у движку візуалізації, механізмі JavaScript та інтерфейсі користувача. Firefox використовує в основному в движку рендеринга і трохи в інтерфейсі користувача. Safari також використовує C++ у движку рендерингу та рушії JavaScript. Всі ці веб-браузери та багато іншого використовують C++, особливо в движках рендеринга, оскільки він

забезпечує необхідну швидкість, необхідну для механізмів візуалізації, оскільки їм потрібно відображати вміст з прискореною швидкістю.

8. C++ у графіці

Програми, що вимагають графіки, такі як цифрова обробка зображень, комп'ютерний зір, програми запису екрану тощо. використовувати C++ завдяки високій швидкості. Це також може включати різні ігри, графіка яких є великою частиною своєї структури.

9. C++ у вбудованих системах

C++ ближче до апаратного рівня, тому він дуже корисний у вбудованих системах, оскільки програмне та апаратне забезпечення в них тісно пов'язані. Існує багато вбудованих систем, які використовують C ++, таких як розумні годинники, MP3-плеєри, системи GPS тощо.

10. C ++ є портативним

Програма, розроблена на C++, яку можна переміщати з однієї платформи на іншу. Це одна з основних причин того, що програми, які потребують розробки на кількох платформах або кількох пристроях, часто використовують C++.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Загальні відомості про верстат.

Верстат планується використовувати в якості машини для обробки пласких поверхонь, та зняття заданої товщини матеріалу. Зняття матеріалу вібуватиметься за допомогою абразивного круга, що обертається, а предмет який обробляється переміщується у двох площинах (двох осях). Загальна схема показана на рис. 3.1

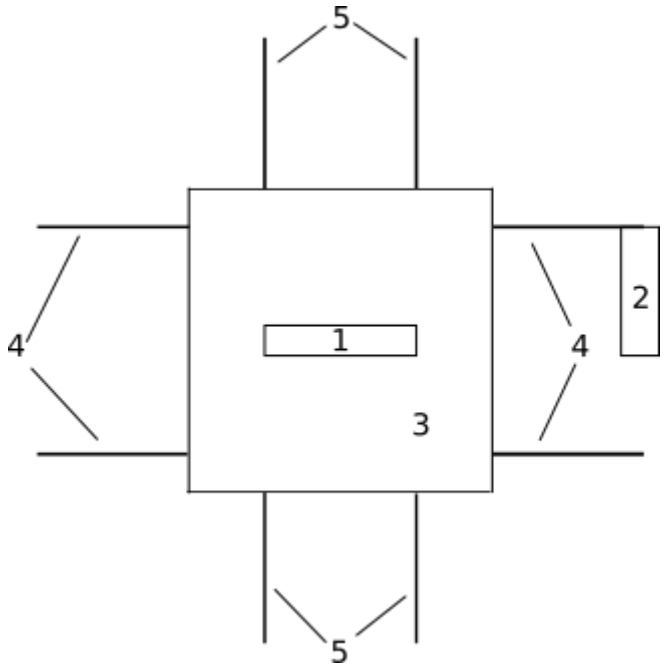

Рис. 3.1 Загальна схема верстата.

1 — абразивний круг.2 — привід абразивного круга. 3 — рухома станина. 4 — напрямні осі X. 5 — напрямні осі Y.

Оброблювана деталь рухатиметься по напрямних з ліва праворуч, та знизу верх за траєкторією “змійки”. На кінцях напрямних встановлені сенсори (на схемі не показано). Сенсори необхідні для задавання меж руху станини по напрямних. Обролювана деталь кріпитиметься для обробки на станині за допомогою електромагніта (на схемі не показано). Привід абразивного круга обладнаний сенсором обертів, для контролю роботи. Приводи переміщення станини відбуватимуться за допомогою крокових двигунів із окремими пристроями управління.

3.2. Загальні відомості про пристрій управління

Загальна схема пристрою управління показана на рис. 3.2.

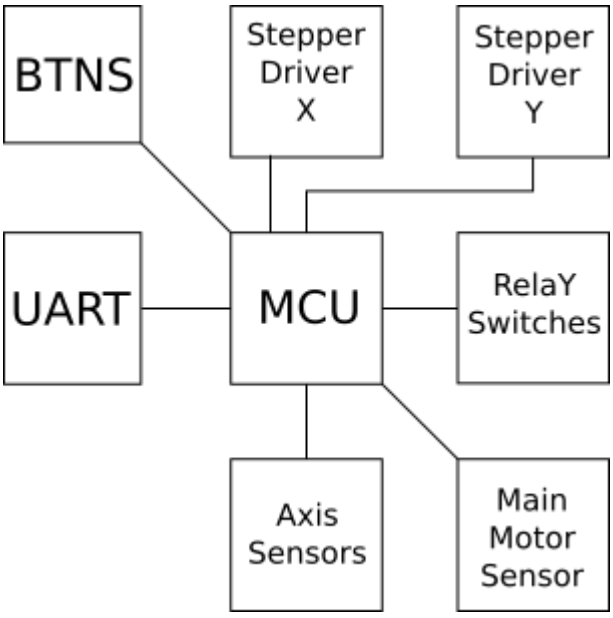

Рис. 3.2 Загальна схема пристрою управління

Пристрій управління верстатом побудований за допомогою мікроконтролера фірми Atmel ATMega-2560, модуль MCU. За допомогою модуля UART на зовнішній термінал передаватиметься службова та діагностична інформація, а також відбуватиметься програмування модуля MCU.

Для управління верстатом слугує модуль BTNS, що являє собою набір кнопок для запуску робочого циклу “Старт”, і в разі потреби або аварійного переривання робочого циклу “Стоп”.

Модуль Relay Switches керує вмиканням-вимиканням обладнанням верстата. Модуль містить два канали управління для вмикання утримуючого магніта на станині, і для вмикання привода абразивного круга (електричний двигун).

Для зворотнього зв’язку слугують два модулі Axis Sensors і Main Motor Sensor.

Axis Sensors — це чотири оптичні сенсори із цифровим виходом фізично розташовані на кінцях осей переміщення станини. У випадку, коли станина переміщується до граничної точки розташованій на будь якій осі, сенсор змінить свій стан. Зміну стану зафіксує мікроконтролер. Також кінцеві сенсори

застосовуються для діагностики осевих приводів.

Main Motor Sensor - це оптичний сенсор розташований на валу головного мотора, що слугує приводом абразивного круга. При роботі привода сенсор видає послідовність нулів і одиниць. Послідовність слугуватиме інформацією для оцінки роботи привода.

Stepper Driver X, Stepper Driver Y - модулі управління кроковими двигунами, що слугую приводами переміщення станини по осях “X” і “Y” віповідно.

Модулі управління у свою чергу керуються командами, що надходять із мікроконтролера за допомогою протокола STEP/DIR. Графічно робота протокола показана на рис.3

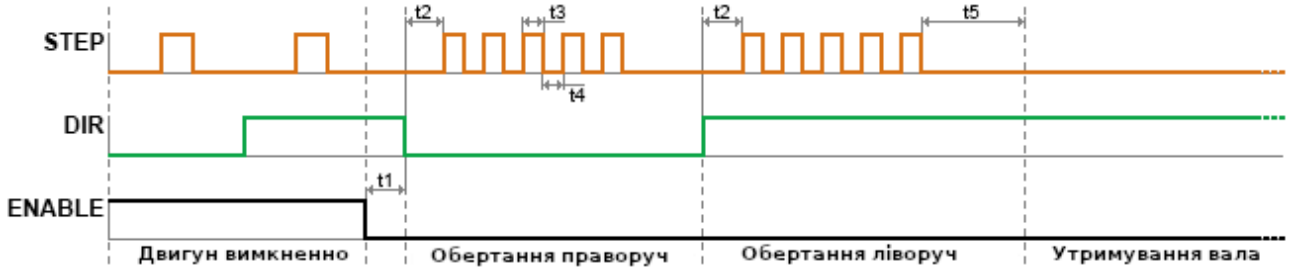

Рис. 3.3 Розгортка роботи протоколу STEP/DIR в часі.

Протокол управління STEP/DIR (крок/напрям), широко застосовується у верстатах із числовим програмним управлінням. Його застосування спростить розробку, а також дозволить роботу із любими управляючими модулями крокових двигунів, що підтримують цей протокол. Для роботи протоколу необхідно три лінії зв’язку. Оскільки одну лінію зв’язку можна розглядати як один біт, то для керування роботою управляючого модуля знадобиться три біти. Наприклад восьмибітний вихідний регістр мікроконтролера може керувати двома кроковими двигунами.

Біт STEP - за умови коли біт встановлений 1 кроковий двигун здійснює переміщення на 1 крок. При подачі послідовності бітів кількість кроків двигуна буде дорівнювати кількості бітів у послідовності.

Біт DIR - якщо біт встановлений в 1, кроковий мотор буде обертатись за годинниковою стрілкою, а при встановлені 0 проти годинникової стрілки.

Біт EN - біт для вмикання та вимикання приводу. Для увімкнення приводу біт встановлений у 0. При цьому якщо відсутні послідовності бітів на інших лініях то кроковий мотор буде утримувати поточну позицію.

Часові зактримки t1...t5 необхідні для узгодження роботи електронних пристроїв із більш інертними механічними пристроями та вузлами. Затримки t1,t2 і t5 необхідні для надання можливості механічному пристрою завершити свою робту. Затримки t3, t4 встановлюють швидкість обертання крокового двигуна.

3.3. Розробка програмного забезпечення.

Розробку програмного забезпечення почнемо із налаштування інтегрованого середовища розробки (IDE). Для використаного у пристрої мікроконтролера ATMEGA 2560 слід обрати відповідну плату. Для цього в меню “Інструменти” вибрати пункт “Плата”, і у списку що випаде вибрати плату “Arduino Mega” або “Mega 2650”. Порт до якого підключена плата налаштується автоматично.

Створимо кілька констант типу “byte”, в них буде задана прив’язка адреси і біта у виходному регістрі до фізичних виводів мікроконтролера, а також для подальшої зручності і наглядності коду задамо їх назви. Фрагмент коду показаний на рис. 3.4.

```
byte Dir_Pin_X = 9;
byte Step_Pin_X = 8;
byte En_Pin_X = 7;
byte Dir_Pin_Y = 6;
byte Step_Pin_Y = 5;
byte En_Pin_Y = 4;
```

Рис. 3.4 Фрагмент коду.

Оскільки для управління крововими двигунами буде використовуватись протокол STEP/DIR формувати управляючі послідовності будемо програмно. Із назв констант видно, що будемо використовувати 3 бітну версію протоколу.

На виході Dir_Pin_... - будуть формуватись сигнали напряму обертання

крокового двигуна.

На виході Step_Pin_... - кількість сформованих імпульсів буде дорівнювати кількості кроків, що їх виконає кроковий двигун.

На виході En_Pin_... - при наявності на цьому виході логічної “1” кроковий двигун буде вимкнений.

```
byte Magnet_Relay_Pin = 25;
byte Main_Motor_Relay_Pin = 26;

byte X_Axis_Sensor_Begin_Pin = 20;
byte X_Axis_Sensor_End_Pin = 21;
byte Y_Axis_Sensor_Begin_Pin = 22;
byte Y_Axis_Sensor_End_Pin = 23;

byte Main_Motor_Sensor_Pin = 24;

byte Start_Button_Pin = 10;
byte Stop_Button_Pin = 11;
```

Рис. 3.5. Фрагмент коду із описом констант прив’язки.

На рис. 3.5 показаний фрагмент коду із прив’язкою кінцевих сенсорів осей переміщення “..._Axis_Sensor_Begin_Pin” і “..._Axis_Sensor_End_Pin”. Також за допомогою кінцевих сенсорів перевіряється працездатність приводів осей. Деталь, яка підлягає обробці фіксується на рухомій станині за допомогою електромагніта. Прив’язка біта , що керуватиме роботою електромагніта описана у змінній “Magnet_Relay_Pin”. Робота двигуна привода абразивного круга прив’язується змінною “Motor_Relay_Pin”. Робота привода абразивного круга контролюється сенсором, що при обертанні має на виході послідовність бітів. При цьому не відбувається регулювання кількості обертів привода.

Прив’язка бітів кнопок управління “СТАРТ” і “СТОП” описана у відповідних змінних “Start_Button_Pin”, “Stop_Button_Pin”.

Для роботи деяких функцій і циклів будуть необхідні деякі допоміжні змінні. Фрагмент коду із описом допоміжних змінних показаний на рис. 3.6.

```
bool Direction_X = false;
bool Enable_X = false;
bool Direction_Y = false;
bool Enable_Y = false;

bool X_Axis_Sensor_Begin = false;
bool X_Axis_Sensor_End = false;
bool Y_Axis_Sensor_Begin = false;
bool Y_Axis_Sensor_End = false;

uint32_t f = 1000;
uint32_t t = 1000000/f/2;
uint32_t s;
```

Рис. 3.6 Фрагмент коду із описом допоміжних змінних.

Змінні “Direction_X”, “Direction_Y” описують біти напряму руху осевих приводів. За їх допомогою формуються біти “DIR” протоколу “STEP/DIR”. Змінні “Enable_X” та “Enable_Y” слугують для вмикання і вимикання приводів осей. При встановлені бітів у стан “0”, приводи будуть увімкнені і за відсутності послідоностей на інших управляючих бітах, пристрій управління кроковид двигуном переведе двигун у режим утримання позиції.

Змінними “f” і “t” задається швидкість обертання крокових двигунів. Змінною “f” задаємо кількість кроків за секунду, а змінною “t” задається пауза між бітами “STEP”. Ця пауза необхідна для надання можливості кроковому двигуну зробити перепозиціонування на один крок. Для різних моделей двигунів кількість кроків на оберт є різною. Найбільш розповсюдженими є двигуни із кількістю 200 кроків на оберт.

Змінна “s” задає кількість кроків для технологічного переміщення під час виконання робочого циклу.

При налагодженні верстата змінні “f”, “t” і “s” перезаписуються шляхом перепрограмування мікроконтролера відповідно до тежнологічного завдання.

Наступним кроком створимо функції для роботи із кнопками та сенсорами. Фрагмент коду для опитування стану кнопок показаний на рис. 3.7.

```
int Read_Button() {
  if (digitalRead(Start_Button_Pin) == LOW) {
    Serial.print ("Кнопка СТАРТ натиснена \n");
    return 1;
    }
  if (digitalRead(Stop_Button_Pin) == LOW) {
    Serial.print ("Кнопка СТОП натиснена \n");
    return 2;
    }
}
```

Рис. 3.7 Фрагмент коду функції опитування кнопок управління.

Функція “Read_Button” послідовно опитує стан кнопок управління. За допомогою умовних операторів “if” перевіряється стан кнопок. При натисненні однієї із кнопок на вході контролера з’явиться логічний “0”. За умови натиснення кнопки “СТАРТ” фукція поверне значення “1”, а кнопки “СТОП” повертаєме значення дорівнюватиме “2”.

Для опитування стану кінцевих сенсорів слугує функція “Read_sensor”. Робота функції схожа на роботу функції опитування кнопок. Основною відмінністю є більша кількість повертаємих значень. Такий підхід дозволяє отримати деяку гнучкість логіки роботи, а також описати своєрідну таблицю істиності станів кінцевих сенсорів. Якщо повертаємі значення функції дорівнюють “1” або “3”, то спрацювали кінцеві сенсори на початку осей “X” і “Y” відповідно. Значення “2”, “4” повертаються при спрацюванні сенсорів в кінцях осей “X” і “Y” відповідно. Значення “5” повертається при досягнені рухомої станини початкового положення. При поверненні значення “0” слід розуміти що станина знаходиться будь-де на площині обмеженій сенсорами, а відтак можна дозволити рух у будь-якому напрямку. Фрагмент коду функції показаний на рис. 3.8.

```
int Read_Sensor () {
  X_Axis_Sensor_Begin = digitalRead (X_Axis_Sensor_Begin_Pin);
  X_Axis_Sensor_End = digitalRead (X_Axis_Sensor_End_Pin);
  Y_Axis_Sensor_Begin = digitalRead (Y_Axis_Sensor_Begin_Pin);
  Y_Axis_Sensor_End = digitalRead (Y_Axis_Sensor_End_Pin);
    if (X_Axis_Sensor_Begin == true) {
      return 1;
    }
    if (X_Axis_Sensor_End == true) {
      return 2;
    }
    if (Y_Axis_Sensor_Begin == true) {
      return 3;
    }
    if (Y_Axis_Sensor_End == true) {
      return 4;
    }
    if (X_Axis_Sensor_Begin == true and
        Y_Axis_Sensor_Begin == true) {
      return 5;
    }
    if (X_Axis_Sensor_Begin == false and
        X_Axis_Sensor_End == false and
        Y_Axis_Sensor_Begin == false and
        Y_Axis_Sensor_End == false) {
      return 0;
    }

}
```

Рис. 3.8 Фрагмент коду із тілом функції “Read_Sensor”.

Наступним кроком створимо функцію управління двигуном привода абразивного круга. Фрагмент коду із описом функції показаний на рис. 3.9.

```
int Main_Motor_On_Off (bool Motor_Flag) {
  if (Motor_Flag == true) {
      digitalWrite (Main_Motor_Relay_Pin, HIGH);
    return 1;
  }
  if (Motor_Flag == false) {
      digitalWrite (Main_Motor_Relay_Pin, LOW);
    return 2;
  }
}
```

Рис. 3.9 Тіло функції управління двигуном привода абразивного круга

Функція на вході отримує логічну змінну “Motor_Flag”. При значенні вхідної змінної логічна “1”, на відповідному біті вихідного регістра встановлюється логічна “1”, що відповідає увімкненому стану двигуна, а також повертається значення “1”. При значенні “0” функція повертає значення “2”, а на біті вихідного регістра встановлюється логічний “0”, це значення відповідне вимкненому стану двигуна.

На рис. 3.10 показаний фрагмент коду функції діагностики привода абразивного круга.

```
bool Main_Motor_Test () {
  bool Motor_On = true;
    Main_Motor_On_Off (Motor_On);
      int pulses=0;
      if (digitalRead(Main_Motor_Sensor_Pin)) {
          pulses = pulses+1;
      }
      delay (1000);
      if (pulses >20) {
        return true;
      }
      else {
        Motor_On = false;
          Main_Motor_On_Off (Motor_On);
        return false;
      }
}
```

Рис. 3.10 Фрагмент коду діагностики привода абразивного круга.

На приводі встановлений сенсор який при роботі привода видає на виході послідовність бітів. При виклику функції “Main_Motor_Test” у фукцію “Main_Motor_On_Off” передається логічна “1”, і привід вмикається. У змінну “pulses” буде зберігатись кількість бітів, що надійшли від сенсора. Значення змінної буде збільшуватись тільки за умови наявності на виході сенсора логічної “1”. При досягненні змінної “pulses” значення 20 функція поверне логічну “1”, що відповідатиме справності привода. В іншому випадку повертається значення

логічного “0”, що відповідає несправності привода.

Функція “Magnet_On_Off” керує роботою утримуючого електромагніта. Алгоритмічно функція аналогічна функції керування привода абразивного круга. Фрагмент коду функції показаний на рис. 3.11.

```
int Magnet_On_Off (bool Magnet_Sw) {
  if (Magnet_Sw == true) {
      digitalWrite (Magnet_Relay_Pin, HIGH);
    return 1;
  }
  if (Magnet_Sw == false) {
      digitalWrite (Magnet_Relay_Pin, LOW);
    return 2;
  }
}
```

Рис. 11 Фрагмент коду управління утримуючим електромагнітом

Наступним етапом буде створення функції за допомогою якої крокові двигуни будуть переводитись у режим утримування позиції. Це свого роду електромагнітні гальма механічної системи верстата. Код функції показаний на рис. 3.12.

```
void Stop_Steppers () {
  digitalWrite (En_Pin_X, LOW);
  digitalWrite (Dir_Pin_X, HIGH);
  digitalWrite (Step_Pin_X, HIGH);
  digitalWrite (En_Pin_Y, LOW);
  digitalWrite (Dir_Pin_Y, HIGH);
  digitalWrite (Step_Pin_Y, HIGH);
}
```

Рис. 3.12 Фрагмент коду функції утримування позиції приводів.

Для утримування позиції крокових двигунів біти “STEP” і “DIR” встановлюються логічні “1”-ці, але приводи залишаються увімкненими шляхом встановлення бітів “En” у стан логічного “0”. Функція вмикає утримування позиції на обидвох осях одночасно.

Наступним кроком створимо функцію управління крововими двигунами у режимі руху до спрацювання кінцевого сенсора. Це необхідно для переміщення рухомої станини з будь-якого розташування на осі до спрацювання кінцевого сенсора без проміжних зупинок згідно заданого напрямку руху. Вхідна змінна “Direction_X” керує напрямом обертання крокового двигуна, а змінна “Enable_X” — вмиканням відповідного привода. Послідовності бітів “STEP” формуються програмним шляхом за допомогою цикла “while” і послідовним записом логічної “1” і із затримкою вказаній у глобальній змінній “t” записом логічного “0”. У тілі цикла перевіряється стан управляючої кнопки “СТОП”, за умови натискання якої відбудеться примусова аварійна зупинка верстата із виводом діагностичного повідомлення у послідовний порт.Код функції для осі “X” показаний на рис. 3.13. Аналогічно виглядає алгоритм функції для осі “Y”. Відмінність полягає у бітах управління.

```
void Rotate_Motor_X (bool Direction_X, bool Enable_X) {
  if (Direction_X == true and Enable_X == false) {
    digitalWrite (Dir_Pin_X, HIGH);
    while (Read_Sensor == 1) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite( Step_Pin_X, HIGH );
      delayMicroseconds(t);
      digitalWrite( Step_Pin_X, LOW );
      delayMicroseconds(t);
    }
   }
   if (Direction_X == false and Enable_X == false) {
    digitalWrite (Dir_Pin_X, LOW);
    while (Read_Sensor == 2) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite (Step_Pin_X, HIGH);
      delayMicroseconds (t);
      digitalWrite (Step_Pin_X, LOW);
      delayMicroseconds (t);
    }
   }
}
```

Рис. 3.13 Фрагмент коду із тілом функції управління кроковим двигуном.

Для переміщення на задану кількість кроків створимо функцію управління приводом із заданою кількістю кроків. На вході функції змінними “Dicrection_X” і “Enable_X” задаються напрям обертання привода, і вмикання привода відповідно. Послідовність бітів “STEP” формується програмним шляхом за допомогою параметричного циклу “for”, де кількість циклів прямо прив’язана до кількості кроків що задаються глобальною змінною “s”. Затримка між бітами задається глобальною змінною “t”. Також у тілі цикла перевіряється стан управляючої кнопки “СТОП”, для примусової аварійної зупинки верстата. Фрагмент з кодом функції показаний на рис. 3.14.

```
void Stepping_Motor_X (bool Direction_X, bool Enable_X) {
  if (Direction_X == true and Enable_X == false) {
    digitalWrite (Dir_Pin_X, HIGH);
    digitalWrite (En_Pin_X, LOW);
  if (Direction_X == false and Enable_X == false) {
    digitalWrite (Dir_Pin_X, LOW);
    digitalWrite (En_Pin_X, LOW)  ;
  }
    for (int i=0; i<s; i++) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite (Step_Pin_X, HIGH);
      delayMicroseconds(t);
      digitalWrite (Step_Pin_X, LOW);
      delayMicroseconds(t);
    }
  }
}
```

Рис. 3.14 Фрагмент коду із тілом функції для роботи двигуна із заданою кількістю кроків.

Наступним кроком створимо допоміжну функцію для повернення рухомої станини у початкове положення. Код функції показаний на рис. 3.15.

```
bool Return_Home_Pos () {
  bool dir = true;
  bool en = false;
    Rotate_Motor_X (dir, en);
    Rotate_Motor_Y (dir, en);
    Stop_Steppers ();
    if (Read_Sensor == 5) {
      return true;
    }
    if (Read_Sensor != 5) {
      return false;
    }
}
```

Рис. 3.15 Код функції повернення рухомої станини у стратову позицію.

Змінна “dir” задає напрям обертання крокових двигунів, а змінна “en” вмикає приводи. Напрям задається для двох осей одночасно. Обертанням

двигунів керують описані вище функції “Rotate_Motor_X” для осі “X ” і “Rotate_Motor_Y” для осі “Y”. При досягнені рухомою станиною стартового положення функція поверне значення логічної “1” і переведе двигуни в режим утримання позиції.

```
void Alarm_Stop () {
  digitalWrite (En_Pin_X, HIGH);
  digitalWrite (En_Pin_Y, HIGH);
  digitalWrite (Main_Motor_Relay_Pin, LOW);
  delay (3000);
  digitalWrite (Magnet_Relay_Pin, LOW);
  Serial.print ("УВАГА !!! Аварійна Зупинка\n");
}
```

Рис. 3.16 Код функції аварійної зупинки.

У разі виникнення помилок у роботі обладнання верстата, наприклад, вихід з ладу кінцевих сенсорів, привода абразивного круга або помилкової роботи верстата непередбаченої технологами виникають ситуації коли необхідно аварійно зупинити верстат. Для аварійної зупинки верстата або переривання робочого циклу слугує функція “Alarm_Stop”. Код функції показаний на рис. 3.16. В разі виклику функції біти увімкненя осевих приводів перемикаються в стан логічної “1”. в такому випадку приводи вимикаються. Встановленням біта управління приводом абразивного круга у логічний “0” вимикається двигун привода абразивного круга. Затримка у 3 секунди необхідна для повної зупинки абразивного круга, і тільки після затримки шляхм установки біта керування утримуючим електромагнітом вимикається сам магніт. Це необхідно для запобігання зриву оброблюваної деталі зі станини. Після виконання наведених вище операцій видається діагностичне повідомлення про помилку на послідовний порт.

Наступним кроком створимо функцію головного робочого циклу “Work_Cyckle”. Код функції наведений на рис. 3.17. При виклику функції за допомогою допоміжних змінних “dir_x”, “dir_y” задається початковий рух станини. Змінні “en_x”, “en_y” вмикають приводи осей. Змінні “motor”, “magnet” задають роботу приводу абразивного круга і утримуючого електромагніту

відповідно. Наступним кроком вмикаємо електромагніт (функція “Magnet_On_Off”) і виконуємо затримку у 0,5 секунди для насичення осердя електромагніта. Наступним кроком вмикаємо двигун привода абразивного круга (функція “Main_Motor_On_Off “) після чого виконуємо затрмку у 3 секунди для розручення абразивного круга і виходу на робочі оберти. У циклі “while” задаємо необхіну траєкторію руху станини. Умовою виходу із циклу буде досягненя станини кінцевого сенсора осі “Y”. Також в цілях безпеки в середині циклу перевіряємо стан кнопки управління “СТОП” для переривання робочого циклу або аварійної зупинки. Траєкторія руху станини показана на рис. 3.18. По досягнені фінішної точки траєкторії викликається допоміжна функція “Return_Home_Pos” що повертає рухому станину у стартову позицію, а також повертає значення логічної “1”. Наступним кроком вимикаємо двигун привода абразивного круга (функція “Main_Motor_On_Off). Затримка у 3 секунди потрібна для повної зупинки привода і останнім кроком вимикаємо утримуючий електромагніт (функція “Magnet_On_Off”).

```
void Work_Cyckle () {
  bool dir_x = true;
  bool dir_y = true;
  bool en_x = false;
  bool en_y = false;
  bool magnet = true;
  bool motor = true;
    Magnet_On_Off (magnet);
    delay (500);
    Main_Motor_On_Off (motor);
    delay (3000);
    while (Read_Sensor == 4) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      Rotate_Motor_X (dir_x, en_x);
      dir_x = false;
      Rotate_Motor_X (dir_x, en_x);
      Stepping_Motor_Y (dir_y, en_y);
    if (Return_Home_Pos == true) {
      Stop_Steppers ();
      motor = false;
      magnet= false;
      Main_Motor_On_Off (motor);
      delay (3000);
      Magnet_On_Off (magnet);
    }
  }
}
```

Рис. 3.17 Код функції робочого циклу.

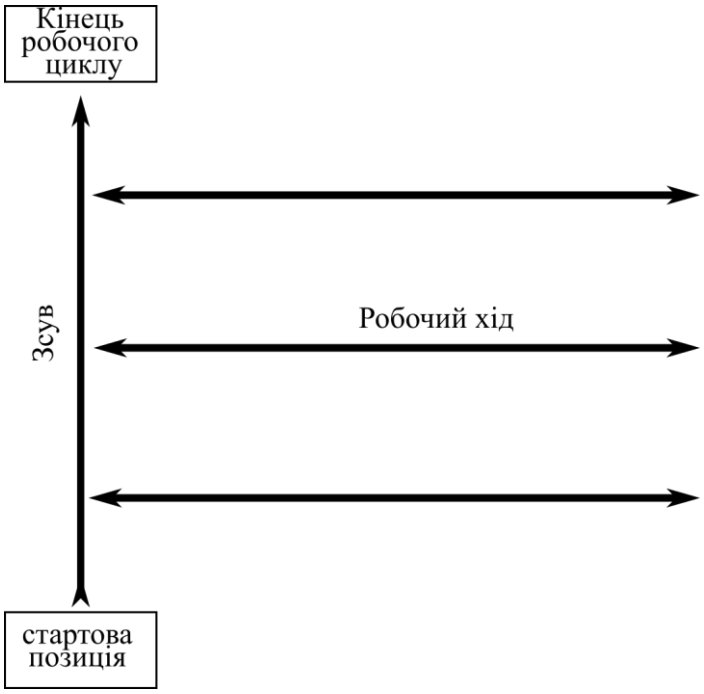

Рис. 3.18 Ілюстрація траєкторії руху станини.

Для стартової діагностики верстата створимо функцію “Power_On_Test”. Ця функція викликатиметься один раз при вмиканні живлення верстата і тестуватиме внутрішнє обладнання верстата. В разі виникненя несправності цикл перевірки перерветься і верстат перейде у аварійну зупинку. Код функції показаний на рис. 3.19. При виклику функції першим кроком тестується двигун привода абразивного круга (функція “Main_Motor_Test”). Якщо функція поверне значення логічного “0” відбудеться аварійна зупинка із виводом відповідного діагностичного повідомлення на послідовний порт. Наступним кроком будемо переміщувати станину по осях до спрацювання кінцевих сенсорів. За умови несправності сенсорів або приводів відбудеться аварійна зупинка із виводом діагностичної інформації. У випадку успішного проходження тестів станина встановлюється у стартову позицію і верстат готовий до роботи.

```
void Power_On_Test () {
      if (Main_Motor_Test == true) {
    Serial.print ("Головний привід ... OK\n");
   }
   else if (Main_Motor_Test == false) {
    Alarm_Stop ();
    Serial.print ("УВАГА !!! Помилка головного привода!!!\n");
   }
  bool dir = true;
  bool en = false;
    Rotate_Motor_X (dir, en);
    // вставити затримку для переміщення
      if (Read_Sensor == 1)  {
        Stop_Steppers ();
        Serial.print ("Сенсор 1 oci X ... OK\n");
      }
   dir = false;
    Rotate_Motor_X (dir, en);
      if (Read_Sensor == 2) {
        Stop_Steppers ();
        Serial.print ("Сенсор 2 oci X ... OK\n");
      }
   dir = true;
    Rotate_Motor_Y (dir, en);
      if (Read_Sensor == 3) {
        Stop_Steppers ();
        Serial.print ("Сенсор 1 oci Y ... OK\n");
      }
   dir = false;
    Rotate_Motor_Y (dir, en);
      if (Read_Sensor == 4) {
        Stop_Steppers ();
        Serial.print ("Сенсор 2 oci Y ... OK\n");
      }
   if (Return_Home_Pos == true) {
    Serial.print ("Стартова позиція встановлена\n");
   } else if (Return_Home_Pos == false) {
    Alarm_Stop ();
    Serial.print ("Увага помилка осевих приводів!!!\n");
   }
}
```

Рис. 3.19 Фрагмент коду. Функція первинного тестування обладнання.

Наступним кроком створимо функцію “setup”. В цій функції здійнюється налаштування портів мікроконтролера навхід і вихід, а також ініціалізується апаратний послідовний порт для виводу діагностичної інформації. Останнім кроком роботи функції буде тестування внутрішнього обладнання верстата і передача роботи головному циклу (функція “loop”).Функція викликається один раз при увікнені живлення мікроконтролера. Фрагмент коду із тілом функції показаний на рис. 3.20.

```
void setup () {
  pinMode (Dir_Pin_X, OUTPUT);
  pinMode (Step_Pin_X, OUTPUT);
  pinMode (En_Pin_X, OUTPUT);
  pinMode (Dir_Pin_Y, OUTPUT);
  pinMode (Step_Pin_Y, OUTPUT);
  pinMode (En_Pin_Y, OUTPUT);
  pinMode (X_Axis_Sensor_Begin_Pin, INPUT);
  pinMode (X_Axis_Sensor_End_Pin, INPUT);
  pinMode (Y_Axis_Sensor_Begin_Pin, INPUT);
  pinMode (Y_Axis_Sensor_End_Pin, INPUT);
  pinMode (Main_Motor_Sensor_Pin, INPUT);
  pinMode (Start_Button_Pin, INPUT);
  pinMode (Stop_Button_Pin, INPUT);
  pinMode (Magnet_Relay_Pin, OUTPUT);
  pinMode (Main_Motor_Relay_Pin, OUTPUT);

  Serial.begin (9600);
  Serial.print ("Тестуємо обладнання !!!\n");
  Power_On_Test ();
}
```

Рис. 3.20 Фрагмент коду із тілом функції “setup”.

Наступним кроком слід створити функцію головного циклу. Ця функція виконуватиметься постійно, поки є присутнє живлення мікроконтролера. У функції постійно опитується стан кнопок управління. В разі натиснення кнопки “СТАРТ” відбудеться запуск робочого циклу верстата, а кнопки “СТОП” його переривання і зупинка. Тіло функції проілюстровано на рис. 3.21.

```
void loop () {
  if (Read_Button == 1) {
    Serial.print ("Запуск робочого циклу ... OK\n");
    Work_Cyckle ();
  }
  if (Read_Button == 2) {
    Serial.print ("УВАГА!!! Зупинка робочого циклу\n");
  }
}
```

Рис. 3.21 Фрагмент коду із тілом функції головного циклу програми.

ВИСНОВКИ

Для пристрою, створеному на базі мікроконтролера фірми Atmel, створено програмне забезпечення (firmware, прошивку). Програмне забезпечення формує управляючі команди (сигнали) для блоків управління кроковими двигунами, надає вивід діагностичної інформації, забезпечує зчитування значень із сенсорів (датчиків), а також кнопок управління. Пристрій встановлювався на верстат для обробки матеріалів для управління роботою верстата. Програмне забезпечення є кросплатформеним, тобто легко переноситись на інші моделі шиирокоросповсюджених мікроконтролерів. А також має низький поріг входу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматичне управління процесами обробки металів різанням / Колодій О.С., Кюрчев С.В., Сушко О.В., Ковальов О.О. Мелітополь: ТОВ «Люкс», 2020. 136с.
2. O. Sushko, S. Kiurchev, O.S. Kolodii and oth. Grains Dynamic Strength Determination and the Optimal Combination of Components of a Diamondiferous Layer of Grinding Wheels. Modern Development Paths of Agricultural Production. Trend and Innovations. Tavria State Agrotechnological University, Melitopol, 2019. P. 259-266.
3. O.V. Sushko, O.S. Kolodii, O.V. Penyov. Individual forecasting of technical condition of machines and development of method for determining the conditional function of distributing their residual resource. Machinery & Energetics. Journal of Rural Production Research. Scientific Herald of National University of Life and Environmental Science of Ukraine. Kyiv. 2019. Vol. 10, № 4. P. 63-69.
4. А.М. Гуржій, В.В. Самсонов, Н.І. Поворознюк Імпульсна та цифрова техніка "Компанія СМІТ" Харків 2005.
5. Навчальний посібник з дисципліни «Проектування мікропроцесорних систем», розділ «Програмування мікроконтролерів родини AVR» для студентів напряму підготовки 6.050201 «Системна інженерія» кафедри Автоматики та управління у технічних системах / Укл.: А.О. Новацький – К: НТУУ „КПІ”, 2013 – 109 с.
6. Stroustrup, B. A Tour of C++, Addison-Wesley, 2019. – 256 p.
7. Meyers, S. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14, O’Reilly Media, 2020. – 334 p.
8. Lospinoso, J. C++ Crash Course: A Fast-Paced Introduction, No Starch Press, 2019. – 792 p.

9. Horton, I. Beginning C++20: From Novice to Professional, Apress, 2021. – 820 p.
10. Grimm, R. Modern C++ Programming Cookbook, Packt Publishing, 2020. – 742 p.
11. Bjarne Stroustrup. Programming: Principles and Practice Using C++, Addison-Wesley, 2020. – 1312 p.
12. Banks, A., Gupta, R. MQTT Version 5.0: Protocol and Applications. – OASIS Standard, 2019. – 120 p.
13. Yiu, J. The Definitive Guide to ARM Cortex-M23 and Cortex-M33 Processors, Newnes, 2020. – 624 p.
14. Margolis, M. Arduino Cookbook, O'Reilly Media, 2020. – 768 p.
15. Blum, J. Exploring Arduino: Tools and Techniques for Engineering Wizardry, Wiley, 2021. – 480 p.

ДОДАТКИ

```
byte Dir_Pin_X = 9;
byte Step_Pin_X = 8;
byte En_Pin_X = 7;
byte Dir_Pin_Y = 6;
byte Step_Pin_Y = 5;
byte En_Pin_Y = 4;

byte Magnet_Relay_Pin = 25;
byte Main_Motor_Relay_Pin = 26;

byte X_Axis_Sensor_Begin_Pin = 20;
byte X_Axis_Sensor_End_Pin = 21;
byte Y_Axis_Sensor_Begin_Pin = 22;
byte Y_Axis_Sensor_End_Pin = 23;

byte Main_Motor_Sensor_Pin = 24;

byte Start_Button_Pin = 10;
byte Stop_Button_Pin = 11;

bool Direction_X = false;
bool Enable_X = false;
bool Direction_Y = false;
bool Enable_Y = false;

bool X_Axis_Sensor_Begin = false;
bool X_Axis_Sensor_End = false;
bool Y_Axis_Sensor_Begin = false;
bool Y_Axis_Sensor_End = false;

uint32_t f = 1000;
uint32_t t = 1000000/f/2;
uint32_t s;

int Read_Button() {
  if (digitalRead(Start_Button_Pin) == LOW) {
    Serial.print ("Кнопка СТАРТ натиснена \n");
    return 1;
    }
  if (digitalRead(Stop_Button_Pin) == LOW) {
    Serial.print ("Кнопка СТОП натиснена \n");
    return 2;
    }
}

int Read_Sensor () {
  X_Axis_Sensor_Begin = digitalRead (X_Axis_Sensor_Begin_Pin);
  X_Axis_Sensor_End = digitalRead (X_Axis_Sensor_End_Pin);
  Y_Axis_Sensor_Begin = digitalRead (Y_Axis_Sensor_Begin_Pin);
```

```
  Y_Axis_Sensor_End = digitalRead (Y_Axis_Sensor_End_Pin);
    if (X_Axis_Sensor_Begin == true) {
      return 1;
    }
    if (X_Axis_Sensor_End == true) {
      return 2;
    }
    if (Y_Axis_Sensor_Begin == true) {
      return 3;
    }
    if (Y_Axis_Sensor_End == true) {
      return 4;
    }
    if (X_Axis_Sensor_Begin == true and
        Y_Axis_Sensor_Begin == true) {
      return 5;
    }
    if (X_Axis_Sensor_Begin == false and
        X_Axis_Sensor_End == false and
        Y_Axis_Sensor_Begin == false and
        Y_Axis_Sensor_End == false) {
      return 0;
    }

}

int Main_Motor_On_Off (bool Motor_Flag) {
  if (Motor_Flag == true) {
      digitalWrite (Main_Motor_Relay_Pin, HIGH);
    return 1;
  }
  if (Motor_Flag == false) {
      digitalWrite (Main_Motor_Relay_Pin, LOW);
    return 2;
  }
}

bool Main_Motor_Test () {
  bool Motor_On = true;
    Main_Motor_On_Off (Motor_On);
      int pulses=0;
      if (digitalRead(Main_Motor_Sensor_Pin)) {
          pulses = pulses+1;
      }
      delay (1000);
      if (pulses >20) {
        return true;
      }
      else {
        Motor_On = false;
          Main_Motor_On_Off (Motor_On);
```

```
      return false;
     }
}

int Magnet_On_Off (bool Magnet_Sw) {
  if (Magnet_Sw == true) {
     digitalWrite (Magnet_Relay_Pin, HIGH);
    return 1;
  }
  if (Magnet_Sw == false) {
     digitalWrite (Magnet_Relay_Pin, LOW);
    return 2;
  }
}

void Stop_Steppers () {
  digitalWrite (En_Pin_X, LOW);
  digitalWrite (Dir_Pin_X, HIGH);
  digitalWrite (Step_Pin_X, HIGH);
  digitalWrite (En_Pin_Y, LOW);
  digitalWrite (Dir_Pin_Y, HIGH);
  digitalWrite (Step_Pin_Y, HIGH);
}

void Rotate_Motor_X (bool Direction_X, bool Enable_X) {
  if (Direction_X == true and Enable_X == false) {
    digitalWrite (Dir_Pin_X, HIGH);
    while (Read_Sensor == 1) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite( Step_Pin_X, HIGH );
      delayMicroseconds(t);
      digitalWrite( Step_Pin_X, LOW );
      delayMicroseconds(t);
    }
   }
   if (Direction_X == false and Enable_X == false) {
    digitalWrite (Dir_Pin_X, LOW);
    while (Read_Sensor == 2) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite (Step_Pin_X, HIGH);
      delayMicroseconds (t);
      digitalWrite (Step_Pin_X, LOW);
      delayMicroseconds (t);
    }
   }
}
```

```
void Rotate_Motor_Y (bool Direction_Y, bool Enable_Y) {
  if (Direction_Y == true and Enable_Y == false) {
    digitalWrite (Dir_Pin_Y, HIGH);
    digitalWrite (En_Pin_Y, LOW);
    while (Read_Sensor == 3) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite (Step_Pin_Y, HIGH);
      delayMicroseconds (t);
      digitalWrite (Step_Pin_Y, LOW);
      delayMicroseconds (t);
    }
   }
   if (Direction_Y == false and Enable_Y == false) {
    digitalWrite (Dir_Pin_Y, LOW);
    digitalWrite (En_Pin_Y, LOW);
    while (Read_Sensor == 4) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite (Step_Pin_Y, HIGH);
      delayMicroseconds(t);
      digitalWrite (Step_Pin_Y, LOW);
      delayMicroseconds(t);
    }
   }
}

void Stepping_Motor_X (bool Direction_X, bool Enable_X) {
  if (Direction_X == true and Enable_X == false) {
    digitalWrite (Dir_Pin_X, HIGH);
    digitalWrite (En_Pin_X, LOW);
  if (Direction_X == false and Enable_X == false) {
    digitalWrite (Dir_Pin_X, LOW);
    digitalWrite (En_Pin_X, LOW)  ;
  }
    for (int i=0; i<s; i++) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite (Step_Pin_X, HIGH);
      delayMicroseconds(t);
      digitalWrite (Step_Pin_X, LOW);
      delayMicroseconds(t);
    }
  }
}

void Stepping_Motor_Y (bool Direction_Y, bool Enable_Y) {
  if (Direction_Y == true and Enable_Y == false) {
```

```
    digitalWrite (Dir_Pin_Y, HIGH);
    digitalWrite (En_Pin_Y, LOW);
  if (Direction_Y == false and Enable_X == false)   {
    digitalWrite (Dir_Pin_Y, LOW);
    digitalWrite (En_Pin_Y, LOW);
  }
    for (int i=0; i<s; i++) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      digitalWrite (Step_Pin_Y, HIGH);
      delayMicroseconds (t);
      digitalWrite (Step_Pin_Y, LOW);
      delayMicroseconds (t);
    }
  }
}

bool Return_Home_Pos () {
  bool dir = true;
  bool en = false;
    Rotate_Motor_X (dir, en);
    Rotate_Motor_Y (dir, en);
    Stop_Steppers ();
    if (Read_Sensor == 5) {
      return true;
    }
    if (Read_Sensor != 5) {
      return false;
    }
}

void Alarm_Stop () {
  digitalWrite (En_Pin_X, HIGH);
  digitalWrite (En_Pin_Y, HIGH);
  digitalWrite (Main_Motor_Relay_Pin, LOW);
  delay (3000);
  digitalWrite (Magnet_Relay_Pin, LOW);
  Serial.print ("УВАГА !!! Аварійна Зупинка\n");
}

void Work_Cyckle () {
  bool dir_x = true;
  bool dir_y = true;
  bool en_x = false;
  bool en_y = false;
  bool magnet = true;
  bool motor = true;
    Magnet_On_Off (magnet);
    delay (500);
    Main_Motor_On_Off (motor);
```

```
    delay (3000);
    while (Read_Sensor == 4) {
      if (Read_Button == 2) {
        Alarm_Stop ();
      }
      Rotate_Motor_X (dir_x, en_x);
      dir_x = false;
      Rotate_Motor_X (dir_x, en_x);
      Stepping_Motor_Y (dir_y, en_y);
    if (Return_Home_Pos == true) {
      Stop_Steppers ();
      motor = false;
      magnet= false;
      Main_Motor_On_Off (motor);
      delay (3000);
      Magnet_On_Off (magnet);
    }
  }
}

void Power_On_Test () {
      if (Main_Motor_Test == true) {
    Serial.print ("Головний привід ... OK\n");
   }
   else if (Main_Motor_Test == false) {
    Alarm_Stop ();
    Serial.print ("УВАГА !!! Помилка головного привода!!!\n");
   }
  bool dir = true;
  bool en = false;
    Rotate_Motor_X (dir, en);
    // вставити затримку для переміщення
      if (Read_Sensor == 1)  {
        Stop_Steppers ();
        Serial.print ("Сенсор 1 осі X ... OK\n");
      }
   dir = false;
    Rotate_Motor_X (dir, en);
      if (Read_Sensor == 2) {
        Stop_Steppers ();
        Serial.print ("Сенсор 2 осі X ... OK\n");
      }
   dir = true;
    Rotate_Motor_Y (dir, en);
      if (Read_Sensor == 3) {
        Stop_Steppers ();
        Serial.print ("Сенсор 1 осі Y ... OK\n");
      }
   dir = false;
    Rotate_Motor_Y (dir, en);
      if (Read_Sensor == 4) {
```

```
        Stop_Steppers ();
        Serial.print ("Сенсор 2 осі Y ... OK\n");
       }
   if (Return_Home_Pos == true) {
    Serial.print ("Стартова позиція встановлена\n");
   } else if (Return_Home_Pos == false) {
    Alarm_Stop ();
    Serial.print ("Увага помилка осевих приводів!!!\n");
   }
}

void setup () {
  pinMode (Dir_Pin_X, OUTPUT);
  pinMode (Step_Pin_X, OUTPUT);
  pinMode (En_Pin_X, OUTPUT);
  pinMode (Dir_Pin_Y, OUTPUT);
  pinMode (Step_Pin_Y, OUTPUT);
  pinMode (En_Pin_Y, OUTPUT);
  pinMode (X_Axis_Sensor_Begin_Pin, INPUT);
  pinMode (X_Axis_Sensor_End_Pin, INPUT);
  pinMode (Y_Axis_Sensor_Begin_Pin, INPUT);
  pinMode (Y_Axis_Sensor_End_Pin, INPUT);
  pinMode (Main_Motor_Sensor_Pin, INPUT);
  pinMode (Start_Button_Pin, INPUT);
  pinMode (Stop_Button_Pin, INPUT);
  pinMode (Magnet_Relay_Pin, OUTPUT);
  pinMode (Main_Motor_Relay_Pin, OUTPUT);

  Serial.begin (9600);
  Serial.print ("Тестуємо обладнання !!!\n");
  Power_On_Test ();
}

void loop () {
  if (Read_Button == 1) {
    Serial.print ("Запуск робочого циклу ... OK\n");
    Work_Cyckle ();
  }
  if (Read_Button == 2) {
    Serial.print ("УВАГА!!! Зупинка робочого циклу\n");
  }
}
```