Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Лисенко Д.С.
«____» _____ 2024 р.

Керівник проекту
кандидат технічних наук
_____ Улічев О.С.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 123 *"Комп'ютерна інженерія"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Лисенку Дмитру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU*

2. Керівник роботи *Улічев Олександр Сергійович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 07.08.2024 року

3. Строк подання студентом роботи до захисту *2.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Маркетингове та економічне обґрунтування ІТ-проєкту.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Доренська А.О.	05.10.2024	14.11.2024
Охорона праці	Марченко К.М., к.т.н., доцент	06.10.2024	16.11.2024

7. Дата видачі завдання « 6 » вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2024 р.	
3.	Розробка моделі компонента	20.10.2024 р.	
4.	Розробка структур даних	25.10.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2024 р.	
6.	Програмування алгоритмів	10.11.2024 р.	
7.	Розрахунок економічної ефективності	13.11.2024 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2024 р.	
9.	Оформлення ПЗ	17.11.2024 р.	
10.	Попередній захист роботи	2.12.2024 р.	

Дата видачі завдання
« 6 » вересня 2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Лисенко Д.С. Дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Метою розробки є дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Об'єктом дослідження є процес програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Предметом дослідження є методи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Методи дослідження базуються на методах цифрової схемотехніки, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, програматор, верстат ЧПК, AT90USB647-AU

ABSTRACT

Lysenko D.S. Research and software implementation of the CNC machine programmer system using the AT90USB647-AU controller. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the programmer system of CNC machines using the AT90USB647-AU controller.

The goal of the development is the research and software implementation of the CNC machine programmer system using the AT90USB647-AU controller.

The object of the study is the process of the CNC machine programmer using the AT90USB647-AU controller.

The subject of the study is the methods of the CNC machine programmer using the AT90USB647-AU controller.

Research methods are based on methods of digital circuitry, methods of mathematical statistics, and software development methods.

The result of the work is the software implementation of the CNC machine programmer system using the AT90USB647-AU controller.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Python environment.

Keywords: computer engineering, programmer, CNC machine, AT90USB647-AU

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

AC	–	розрахунок частотних характеристик лінеаризованого ланцюга
DC	–	розрахунок режиму по постійному струмі
FOUR	–	спектральний аналіз
MC	–	статистичний розрахунок по методу Монте-карло
NOISE	–	розрахунок спектральної щільності внутрішнього шуму
SENS	–	розрахунок малосигнальних чутливостей
STEP	–	параметричний аналіз
TF	–	розрахунок малосигнальних передатних функцій
TRAN	–	розрахунок перехідних процесів при впливі сигналів
WCASE	–	розрахунок на найгірший випадок
БД	–	база даних
ВДТ	–	відеодисплейний термінал
ГМД	–	гнучкий магнітний диск
ДП	–	друкуючий пристрій
ЕОМ	–	електронно обчислювальна машина
ІП	–	імпульсні перетворювачі
КІП	–	пристрій контролю
ОЗП	–	оперативно-запам’ятовувальний пристрій
ОЦ	–	обчислювальний центр
ПДП	–	прямий доступ до пам’яті
ПЕОМ	–	персональна електронно обчислювальна машина
ПЗ	–	програмне забезпечення
ПК	–	персональний комп’ютер
ПП	–	програмний продукт
ПРЦ	–	процесор
ПТМ	–	пожежно-технічний мінімум

ПУ – пульт управління

ПЧПУ – пристрій числового програмного управління

РПТ – резидентний перевіряючий тест

СУБДЧ – система управління БД

ТПАМТ – програма перевірки модуля адаптера магістралей і таймера

ТПКВП – програма модуля контролера вимірювальних перетворювачів

ТПКЕ – програма перевірки модуля контролера електроавтоматики

ТПКП – тестова програма перевірки модуля контролера приводів

ТПОЗП – тестова програма перевірки модуля ОЗП

ТППО – тестова програма перевірки модуля пульта оператора ПЧПУ

ТППРЦ – тестова програма перевірки модуля процесора

ТПШВИД – тестова програма перевірки швидкодію ПЧПУ

ЧПК – чисельно програмне управління

УФК – пристрій тестування плат

ВСТУП

Актуальність теми. Фахівець, що припускає використовувати всі переваги ЧПК, зобов'язаний, насамперед, вивчити його, причому одночасно із двох точок зору: спочатку з позиції програміста й з позиції оператора-верстатника. Першим завданням будь-якої людини, що опановує технологією обробки металів за допомогою ЧПК, є вивчення основ традиційної механічної обробки металів. Спроба вивчити нове обладнання без сукупності знань про загальні принципи металообробки порівнянна з вивченням керування літаком без знань про основи аеродинаміки. Основними поняттями при роботі з верстатами зі ЧПК є: попереднє й чистове гостріння, нарізування різьблення й т.п. З погляду програміста вивчення будь-якого верстата зі ЧПК необхідно починати з вивчення чотирьох основних моментів:

– основні компоненти верстата;

– рух інструмента або стола у всіх припустимих напрямках (осях);

– допоміжні елементи верстата;

– тонкості програмування кожної з функції верстата.

Знання конструкції верстата допомагає чітко усвідомити межі можливих операцій на даному верстаті для найбільш ефективного його застосування.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

– Дослідження системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

– Програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Об'єктом дослідження є процес програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Предметом дослідження є методи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Методи дослідження базуються на методах цифрової схемотехніки, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

– Розроблено вітчизняний продукт програматора верстатів ЧПК з використанням контролера AT90USB647-AU, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2024 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Досягнення точності у виробничих процесах часто передбачає використання передових технологій. Однією з таких технологій, яка відіграє вирішальну роль, є програмування з ЧПК (комп'ютерне числове керування).

Програмування з ЧПК є ключем до розкриття повного потенціалу верстатів з ЧПК, дозволяючи підприємствам виробляти складні й точні компоненти з ефективністю та послідовністю. Це код, який вдихає життя в складні конструкції. Він керує машинами, щоб ретельно формувати сировину в складні деталі, корисні в кількох галузях промисловості.

Програмування з числовим програмним керуванням (ЧПК) – це мистецтво та наука створення набору інструкцій, які комп'ютер може використовувати для керування інструментами та верстатами з ЧПК. Зрештою це створює складні деталі або вироби. Програміст ЧПК оцінює паперову або цифрову модель ЧПК деталі, після чого вони перетворюють дизайн у серію комп'ютерних інструкцій.

Потім ці інструкції передаються на автоматизовану машину через датчики та електроприводи. Машина (млин, фреза, шліфувальна машина або токарний верстат) слідує за точним керуванням комп'ютера, щоб пресувати, подрібнювати або різати матеріали, щоб надати їм бажаної форми.

У програмуванні ЧПК різні коди відіграють певну роль. Деякі з основних кодів включають:

– T-коди: Ідентифікаційні коди інструменту, що визначають ідеальний інструмент для обробки певної деталі заготовки.

– S-коди: коди швидкості, що вказують бажані швидкості шпинделя інструменту протягом операції обробки.

– N-коди: Ідентифіковані рядки або блоки машинного коду, які допомагають у кращій організації та розумінні створеного вручну коду ЧПК.

– M-коди: різні коди, що керують негеометричними діями верстата з

ЧПК, такими як керування шпинделем і охолоджуючою рідиною.

– G-коди: Геометричні коди вказують машині, як рухатися, коли рухатися, звідки рухатися та де зупиняти рух.

– F-коди: коди швидкості подачі, що представляють різні швидкості різання, з якими має рухатися інструмент під час процесу обробки.

– D-коди: Опис зміщення інструменту верстата з ЧПК, наприклад вимірювання того, наскільки далеко інструмент виступає з тримача інструменту або відстані між центральною лінією інструменту та ріжучою кромкою.

1.2 Область застосування

Інформація, що пояснює конструкцію верстата, звичайно приводиться в супровідній документації на верстат. Там же можна знайти відповіді на більшість питань про характеристики верстата й про його конструкцію. Наприклад:

– Які максимальні оберти шпинделя верстата?

– Скільки діапазонів швидкостей має шпиндель?

– Як велика потужність приводного електродвигуна для кожної з осей?

– Яке максимальна відстань переміщення інструмента або стола уздовж кожної осі?

– Скільки інструментів може поміститися в інструментальній головці (транспортері)?

– Яка найбільша швидкість прискореного переміщення?

– Яка найбільша швидкість різання?

Чим більше відомо про характеристики й конструкцію верстата зі ЧПК, тим легше буде процес керування цим устаткуванням у майбутньому.

Напрямки для руху (осі)

Кожної осі відповідає символ адреси, під яким видається рух у керуючу програму. Звичайно для цих цілей застосовують символи X, Y, Z, U, V і W для лінійних осей, а також A, B і C – для осей обертання.

Наприклад, рядок X3.5 в NC-програмі означає переміщення уздовж осі X на 3.5 дюйма від початку системи координат (припускаємо абсолютну систему кодування переміщень).

Круговий рух вимагає додаткової адреси (звичайно A, B, або C) і двох адрес для кінцевої точки. Додаткова адреса має числове значення, причому не в дюймах або міліметрах, а в градусах. Так наприклад, рядок B45 в NC-програмі означає обертання на 45 градусів щодо нуля програми навколо осі B.

Точка відліку для кожної з осей

Кожний верстат зі ЧПК має початкову точку для кожної з можливих осей переміщення інструмента або стола. Цю точку називають по-різному: нульова позиція, нуль системи координат, початкова позиція. Числове програмне керування вимагає переміщення в початкову точку по кожній з осей верстата як частина процедури налагодження верстата. Цим досягається синхронізація початкового фізичного розташування інструмента й початкових нульових значень суматорів системи зі ЧПК.

Безумовно те, що початкові позиції по кожній з осей досить розрізняються від верстата до верстата. Ви повинні уважно вивчити інструкцію із програмування вашого верстата для правильного розуміння розташування початкових позицій по кожній з осей .

Допоміжні елементи верстата

До числа додаткових елементів верстата ставляться: вимірники довжини робочої частини інструмента, пристрою зміни паллет і багато чого іншого. Список додаткового встаткування безупинно поповнюється.

У ряді випадків додаткові вузли можуть бути виготовлені виробником верстата, а в інші – сторонніми організаціями.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Studio Introduction to Software Framework (ASF)

Atmel Studio – це професійне інтегроване середовище розробки, яке пропонується абсолютно безкоштовно. Він підтримує мікроконтролери AVR і ARM Cortex-M.

Щоб розкрити свій потенціал, всім мікроконтролерам Atmel AVR необхідно програмне забезпечення. Для його створення й налагодження застосовується інтегроване середовище розробки (IDE), наприклад Atmel Studio 7. Це середовище розробки містить все необхідне для написання, компіляції й налагодження коду й дозволяє завантажувати його прямо у флеш-пам'ять на чипі мікроконтролера AVR без застосування додаткових програмних компонентів.

Середовище Atmel Studio 7 може використовуватися з будь-якими мікроконтролерами Atmel AVR і Atmel® | SMART на базі процесорів ARM®.

Це інтегроване середовище розробки призначене для написання й налагодження коду, що вбудовується, для пристроїв Atmel AVR і Atmel | SMART на базі процесорів ARM.

Редактор Atmel Studio 7 дозволяє легко вносити зміни в код і підвищує ефективність його написання. Редактор підтримує всі мікроконтролери AVR і Atmel | SMART на базі процесорів ARM і без праці взаємодіє з відладниками й комплектами розроблювача Atmel.

Також доступні інтегровані середовища розробки від наших партнерів по інструментах, у тому числі IAR Systems й ImageCraft.

Створення програми з нуля – непросте завдання, тому на нашім сайті ми виклали сотні прикладів готових проектів. Вони називаються вказівками по застосуванню й описують різні аспекти програмування й використання мікроконтролерів AVR. Просто відкрийте сторінку потрібного пристрою й перейдіть на вкладку "Документи", де зібрані всі актуальні вказівки по його застосуванню. Рекомендуємо почати з 8-розрядного МК ATmega2560 або 32-розрядного МК AT32UC3A3256.

– Вказівки по застосуванню для МК ATmega2560 (8-розрядний мікроконтролер megaAVR).

– Вказівки по застосуванню для МК AT32UC3A3256 (32-розрядний мікроконтролер AVR UC3).

Бібліотека Atmel Software Framework

Бібліотека Atmel Software Framework (ASF), убудована в Atmel Studio 7, являє собою колекцію готового до використання вихідного коду й включає понад 1600 прикладів проектів, написаних і оптимізованих експертами й випробуваних у сотнях промислових систем. Вхідні до складу колекції драйвери периферійних пристроїв, комунікаційні стеки й спеціальні прикладні бібліотеки прискорюють і спрощують розробку проектів.

Додаткові відомості про бібліотеку ASF можна знайти по посиланню Atmel Software Framework (ASF) .

На додаток до AVR Studio вас знадобляться й деякі апаратні засоби. Два найбільш важливі класи таких засобів – відладники й комплекти розроблювача.

megaAVR

Ця презентація представить сімейство мікроконтролерів Atmel® megaAVR®. Він додатково зосереджується на тому, як megaAVR дозволяє повторно використовувати знання, повторне використання коду та скорочення зусиль, необхідних для проектування та розробки продукту на основі цього сімейства мікроконтролерів.

Початковий комплект STK600

Крім відладника, багатьом клієнтам для початку розробки необхідна макетна плата. Компанія Atmel пропонує різноманітні асортименти таких плат: від невеликих оцінних комплектів до повноцінних готових типових проектів для деяких пристроїв. Найбільш популярною макетною платою для новачків є початковий комплект загального призначення STK600.

Додаткові комплекти розроблювача

Якщо у вашім проекті потрібне наявність кнопок, повзунків і коліс прокручування з підтримкою сенсорного уведення, рекомендуємо звернути увагу на комплект розроблювача QT600. Ця платформа дозволяє експериментувати із сенсорними технологіями Atmel і є найпростішим способом аналізу й оцінки сенсорних рішень Atmel.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктноорієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктноорієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення

вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веброзробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв'язанні проблеми. Їм не потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python

виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов’язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати, перетворювати, розділяти, об'єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веброзробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Програмування з ЧПК – це послідовність кроків, які перетворюють концептуальний проект у відчутну реальність. Розуміння тонкощів цього процесу проливає світло на те, як верстати з ЧПК забезпечують точність і ефективність виробництва.

Доступ до дизайну: моделі 3D або CAD

Подорож починається з того, що програміст ЧПК отримує доступ до тривимірної (3D) або моделі автоматизованого проектування (CAD) запланованої деталі. Це цифрове представлення служить основою для подальших кроків програмування. Модель CAD інкапсулює зовнішню геометрію деталі та інші важливі деталі, такі як розміри, характеристики та характеристики матеріалів.

Перехід на програмне забезпечення CAM

Перехід від моделі САПР до машинних інструкцій з ЧПК передбачає використання програмного забезпечення автоматизованого виробництва (CAM). Програмне забезпечення CAM діє як міст, перекладаючи геометричну інформацію з **CAD файлу** відчутні траєкторії. Верстат з ЧПК слідуватиме цим траєкторіям під час процесу обробки. Цей крок має вирішальне значення для оптимізації траєкторії інструменту на основі складності конструкції для забезпечення ефективності та точності.

Генерація G-коду

Після встановлення траєкторії інструменту програмне забезпечення CAM починає генерувати код обробки, широко відомий як G-код. G-код містить серію буквено-цифрових команд, які диктують дії машини. Ці команди містять інформацію про рух інструменту, швидкість шпинделя, швидкість подачі та інші важливі параметри для процеси механічної обробки.

Подача в машину з ЧПК

Програміст ЧПК передає інструкції в блок керування верстата з ЧПК. Зазвичай це досягається шляхом передачі файлу G-коду на машину за допомогою портативних пристроїв зберігання. Верстат з ЧПК виконує інструкції, готовий розпочати процес виробництва.

Початок виробництва деталей

Останнім кроком є запуск верстата з ЧПК для початку процесу виробництва деталей. Оскільки верстат з ЧПК отримує та інтерпретує G-код, він мобілізує свою автоматизовану систему. Потім він точно переміщує ріжучі інструменти для виконання запрограмованих траєкторій. Результатом є систематичне видалення матеріалу, поступове розкриття бажаної частини з неперевершеною точністю.

Види програмування ЧПК

Програмування з ЧПК – це багатогранна сфера з різними підходами, що задовольняють різні рівні складності, досвід користувача та вимоги до обробки. Розуміння різноманітних типів методів програмування ЧПК дає змогу зрозуміти, як вибрати найбільш прийнятний підхід для конкретних виробничих потреб.

Ручне програмування ЧПК

Ручне програмування з ЧПК є основоположним і основним стилем навчання верстатів з ЧПК. У цьому методі програмісти ЧПК вручну вводять команди безпосередньо в консоль керування верстатом ЧПК. Це ідеальний підхід для обробки простих деталей і часто використовується, коли операція проста.

Ручне програмування ЧПК добре підходить для ситуацій, коли геометрія деталей нескладна, а вимоги до програмування відносно прості. Однак його обмеження роблять його менш ідеальним для обробки складних деталей зі складними функціями.

Незважаючи на свою простоту, ручне програмування може бути трудомістким і займати багато часу. Кожна команда повинна бути введена ретельно, залишаючи місце для помилок. Крім того, відсутність попереджень або

сповіщень про можливі помилки збільшує ймовірність помилок. Це може призвести до пошкодження інструменту або машини.

Розмовне програмування

Це передбачає використання верстатів з ЧПК, оснащених вбудованим розмовним інтерфейсом програмування. На відміну від традиційного програмування за допомогою G-коду, цей метод дозволяє користувачам вводити команди простою англійською мовою. Він також може використовувати відповіді на серію керованих запитань, які надає система.

Розмовне програмування спрощує процес програмування ЧПК, роблячи його більш доступним для користувачів із різним рівнем досвіду. Це зручно для швидкого створення програм, особливо для простих дизайнів. Простий мовний інтерфейс підвищує зручність для користувача порівняно з традиційними методами G-коду.

Хоча розмовне програмування ефективне для простих проектів, його простота може стати обмеженням при роботі зі складнішими частинами. Крім того, недостатня точність G-коду може бути непридатною для застосувань, які вимагають найвищого рівня точності та складної обробки.

Програмування CAM

Програмне забезпечення автоматизованого виробництва (CAM) є широко поширеним методом програмування з ЧПК, особливо в галузях, які вимагають точності та універсальності. Програмне забезпечення CAM автоматизує створення траєкторій на основі 3D або CAD моделі деталі. Це забезпечує ефективний і точний засіб програмування ЧПК.

Програмування CAM забезпечує неперевершену швидкість і точність порівняно з ручними методами. Це дозволяє програмістам ЧПК візуалізувати траєкторії інструменту. Таким чином вони можуть визначити потенційні проблеми під час обробки. Програмне забезпечення CAM є гнучким, воно може легко передавати код між різними верстатами з ЧПК без необхідності переписувати всю програму.

Програмування CAM відмінно підходить для створення деталей зі складними функціями та підходить для простих і складних конструкцій. Можливість імітувати траєкторії інструменту та візуалізувати процес обробки покращує контроль програміста над процесом виробництва. Це зменшує ймовірність помилок.

Застосування програмування ЧПК в різних галузях промисловості

Програмування з ЧПК стало основою точного виробництва, знайшовши застосування в різних галузях. Верстати з ЧПК пропонують адаптивність і точність для створення різних продуктів, від складних прототипів до основних компонентів для різних секторів.

Сектор охорони здоров'я

Програмування з ЧПК революціонізує галузь охорони здоров’я, дозволяючи виготовляти індивідуальні протези, імплантати та інші штучні анатомічні частини. Заміщення суглобів, черепні імплантати та реставрації зубів виготовляються з точністю, щоб забезпечити ідеальне прилягання до пацієнтів.

Виробники використовують програмування з ЧПК для виготовлення передових компонентів для медичних пристроїв. Обробка з ЧПК дозволяє тестувати та виготовляти складні деталі, які мають вирішальне значення для функціональності та надійності медичного обладнання.

Побутова електроніка

Механічна обробка з ЧПК, що керується точним програмуванням з ЧПК, відіграє вирішальну роль у секторі побутової електроніки. Ця технологія широко використовується для створення прототипів і компонентів таких пристроїв, як ноутбуки, смартфони та інша електроніка. Висока точність, яку забезпечують верстати з ЧПК, є корисною для виготовлення деталей, таких як друковані плати.

Масштабованість програмування ЧПК робить його добре придатним для великомасштабного виробництва в промисловості побутової електроніки. Верстати з ЧПК можуть ефективно виробляти компоненти незмінної якості, щоб задовольнити попит на виробництво великої кількості.

Аерокосмічна та авіаційна промисловість

Програмування з ЧПК полегшує виробництво різних компонентів і деталей, необхідних для літаків. Пропелери, крила, рами, компоненти шасі та двигуни є одними з багатьох продуктів, створених із високою точністю за допомогою обробки з ЧПК.

Програмування з ЧПК також забезпечує високоякісне виготовлення компонентів, таких як лопаті гвинтів вертольотів і елементів космічних кораблів, таких як супутники та ракети.

Автомобільна промисловість

Верстати з ЧПК використовуються в різноманітних сферах застосування в автомобільному секторі, починаючи від деталей рідинних систем і підвісок і закінчуючи складними компонентами для внутрішніх і зовнішніх характеристик автомобілів. Програмування з ЧПК широко використовується в автомобільній промисловості для виготовлення основних компонентів для двигунів. Деякі приклади включають головки циліндрів, колінчасті вали, розподільні вали, клапани та різні кронштейни.

Поради щодо успішного програмування верстатів з ЧПК

Освоєння програмування верстатів з ЧПК – це поєднання технічного досвіду, уваги до деталей і прагнення до точності. Незалежно від того, чи є ви досвідченим програмістом з ЧПК, чи новачком, наведені нижче поради є безцінними для забезпечення успішного програмування верстатів із ЧПК та отримання оптимальних результатів обробки.

Опануйте основи

Перш ніж приступати до будь-якого проекту з програмування ЧПК, переконайтеся, що ви добре володієте основами машинного програмування. Незалежно від того, використовуєте розмовний або ручний метод програмування, ще раз перевірте обчислення, щоб гарантувати використання точних специфікацій машини. Точність верстата з ЧПК безпосередньо пов’язана з точністю основних математичних розрахунків.

Програми тонкого налаштування

Інвестуйте час у доопрацювання програми ЧПК, особливо в проектах, які передбачають значні обсяги обробки. Коли програму оптимізовано та підтверджено, що вона функціонує правильно, її часто можна «встановити та забути». Однак оптимізація окремих елементів машинного коду ЧПК може значно підвищити загальну ефективність процесу обробки.

Налаштуйте швидкість подачі на основі тонкощів проекту обробки. Пристосування швидкості подачі до конкретних вимог різних секцій деталі може оптимізувати видалення матеріалу, забезпечуючи незмінну якість.

Зрозумійте типи компенсації

Компенсація в програмуванні ЧПК відноситься до обліку відхилень, які можуть виникнути під час обробки. Ознайомтеся з різними типами компенсації, включаючи зміщення для пристосувань, знос інструменту, компенсацію радіуса різця та компенсацію довжини інструменту. Розуміння та відповідне застосування методів компенсації сприяє досягненню точних і безпомилкових результатів обробки.

Багато сучасних верстатів з ЧПК оснащені інструментами автоматичної попередньої настройки для програмування компенсацій. Використовуйте ці інструменти, щоб оптимізувати процес компенсації або вводити компенсації вручну, якщо необхідно. Ця увага до деталей гарантує, що верстат з ЧПК пристосовується до таких факторів, як знос інструменту. Це забезпечує послідовну та точну обробку.

Зробіть програми простими

У багатьох додатках обробки з ЧПК використання G-кодів як частини стандартних стандартних циклів може значно скоротити час програмування. Стандартні цикли дозволяють повторювати звичайні операції обробки без необхідності програмувати кожну операцію окремо. Це спрощує машинний код ЧПК і підвищує ефективність програмування.

Хоча простота має вирішальне значення, переконайтеся, що програма

ЧПК оптимізована для підвищення ефективності. Оцініть процес обробки, щоб визначити можливості оптимізації програми без шкоди для точності. Це може включати консолідацію повторюваних команд або використання ефективних траєкторій для мінімізації часу обробки.

Моніторинг і коригування в реальному часі

Інвестуйте в верстати з ЧПК із можливістю моніторингу в реальному часі. Інструменти моніторингу дозволяють операторам спостерігати за процесом обробки, коли він розгортається. Цей проактивний підхід дозволяє машиністам негайно виявляти потенційні проблеми, такі як знос інструменту або відхилення від запрограмованих траєкторій.

Активно стежте за процесом обробки та будьте готові вносити коригування в режимі реального часу, якщо це необхідно. Це може включати налаштування швидкості подачі, зміну траєкторії інструменту або вирішення неочікуваних проблем. Можливість робити спонтанні налаштування підвищує загальну ефективність і успіх програмування верстатів з ЧПК.

Мікросхема AT90USB647-AU

Характеристики мікросхеми: 64KB 2.7V~5.5V AVR 16MHz FLASH 48 TQFP-64(14x14) Блоки мікроконтролерів (MCU/MPU/SOC) ROHS.

Нижче наведемо більш детальні характеристики:

– Упаковка: Піднос.

– Статус частини: Активний.

– Ядро процесора: AVR.

– Розмір ядра: 8-розрядний.

– швидкість: 16 МГц.

– Підключення: EBI/EMI, I2C, SPI, UART/USART, USB OTG.

– Периферійні пристрої: Виявлення/скидання вимкнення, POR, PWM, WDT.

– Кількість входів/виходів: 48.

– Розмір програмної пам'яті: 64 Кб (64 Кб x 8).

– Тип пам'яті програми: СПАЛАХ.

– Розмір EEPROM: 2K x 8.

– Розмір оперативної пам'яті: 4K x 8.

– Напруга - живлення (Vcc/Vdd): 2,7 В ~ 5,5 В.

– Перетворювачі даних: A/D 8x10b.

– Тип осцилятора: внутрішній.

– Робоча температура: -40°C ~ 85°C (TA).

– Тип монтажу: Поверхневий монтаж.

– Пакет пристроїв постачальника: 64-TQFP (14x14).

– Пакет / футляр: 64-TQFP.

– Базовий номер продукту: AT90USB647.

3.2 Розробка структурної схеми

У будь-якого верстата зі ЧПК є певний набір функцій, які можуть бути перепрограмовані. Часто недороге встаткування допускає тільки ручне керування більшістю своїх функцій. Наприклад, багато фрезерних верстатів дозволяють запрограмувати тільки рух інструмента. А такі функції, як напрямок і швидкість обертання шпинделя, подача охолодження або зміна інструмента можуть бути зроблені оператором тільки вручну.

З іншої сторони більш дороге встаткування допускає програмне керування більшістю своїх функцій, а завдання оператора зводиться до завантаження заготівлі й зніманні готової деталі. У цьому випадку після запуску обробки оператор повністю вільний для виконання інших функцій.

Нижче наведений список найбільше часто використовуваних функцій разом з відповідними адресами:

– Керування шпинделем. Слово з адресою «S» використовується для завдання швидкості обертання шпинделя (число обертів у хвилину). Крім того, застосовується одна з допоміжних функцій: M03 задає обертання шпинделя за

годинниковою стрілкою; М04 визначає обертання шпинделя проти годинникової стрілки; нарешті М05 виключає двигун, що обертає шпиндель. Зверніть увагу на особливість токарських центрів – функцію постійної швидкості різання, що визначає швидкість обертання шпинделя так, щоб забезпечити постійне знімання точної кількості металу за хвилину.

– Пристрій автоматичної зміни інструмента (Обробні Центри). Слово з адресою «Т» повідомляє систему зі ЧПК про номер інструмента, що повинен бути використаний далі. У більшості випадків саме допоміжна функція М06 повідомляє про необхідність зробити фізичну зміну інструмента.

– Пристрій автоматичної зміни інструмента (Токарські центри). У більшості випадків використовується адреса «Т» і чотири цифри. Перші дві цифри визначають номер турелі, а останні – номер позиції з потрібним інструментом на турелі.

– Керування охолодженням. Допоміжна функція М08 використовується для включення подачі охолодної рідини. У ряді випадків доступна функція М07 для включення подачі холодоагенту у вигляді туману. Функція М09 виключає подачу охолодної рідини.

– Пристрій для автоматичної зміни паллет. Допоміжна функція М60 звичайно використовується для того, щоб замінити паллету.

Токарські верстати зі ЧПК в порівнянні зі звичайними верстатами мають більше широкі технологічні можливості:

Компенсація погрішностей

Верстати зі ЧПК дозволяють проводити компенсацію погрішностей як самого верстата, так і погрішностей, що виникають у процесі обробки деталей.

Перший метод полягає в проведенні компенсації систематичної складової погрішностей на основі апріорної інформації про погрішності верстата зі ЧПК або всієї технологічної системи, отриманої аналітичними розрахунками або експериментальними дослідженнями (пружних деформацій, люфту в приводі подач, погрішностей інструмента, ходового гвинта й ін.).

Компенсація накопиченої погрішності кроку ходового гвинта, зазорів у приводі подачі, теплових деформацій здійснюється за допомогою постійно діючих програм корекції, закладених на згадку пристрою ЧПК. Причому такі можливості дозволяють не тільки підвищити початкову точність верстата зі ЧПК, але й шляхом періодичного виміру мінливих погрішностей верстата і їхньої корекції в пам'яті пристрою ЧПК підтримувати точність верстата в заданій межі в процесі експлуатації.

При іншому методі компенсація погрішностей верстата зі ЧПК здійснюється на основі інформації, що надходить безупинно або переривчасто від систем зворотного зв'язка з датчиками, що вимірюють

– початкові погрішності верстата (наприклад, геометричні параметри);

– погрішності, що виникають у процесі експлуатації (наприклад, вібрації, теплові деформації, інструмента й ін.);

– погрішності, викликувані зовнішніми впливами (температура, вібрації, припуск на заготівлі, твердість оброблюваного матеріалу й ін.);

– погрішності безпосередньо оброблюваних деталей (розмірів, форми, шорсткості й ін.).

Застосування зворотних зв'язків ускладнює верстат зі ЧПК, але дозволяє компенсувати не тільки систематичну, але й випадкову складову погрішностей і проводити цю компенсацію безупинно в процесі експлуатації. По такому методі працюють адаптивні системи керування, які дозволяють компенсувати погрішності обробки, обумовлені таким випадковими факторами, як коливання припуску на заготівлі й твердості оброблюваного матеріалу, а також затуплення різального інструменту.

Методи контролю верстатів зі ЧПК

Для верстатів зі ЧПК розроблені й застосовуються нові види контролю їхньої початкової точності. При роботі верстата зі ЧПК вхолосту визначається комплексний показник початкової точності – погрішності позиціонування.

Комплексна перевірка верстата зі ЧПК в роботі проводиться шляхом обробки заданої деталі (або декількох різних деталей) на певних режимах різання з регламентацією розмірів, що доручаються. Застосовують також контроль

точності верстатів зі ЧПК обмацуванням спеціального еталона по заданій програмі й фіксації одержуваних відхилень. Контроль надійності роботи верстата й пристрою ЧПК проводиться при їхній безперервній роботі вхолосту по певній тест-програмі протягом заданого часу.

Рисунок 3.1 – Структурна схема системи

Діагностика технічного стану верстата зі ЧПК

Сучасні системи ЧПК дозволяють проводити діагностику технічного стану верстата й пристрою ЧПК, забезпечують оперативну видачу інформації про виникаючі несправності, а також дозволяють прогнозувати стан як окремих механізмів і блоків, так і верстатів зі ЧПК в цілому.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Розглянемо, як відбувається програмування верстатів ЧПК з використанням контролера AT90USB647-AU.

Програмування верстатів зі ЧПК – кадр керуючої програми

Кадр являє собою наступний в ієрархії після слова елемент тексту керуючої програми. Кожний кадр складається з одного або декількох слів, розташованих у певному порядку, які сприймаються системою ЧПК як єдине ціле й містять як мінімум одну команду. Відмітною ознакою кадрів як сукупності слів є те, що в них утримується вся геометрична, технологічна й допоміжна інформація, необхідна для виконання робочих або підготовчих дій виконавчих органів верстата. Робоча дія в цьому випадку означає обробку заготівлі за рахунок однократного переміщення інструмента по одній елементарній траєкторії (прямолінійне переміщення, переміщення по дузі й т.п.), а підготовча дія – дія виконавчих органів верстата для виконання або завершення робочої дії.

Приклад запису кадру: **N125 G01 Z-2.7 F30.**

Даний кадр складається із чотирьох слів: порядкового номера кадру «N125» і трьох слів «G01», « Z-2.7» і «F30», якими задається прямолінійне переміщення інструмента по осі Z до точки з координатою Z=–2,7 мм зі швидкістю подачі 30 мм/хв.

Текст **керуючої програми для верстата зі ЧПК** є не що інше, як сформована за певними правилами сукупність кадрів. У загальному випадку система ЧПК верстата виконує команди керуючої програми строго в порядку проходження кадрів, при цьому перехід до кожного чергового кадру здійснюється тільки по закінченні виконання попереднього кадру.

Таблиця 3.1 – Складові кадру

Символ	Призначення	Застосування
%	Початок програми	Символ позначення початку керуючої програми. У випадку використання програмоносія у вигляді перфострічки використовується також для зупинки носія даних при зворотному перемотуванні перфострічки.
LF або ПС	Кінець кадру	Символ позначення кінця кадру й переходу на наступний рядок тексту керуючої програми. У сучасних системах ЧПК використовується відносно рідко.
:	Головний кадр	Символ позначення кадру, у якому повинні бути записані вся інформація, необхідна для початку або поновлення обробки. У головному кадрі даний символ записується замість символу «N» у слові «Номер кадру».
/	Пропуск кадру	Символ, що позначає, що інформація, що втримується після нього до кінця кадру в якому він розташований, буде або спрацьовуватися, або пропускатися – залежно від настроювань на пульті керування. Якщо із цього символу починається кадр, то його дія поширюється на весь цей кадр.
(	Кругла дужка ліва	Символ, що позначає, що інформація, поміщена за ним, не повинна прийматися системою ЧПК до виконання. Використовується разом із символом «)».
)	Кругла дужка права	Символ, що позначає, що інформація, поміщена за ним, повинна прийматися системою ЧПК до виконання. Використовується разом із символом «(».
NUL або ПУС	Порожньо	Символ пропуску рядка перфострічки. Використовується тільки при написанні програми на перфострічці. Не сприймається системою ЧПК.

Щоб окремі кадри можна було зв'язати в єдину систему, крім буквених символів, наведених у табл. 3.1, при складанні керуючих програм для систем ЧПК застосовують і багато інших текстових символів. У табл. 1.3 наведені деякі додаткові символи, які рекомендовані до застосування стандартами України (ДСТ 20999-83 і ДСТ 19767-74).

Слова, довільно розташовані в тексті керуючої програми, сприймаються системою ЧПК всього лише як деякий набір слів і не будуть прийняті до виконання. Щоб дані слова являли собою команду, зрозумілу для системи ЧПК, вони повинні бути записані в кадрі керуючої програми в певному виді й порядку відповідно до прийнятого для конкретної системи ЧПК форматом кадру.

Міжнародний стандарт містить наступні загальні рекомендації, що ставляться до формату кадру при ручному програмуванні:

– Слова кадру, так само як і у звичайному тексті, повинні відділятися друг від друга інтервалами (пробілами). (Необхідно відзначити, що дана вимога не дотримується в багатьох сучасних системах ЧПК).

– Кожний кадр починається словом, що позначає номер кадру. Дане слово – «номер кадру» – містить буквений символ N і число, що відповідає порядковому номеру кадру.

– Кожний кадр закінчується словом, що позначає кінець кадру. Варіант написання, що рекомендується, даного слова для більшості імпортних систем ЧПК – LF, для вітчизняних систем ЧПК – ПС.

– Командні й розмірні слова, а також слова, що задають величини технологічних параметрів обробки деталей, розташовуються в тексті кадру між словами «номер кадру» і «кінець кадру» у порядку, певним виробником системи ЧПК. В одних системах ЧПК він може бути тільки строго певним, в інші – довільним.

Для зручності роботи міжнародний стандарт рекомендує наступний порядок розташування слів у кадрі: N..., G..., X..., Y..., Z..., U..., V..., W..., P..., Q..., R..., A..., B..., C..., I..., J..., K..., ..., LF.

Якщо задається швидкість подачі по одній певній осі координат, то слово, що позначає швидкість подачі, повинне випливати безпосередньо за словом, що задає переміщення по даній осі. Якщо задається швидкість подачі одночасно по двох і більше осях координат, то слово, що позначає швидкість подачі, повинне випливати безпосередньо за останнім словом, що задає переміщення по даних осях.

– Не допускається наявність в одному кадрі слів з однаковими буквеними символами. У той же час будь-яке слово може бути пропущено, якщо воно не є обов'язковим у даному кадрі.

– З метою зменшення обсягу тексту керуючої програми в кожному кадрі записується тільки нова інформація з відношення до попереднього кадру, при цьому незмінна частина інформації з попереднього кадру сприймається системою ЧПК за замовчуванням як діюча.

Як приклад проведемо аналіз структури наступного кадру:

N75 G01 Z-10.75 F0.3 S1800 T03 M08 LF

Таблиця 3.2 – Результат аналізу

Слово	Адреса	Число	Значення
N75	N	75	Слово, що складається з адреси N і порядкового числа 75, позначає порядковий номер кадру.
G01	G	01	Слово, що складається з адреси G і кодового числа 01, позначає підготовчу функцію, що пропонує виконати переміщення інструмента по прямої лінії із заданою швидкістю подачі.
Z-10.75	Z	-10.75	Слово, що складається з адреси Z і розмірного числа – 10.75, позначає координату розташування по осі Z точки, у яку інструмент повинен виконати переміщення у зв'язку з отриманою командою G01.

Продовження таблиці 3.2

Слово	Адреса	Число	Значення
F0.3	F	0.3	Слово, що складається з адреси F і розмірного числа 0.3, позначає величину швидкості подачі по осі Z при виконанні команди G01.
S1800	S	1800	Слово, що складається з адреси S і розмірного числа 1800, позначає величину швидкості обертання шпинделя
T03	T	03	Слово, що складається з адреси T і порядкового числа 03, позначає порядковий номер інструмента, установленого в робочу позицію із пристрою автоматичної зміни інструмента.
M08	M	08	Слово, що складається з адреси M і кодового числа 08, позначає допоміжну функцію, що пропонує при виконанні команди G01 включити подачу СОЖ.
LF	LF	-	Слово, що позначає закінчення кадру. Застосовується тільки у випадку рукописного складання тексту керуючої програми. При роздруківці програми на пристрої печатки не друкується.

Склад програми, кількість слів і структура слів визначається форматом кадру.

Наприклад для системи «Розмір– 4» верстатів типу 2204ВМ1Ф4 формат кадру має вигляд:

N79G2X+-43Y+-43Z+-43R+-43I+-43J+-43K+-43Y+-43B+-7

C+-7F41S51T46M2E7H7ПС

Тут **N7** означає семирозрядний номер кадру, тобто скільки кадрів може містити УП;

9G2 – дворозрядна підготовча функція, розбита на 9 груп;

X+–43Y – семирозрядна функція переміщення по осі X, остання цифра (**3)** означає кількість знаків після коми, тобто тисячні частки мм.;

E7 – витримка часу;

H7 – число повтору програми й т.д.

Таблиця 3.3 – Деякі системи ЧПК можуть мати таке число кадрів в УП

Система ЧПК	Максимальне число кадрів
Розмір – 4	9999999
Промінь – 430	32767
2В32	9999
Фанук – 6М	999
CNC – 600	9999
2С42	9999

Структура керуючої програми

Відповідно до міжнародних стандартів і ДСТ 20999-83 структура керуючої програми в загальному випадку підкоряється наступним правилам:

– У тексті керуючої програми повинна втримуватися геометрична, технологічна й допоміжна інформація, що необхідна для проведення заданої обробки. У кожному кадрі програми записується тільки та інформація, що змінюється стосовно попереднього кадру. При цьому виконання системою ЧПК незмінної інформації, що залишилася, припиняється тільки після надходження команди на її скасування (вид цієї команди й спосіб скасування визначається особливостями конкретної системи ЧПК).

– Кожна керуюча програма починається символом «початок програми», що подає системі керування сигнал про початок виконання програми. Вид символу «початок програми» залежить від особливостей застосовуваної системи ЧПК. Найбільше часто у вітчизняних і закордонних системах ЧПК використовується символ %. При цьому кадр із символом «початок програми» не нумерується. Нумерація кадрів починається з наступного кадру.

– Якщо керуючій програмі необхідно привласнити позначення, то його розташовують у кадрі із символом «початок програми» безпосередньо за символом.

– Якщо текст керуючої програми необхідно супроводити коментарем, наприклад відомостями про особливості налагодження верстата, то його розміщають перед символом «початок програми».

– Керуюча програма повинна закінчуватися символом «кінець програми», що подає системі керування сигнал на припинення виконання керуючої програми, зупинка шпинделя, приводів подач і вимикання охолодження. Інформація, поміщена в тексті керуючої програми після цього символу не повинна сприйматися системою ЧПК.

– Інформація, розташована в тексті керуючої програми між символами «початок програми» і «кінець програми» і укладена в круглі дужки не повинна прийматися системою ЧПК до виконання. При цьому в тексті усередині дужок не повинні застосовуватися символи «початок програми» і «головний кадр».

Таблиця 3.4 – Приклад того, як виглядає роздруківка тексту керуючої програми з погляду її структури

KORPUS-3506-12	Коментар із вказівкою назви деталі
% TP0147	Команда на початок виконання програми із вказівкою назви програми
N10 G54 X80 Y100 …	Послідовність кадрів, що містять інформацію з обробки деталі
…	
(Podrezka torca)	Інформація для програміста, не сприймана системою ЧПК
N75 G01 Z-10 F0.3 S1800 T03 M08…	Поновлення послідовності кадрів, що містять інформацію з обробки деталі
N435 M30	Команда на закінчення виконання програми

Кодування підготовчих і допоміжних функцій

У цей час на міжнародному ринку верстатів зі ЧПК широко застосовується понад 100 різні види систем зі ЧПК й стільки ж мов (кодів) програмування. Більшість із розповсюджених мов програмування в цілому однотипно й у своїй основі відповідають універсальній міжнародній мові програмування ІСО-7біт. Проте, у зв'язку з тим, що кількість команд використовуваних у програмуванні верстатів зі ЧПК, уже становить біля тисячі, і кожний виробник системи керування доповнює основні команди власними варіантами, немає можливості привести в одному місці відомості навіть по найбільш відомих мовах програмування.

У цей час для верстатів зі ЧПК в якості програмоносія прийнята восьмидорожкова перфострічка шириною 25,4 мм (1 дюйм), або її аналог (магнітний носій), на яких інформація для системи ЧПК представлена у вигляді двійкового семиелементного коду ІСО – 7 біт (ДСТ 13052 – 74). Кожному символу цього коду (цифри 0...9;букви латинського алфавіту A…Z; знаки %, дужки, +, – і ін.)відповідає цілком певна комбінація отворів (або сполучення **0** і **1**)на семи доріжках.

Восьма доріжка – для пробивання додаткового отворів у рядку (у символі), що дозволяє контролювати правильність перфорації й зчитування інформації у ЧПК.

G- і M-коди

Програмування обробки на сучасних верстатах зі ЧПК здійснюється мовою, що звичайно називають мовою ІСО (ISO) 7 біт, або мовою G- і M-кодів. Коди з адресою G, називані підготовчими, визначають настроювання СЧПУ на певний вид роботи. Коди з адресою M називаються допоміжними й призначені для керування режимами роботи верстата.

Наприклад, якщо програміст хоче, щоб інструмент переміщався по прямої лінії, він використовує G01. А якщо необхідно зробити зміну інструмента, то в програмі обробки він указує M06.

Для керування численними функціями верстата зі ЧПК застосовується досить велике число різних кодів. Проте, вивчивши набір основних G- і M-кодів, ви легко зможете створити керуючу програму.

G-код. Структура керуючої програми для верстата зі ЧПК

Керуюча програма складається з послідовності кадрів і звичайно починається із символу початок програми (%) і закінчується M02 або M30.

Кожний кадр програми являє собою один крок обробки й (залежно від ЧПК) може починатися з номера кадру (N1...N10 і т.д.), а закінчуватися символом кінець кадру (;).

Кадр керуючої програми складається з операторів у формі слів (G91, M30, X10. і т.д.). Слово складається із символу (адреси) і цифри, що представляє арифметичне значення.

Адреси X, Y, Z, U, V, W, P, Q, R, A, B, C, D, E є розмірними переміщенням, використовують для позначення координатних осей, уздовж яких здійснюються переміщення.

Слова, що описують переміщення, можуть мати знак (+) або (-). При відсутності знака переміщення вважається позитивним.

Адреси I, J, K означають параметри інтерполяції.

G – підготовча функція.

M – допоміжна функція.

S – функція головного руху.

F – функція подачі.

T, D, H – функції інструмента.

Символи можуть приймати інші значення залежно від конкретного ЧПУ.

Підготовчі функції (G коди)

G00 – швидке позиціонування.

Функція G00 використовується для виконання прискореного переміщення різального інструменту до позиції обробки або до безпечної позиції. Прискорене переміщення ніколи не використовується для виконання обробки, тому що

швидкість руху виконавчого органа верстата дуже висока. Код G00 відміняється кодами: G01, G02, G03.

G01 – лінійна інтерполяція.

Функція G01 використовується для виконання прямолінійних переміщень із заданою швидкістю (F). При програмуванні задаються координати кінцевої точки в абсолютних значеннях (G90) або збільшеннях (G91) з відповідними адресами переміщень (наприклад X, Y, Z). Код G01 відміняється кодами: G00, G02, G03.

G02 – кругова інтерполяція за годинниковою стрілкою.

Функція GO2 призначена для виконання переміщення інструмента по дузі (окружності) у напрямку годинникової стрілки із заданою швидкістю (F). При програмуванні задаються координати кінцевої точки в абсолютних значеннях (G90) або збільшеннях (G91) з відповідними адресами переміщень (наприклад X, Y, Z).

Параметри інтерполяції I, J, K, які визначають координати центра дуги окружності в обраній площині, програмуються в збільшеннях від початкової точки до центра окружності, у напрямках, паралельних осям X, Y, Z відповідно.

Код G02 відміняється кодами: G00, G01, G03.

G03 – кругова інтерполяція проти годинникової стрілки.

Функція GO3 призначена для виконання переміщення інструмента по дузі (окружності) у напрямку проти годинникової стрілки із заданою швидкістю (F). При програмуванні задаються координати кінцевої точки в абсолютних значеннях (G90) або збільшеннях (G91) з відповідними адресами переміщень (наприклад X, Y, Z).

Параметри інтерполяції I, J, K, які визначають координати центра дуги окружності в обраній площині, програмуються в збільшеннях від початкової точки до центра окружності, у напрямках, паралельних осям X, Y, Z відповідно.

Код G03 відміняється кодами: G00, G01, G02.

G04 – пауза.

Функція G04 – команда на виконання витримки із заданим часом. Цей код програмується разом з X або P адресою, що вказує тривалість часу витримки. Звичайно, цей час становить від 0.001 до 99999.999 секунд. Наприклад G04 X2.5 – пауза 2.5 секунди, G04 P1000 – пауза 1 секунда.

G17 – вибір площини XY.

Код G17 призначений для вибору площини XY у якості робочої. Площина XY стає визначальною при використанні кругової інтерполяції, обертанні системи координат і постійних циклів свердління.

G18 – вибір площини XZ.

Код G18 призначений для вибору площини XZ у якості робочої. Площина XZ стає визначальною при використанні кругової інтерполяції, обертанні системи координат і постійних циклів свердління.

G19 – вибір площини YZ.

Код G19 призначений для вибору площини YZ у якості робочої. Площина YZ стає визначальною при використанні кругової інтерполяції, обертанні системи координат і постійних циклів свердління.

G20 – уведення дюймових даних.

Функція G20 активізує режим роботи з дюймовими даними.

G21 – уведення метричних даних.

Функція G21 активізує режим роботи з метричними даними.

G40 – скасування корекції на радіус інструмента.

Функція G40 скасовує дію автоматичної корекції на радіус інструмента G41 і G42.

G41 – ліва корекція на радіус інструмента.

Функція G41 застосовується для включення автоматичної корекції на радіус інструмента який знаходиться ліворуч від оброблюваної поверхні (якщо дивитися від інструмента в напрямку його руху щодо заготівлі). Програмується разом з функцією інструмента (D).

G42 – права корекція на радіус інструмента.

Функція G42 застосовується для включення автоматичної корекції на радіус інструмента який знаходиться праворуч від оброблюваної поверхні (якщо дивитися від інструмента в напрямку його руху щодо заготівлі). Програмується разом з функцією інструмента (D).

G43 – корекція на положення інструмента.

Функція G43 застосовується для компенсації довгі інструмента. Програмується разом з функцією інструмента (H).

G52 – локальна система координат.

ЧПУ дозволяє встановлювати крім стандартних робочих систем координат (G 54-G59) ще й локальні. Коли ЧПУ верстата виконує команду G52, той початок діючої робочої системи координат зміщається на значення зазначене за допомогою слів даних X, Y і Z. Код G52 автоматично відміняється за допомогою команди G52 XO YO Z0.

G54 – G59 – заданий зсув.

Зсув робочої системи координат деталі щодо системи координат верстата.

G68 – обертання координат.

Код G68 дозволяє виконати поворот координатної системи на певний кут. Для виконання повороту потрібно вказати площина обертання, центр обертання й кут повороту. Площина обертання встановлюється за допомогою кодів G17, G18 і G19. Центр обертання встановлюється щодо нульової точки активної робочої системи координат (G54 – G59). Кут обертання вказується за допомогою R. Наприклад: G17 G68 X0. Y0. R120.

G69 – скасування обертання координат.

Код G69 скасовує режим обертання координат G68.

G73 – високошвидкісний цикл переривчастого свердління.

Цикл G73 призначений для свердлення отворів. Рух у процесі обробки відбувається на робочій подачі з періодичним виводом інструмента. Рух у вихідне положення після обробки йде на прискореній подачі.

G74 – цикл нарізування лівого різьблення.

Цикл G74 призначений для нарізування лівого різьблення мітчиком. Рух у процесі обробки відбувається на робочій подачі, шпиндель обертається в заданому напрямку. Рух у вихідне положення після обробки йде на робочій подачі зі зворотним обертанням шпинделя.

G80 – скасування постійного циклу.

Функція, що скасовує будь-який постійний цикл.

G81 – стандартний цикл свердління.

Цикл G81 призначений для зацентрування й свердлення отворів. Рух у процесі обробки відбувається на робочій подачі. Рух у вихідне положення після обробки йде на прискореній подачі.

G82 – свердління з витримкою.

Цикл G82 призначений для свердління й зенкування отворів. Рух у процесі обробки відбувається на робочій подачі з паузою наприкінці. Рух у вихідне положення після обробки йде на прискореній подачі.

G83 – цикл переривчастого свердління.

Цикл G83 призначений для глибокого свердлення отворів. Рух у процесі обробки відбувається на робочій подачі з періодичним виводом інструмента в площину відводу. Рух у вихідне положення після обробки йде на прискореній подачі.

G84 – цикл нарізування різьблення.

Цикл G84 призначений для нарізування різьблення мітчиком. Рух у процесі обробки відбувається на робочій подачі, шпиндель обертається в заданому напрямку. Рух у вихідне положення після обробки йде на робочій подачі зі зворотним обертанням шпинделя.

G85 – стандартний цикл розточування.

Цикл G85 призначений для розгортання й розточування отворів. Рух у процесі обробки відбувається на робочій подачі. Рух у вихідне положення після обробки йде на робочій подачі.

G86 – цикл розточування із зупинкою обертання шпинделя.

Цикл G86 призначений для розточування отворів. Рух у процесі обробки відбувається на робочій подачі. Наприкінці обробки відбувається зупинка шпинделя. Рух у вихідне положення після обробки йде на прискореній подачі.

G87 – цикл розточування з відводом вручну.

Цикл G87 призначений для розточування отворів. Рух у процесі обробки відбувається на робочій подачі. Наприкінці обробки відбувається зупинка шпинделя. Рух у вихідне положення після обробки йде вручну.

G90 – режим абсолютного позиціонування.

У режимі абсолютного позиціонування G90 переміщення виконавчих органів виробляються щодо нульової точки робочої системи координат G 54-G59 (програмується, куди повинен рухатися інструмент). Код G90 відміняється за допомогою коду відносного позиціонування G91.

G91 – режим відносного позиціонування.

У режимі відносного (інкрементального) позиціонування G91 за нульове положення щораз приймається положення виконавчого органа, що він займав перед початком переміщення до наступної опорної точки (програмується, на скільки повинен переміститися інструмент). Код G91 відміняється за допомогою коду абсолютного позиціонування G90.

G94 – швидкість подачі в дюймах/міліметрах у хвилину.

За допомогою функції G94 зазначена швидкість подачі встановлюється в дюймах за 1 хвилину (якщо діє функція G20) або в міліметрах за 1 хвилину (якщо діє функція G21). Програмується разом з функцією подачі (F). Код G94 відміняється кодом G95.

G95 – швидкість подачі в дюймах/міліметрах на оберт.

За допомогою функції G95 зазначена швидкість подачі встановлюється в дюймах на 1 оберт шпинделя (якщо діє функція G20) або в міліметрах на 1 оберт шпинделя (якщо діє функція G21). Тобто швидкість подачі F синхронізується зі швидкістю обертання шпинделя S. Код G95 відміняється кодом G94.

G98 – повернення до вихідної площини в циклі.

Якщо постійний цикл верстата працює разом з функцією G98, то інструмент вертається до вихідної площини наприкінці кожного циклу й між всіма оброблюваними отворами. Функція G98 відміняється за допомогою G99.

G99 – повернення до площини відводу в циклі.

Якщо постійний цикл верстата працює разом з функцією G99, то інструмент вертається до площини відводу між всіма оброблюваними отворами. Функція G99 відміняється за допомогою G98.

Таблиця 3.5 – М-коди

M00	Запрограмована зупинка – виконання програми тимчасово припиняється
M01	Запрограмована зупинка на вибір – виконання програми тимчасово припиняється, якщо активовано режим зупинки на вибір
M03	Пряме обертання шпинделя – шпиндель обертається за годинниковою стрілкою
M04	Зворотне обертання шпинделя – шпиндель обертається проти годинникової стрілки
M05	Зупинка шпинделя
M06	Автоматична зміна інструмента M06 T02
M08	Включення подачі охолодної рідини
M09	Вимикання подачі охолодної рідини
M30	Кінець програми, переклад курсору до початку програми

Рисунок 3.2 – Функціональна схема системи

Функціональна схема складається с наступних блоків:

– Головне вікно програми.

– G-коди.

– M-коди.

– Блок завдання максимальних обертів шпинделя верстата.

– Блок завдання діапазонів швидкостей шпинделя.

– Блок завдання потужності приводного електродвигуна для кожної з осей.

– Блок завдання максимальної відстані переміщення інструмента або стола уздовж кожної осі.

– Блок завдання кількості інструментів може поміститися в інструментальній головці (транспортері).

– Блок завдання найбільшої швидкості прискореного переміщення.

– Блок завдання найбільшої швидкості різання..

– Блок керування шпинделем.

– Блок керування пристроєм автоматичної зміни інструмента (Обробні Центри).

– Блок керування пристроєм автоматичної зміни інструмента (Токарські центри).

– Блок керування охолодженням.

– Блок керування пристроєм для автоматичної зміни паллет.

– Блок компенсації погрішностей.

– Блок контролю початкової точності.

– Блок діагностики технічного стану верстата.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає

уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

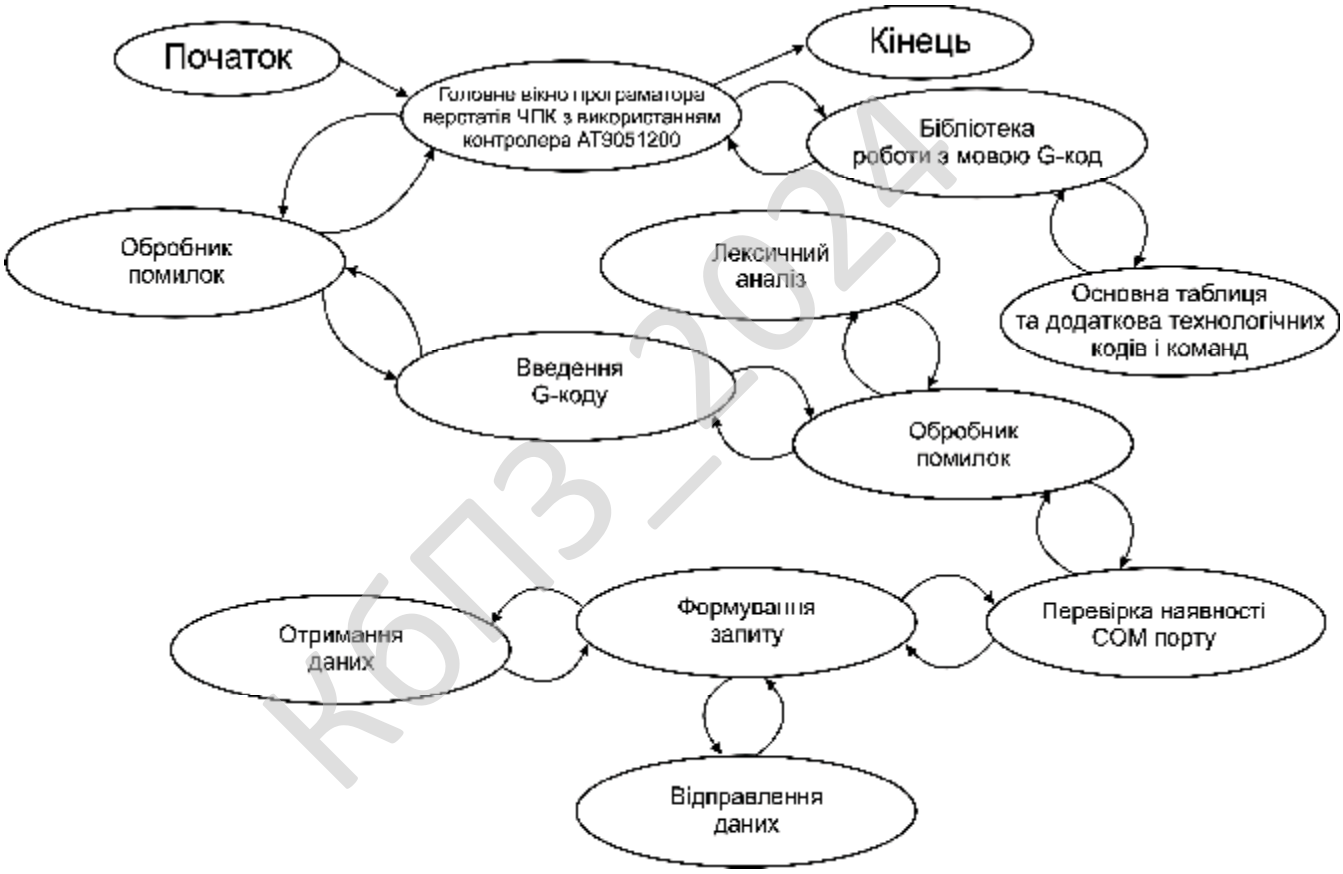

Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Під час роботи над магістерською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято

позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка

потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

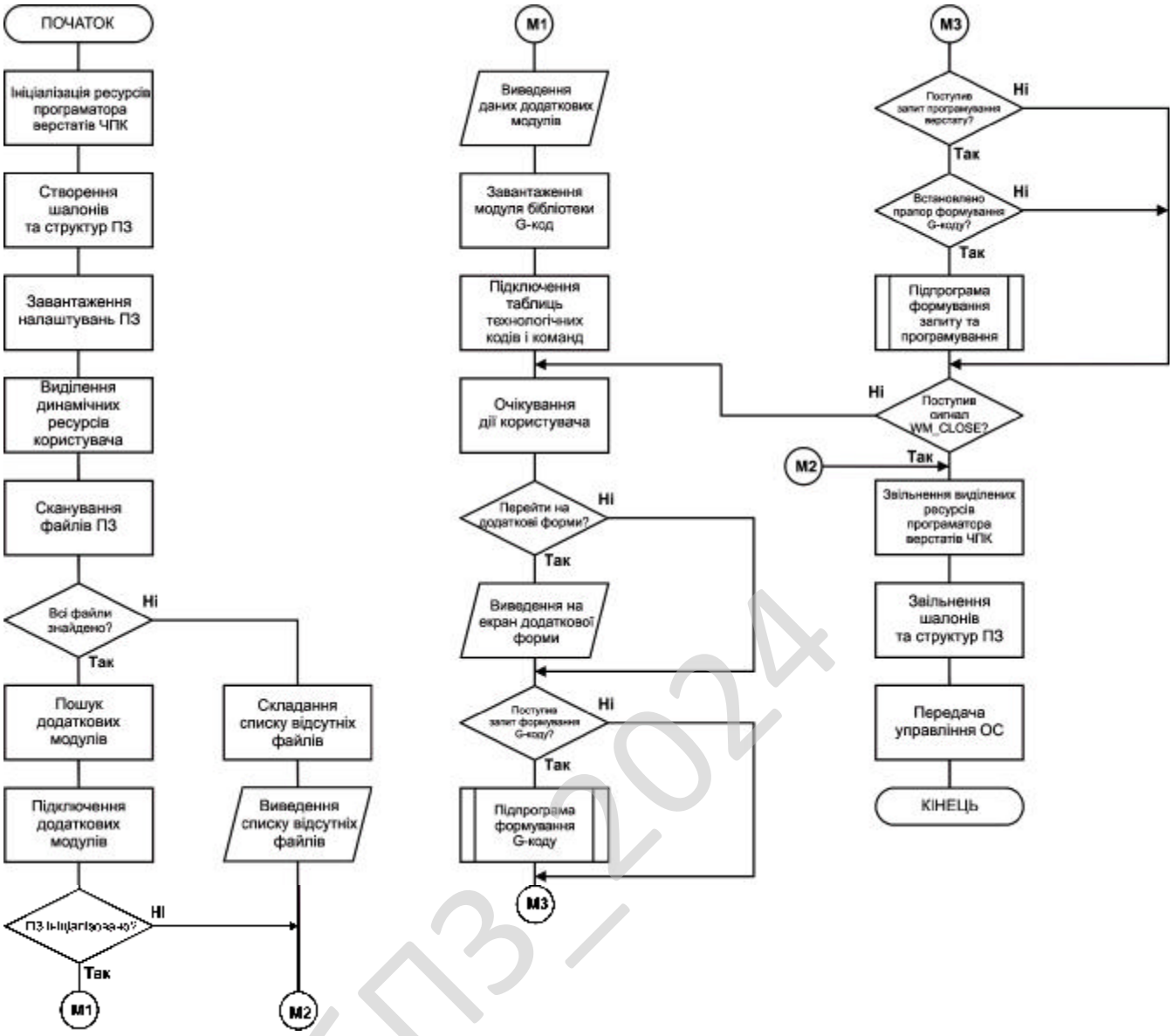

Рисунок 4.1 – Блок-схема основної програми

Також при розробці магістерської дипломної роботи було використано наступні підходи UML: Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

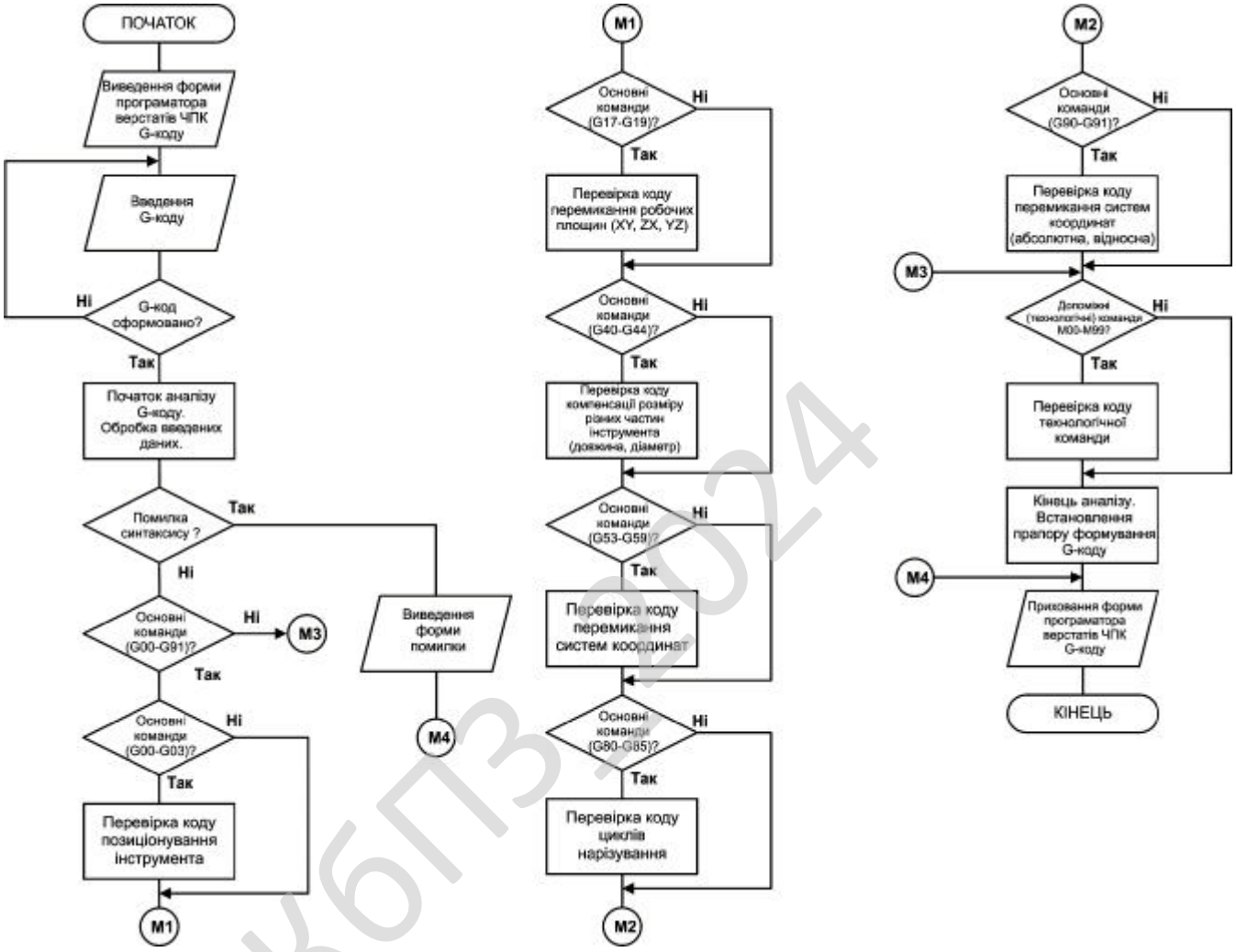

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Система, яка була розроблена на Python, орієнтована на взаємодію з мікроконтролером AT90USB647-AU для програмування верстатів з числовим програмним керуванням (ЧПК).

Основні компоненти системи включають:

1. Взаємодію з контролером через USB для передачі команд G-коду
2. Обробку даних і виконання операцій на верстаті
3. Керування кроковими двигунами та іншими механічними елементами

Архітектура системи

1. Модуль обміну даними через USB
2. Модуль генерації G-коду
3. Модуль управління верстатом
4. Модуль обробки зворотного зв'язку
5. Інтерфейс користувача для взаємодії з системою

Основні модулі

1. Модуль обміну даними через USB

Цей модуль відповідає за комунікацію з мікроконтролером через USB. Використовується бібліотека pySerial для встановлення з'єднання і передачі команд на контролер.

```
import serial

class USBCommunicator:
    def __init__(self, port, baudrate=115200):
        self.ser = serial.Serial(port, baudrate, timeout=1)

    def send_command(self, command):
        self.ser.write(command.encode())
        response = self.ser.readline().decode('utf-8').strip()
        return response

    def close_connection(self):
        self.ser.close()

# Приклад використання
usb_com = USBCommunicator('/dev/ttyUSB0')
```

```
response = usb_com.send_command("G01 X10 Y10 F200")
print("Response from controller:", response)
usb_com.close_connection()
```

2. Модуль генерації G-коду

Модуль для створення G-коду, який керуватиме переміщенням осей верстата. Цей код генерує команди для керування кроковими двигунами.

```
class GCodeGenerator:
    def __init__(self):
        self.commands = []

    def move_to(self, x, y, speed):
        command = f"G01 X{x} Y{y} F{speed}"
        self.commands.append(command)

    def get_gcode(self):
        return '\n'.join(self.commands)

# Приклад використання
generator = GCodeGenerator()
generator.move_to(10, 10, 300)
generator.move_to(20, 5, 200)
print(generator.get_gcode())
```

3. Модуль управління верстатом

Модуль для відправки G-коду на контролер та управління різними механічними компонентами верстата. Він використовує USBCommunicator для передачі команд на мікроконтролер.

```
class MachineController:
    def __init__(self, communicator):
        self.communicator = communicator

    def execute_gcode(self, gcode):
        commands = gcode.split('\n')
        for command in commands:
            response = self.communicator.send_command(command)
            print(f"Executing: {command}, Response: {response}")

# Приклад використання
machine_controller = MachineController(usb_com)
gcode = generator.get_gcode()
```

```
machine_controller.execute_gcode(gcode)
```

4. Модуль обробки зворотного зв'язку

Модуль, який обробляє зворотний зв'язок від контролера (наприклад, кінцеві вимикачі, сенсори) та відправляє корекції у систему управління.

```
class FeedbackProcessor:
    def __init__(self, communicator):
        self.communicator = communicator

    def check_status(self):
        response = self.communicator.send_command("STATUS")
        print(f"Machine status: {response}")
        return response

# Приклад використання
feedback_processor = FeedbackProcessor(usb_com)
status = feedback_processor.check_status()
```

5. Інтерфейс користувача для взаємодії

Інтерфейс може бути реалізований на основі бібліотеки tkinter або через веб-інтерфейс. Він дозволить користувачу вводити параметри та запускати процес управління верстатом. Я використовував зв'язок термінальної програми з окремо розробленою інтересною частиною.

```
import tkinter as tk

class CNCInterface:
    def __init__(self, root, machine_controller):
        self.machine_controller = machine_controller
        self.setup_ui(root)

    def setup_ui(self, root):
        root.title("CNC Machine Controller")

        self.label_x = tk.Label(root, text="X:")
        self.label_x.grid(row=0, column=0)
        self.entry_x = tk.Entry(root)
        self.entry_x.grid(row=0, column=1)

        self.label_y = tk.Label(root, text="Y:")
        self.label_y.grid(row=1, column=0)
        self.entry_y = tk.Entry(root)
```

```
        self.entry_y.grid(row=1, column=1)

        self.button_move = tk.Button(root, text="Move", command=self.move_machine)
        self.button_move.grid(row=2, column=0, columnspan=2)

    def move_machine(self):
        x = self.entry_x.get()
        y = self.entry_y.get()
        gcode = f"G01 X{x} Y{y} F300"
        self.machine_controller.execute_gcode(gcode)

root = tk.Tk()
app = CNCInterface(root, machine_controller)
root.mainloop()
```

Розрахунки та проектні рішення

1. Розрахунок швидкості передачі даних. Для контролера AT90USB647-AU можна використовувати швидкість передачі даних 115200 біт/сек. Це забезпечує достатню швидкість для передачі команд ЧПК і зворотного зв'язку від контролера.

Обрана формула розрахунку швидкості передачі даних наступна

Швидкість (символів/сек) = baudrate / (1 стартовий біт + 8 біт даних + 1 стоп-біт) = 115200 / 10 = 11520 символів/сек

2. Розрахунок продуктивності крокових двигунів. Середня частота кроків для крокових двигунів може бути до 1000 кроків/сек. Це дозволяє точно керувати переміщенням осей з високою точністю (до 0.1 мм в залежності від механіки).

При частоті 1000 кроків/сек і кроковій точності 0.1 мм на крок, рух на 10 мм вимагатиме 100 кроків, що займе 0.1 сек.

3. Розрахунок обробки зворотного зв'язку При використанні датчиків кінцевих вимикачів, система повинна реагувати на сигнали з мінімальним часом затримки. Середній час обробки сигналів повинен складати не більше 1-2 мс для забезпечення точності роботи.

Запропонована система програмування верстатів ЧПК з використанням контролера AT90USB647-AU дозволяє автоматизувати роботу з верстатами і забезпечує просту інтеграцію через USB. Система містить модулі для генерації

G-коду, передачі команд на контролер і обробки зворотного зв'язку, а також підтримує інтерактивний інтерфейс для користувача. Розрахунки підтверджують ефективність вибраних параметрів системи для роботи з верстатами.

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В алгоритмі використовуються 8x16 S- блоки, засновані на бент-функції, операції XOR і модулярной арифметиці (медулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт

Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 – безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256 біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст $m_1 ... m_{64}$, ключ $K = k_1 ... k_{128}$.

ВИХІД: зашифрований текст $c_1 ... C_{64}$.

1. (розгорнення ключа) становить 16 пару підключів $\{Km_i, Kr_i\}$ отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

2. (L_0, R_0) <- $(m_1 .. m_{64})$. (Розділяє текст на ліву й праву 32-бітні половини $L_0 = m_1 ... m_{32}$ і $R_0 = m_{33} ... m_{64}$).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: $L_i = R_{i-1}$; $R_i = L_{i-1}$ ^ $F(R_{i-1},Km_i,Kr_i)$, де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. $c_1 ... c_{64}$ <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 32-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і "-" додавання й вирахування по модулю 2 ** 32, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ \ S2[I_b]) – (S3[I_c]))+ S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] – S2[I_b]) + (S3[I_c]))\ \ S4[I_d]$
Раунди	$I = ((Km_i – R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ \ (S3[I_c]))- S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни. Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування / дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K32.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K32 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1, ..., Km_{16} 32-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 32-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання магістерської дипломної роботи. Процес розрахунків здійснюється в консольному додатку з передачею отриманих результатів у користувацький інтерфейс. Розроблене програмне забезпечення програматора верстатів ЧПК з використанням контролера AT90USB647-AU складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Контролер AT90USB647-AU**;** Налаштування; Дані; Довідка.

– Блоку виведення команд G коду, Блоку складання кроків позиції.

– Блоку візуального представлення позиції, навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

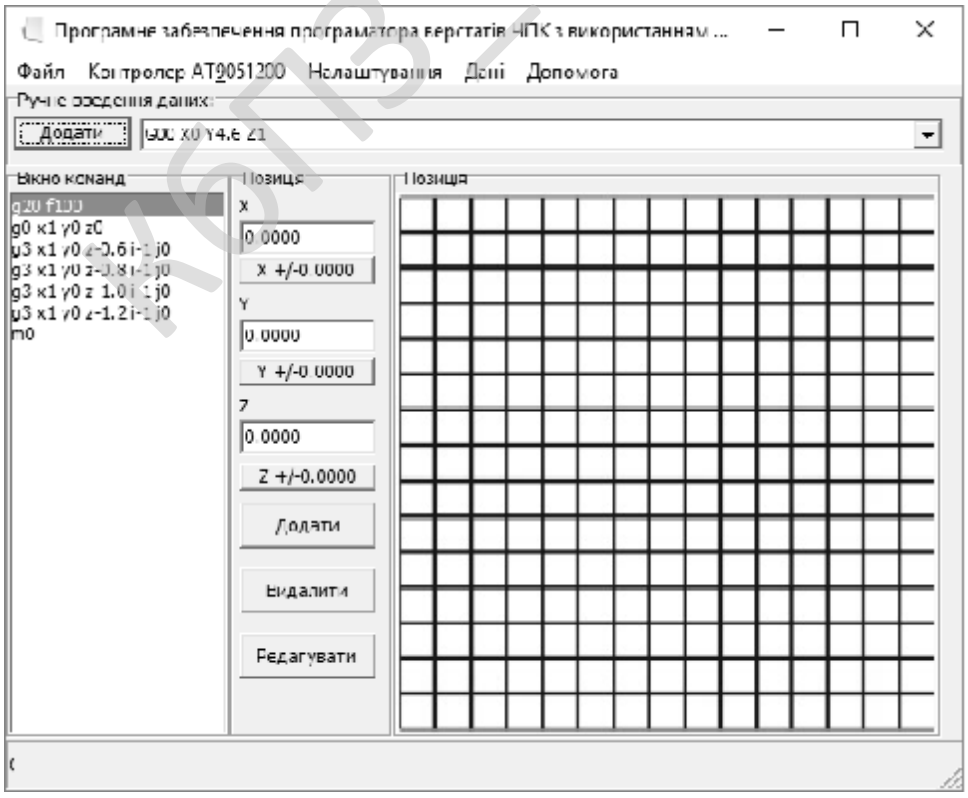

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

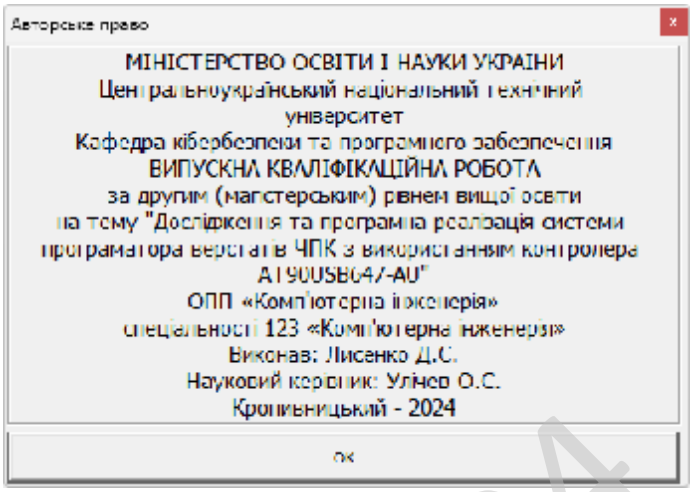

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки. Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Метою розробки є дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Об'єктом дослідження є процес програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Предметом дослідження є методи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Методи дослідження базуються на методах цифрової схемотехніки, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

– Розроблено вітчизняний продукт програматора верстатів ЧПК з використанням контролера AT90USB647-AU, який має більш широкі можливості, на відміну від існуючих аналогів.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи програматора верстатів з числовим програмним управлінням (ЧПК) на базі контролера AT90USB647-AU можуть бути цікавими для різних категорій фахівців та організацій (рисунок 7.1).

Інженери-електроніки: можуть використовувати результати для створення нових пристроїв управління верстатами або вдосконалення існуючих систем.

Розробники програмного забезпечення: зацікавлені в оптимізації програмного забезпечення для ЧПК-верстатів або інтеграції нових функцій.

Промислові підприємства: виробники, які використовують ЧПК-верстати, можуть впроваджувати нові технології програмування для покращення продуктивності та якості виробництва.

Малий і середній бізнес: компанії, що займаються малим серійним виробництвом, можуть виграти від впровадження ефективних і недорогих рішень для управління верстатами.

Дослідницькі лабораторії: можуть проводити додаткові дослідження на основі отриманих результатів для розвитку нових рішень.

Компанії-виробники електронних компонентів: можуть використовувати результати для розвитку нових продуктів на основі контролера AT90USB647-AU або аналогічних рішень.

Консультанти в галузі автоматизації: можуть використовувати результати дослідження для надання рекомендацій клієнтам щодо впровадження технологій ЧПК.

Рисунок 7.1 – Цільова аудиторія

Таким чином, результати дослідження та програмної реалізації системи програматора верстатів ЧПК на базі контролера AT90USB647-AU можуть бути корисними для широкого кола фахівців, від інженерів і підприємств до освітніх установ та ентузіастів. Це може сприяти вдосконаленню технологій, підвищенню ефективності виробництва та розвитку нових інновацій у цій галузі.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Оцінка привабливості програмної реалізації системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU може бути виконана за допомогою методів експертних оцінок. Один з можливих підходів – це метод експертних оцінок на основі багатокритеріального аналізу.

Для оцінки привабливості проекту можна визначити наступні критерії:

– технічна реалізованість: наскільки можлива реалізація проекту з технічної точки зору;

– економічна ефективність: очікувана рентабельність інвестицій (ROI);

– інноваційність: рівень новизни рішення в порівнянні з існуючими;

– конкурентоспроможність: здатність проекту конкурувати з аналогічними рішеннями на ринку;

– ризики: імовірність виникнення ризиків і їх вплив на проект.

Сформуйте групу експертів, яка буде включати: інженерів, що мають досвід роботи з контролерами та ЧПК, фахівців з управління проектами, економістів або фінансових аналітиків.

Кожен експерт оцінює кожен критерій за шкалою, наприклад, від 1 до 5, де: 1 – дуже низька оцінка, 2 – низька оцінка, 3 – середня оцінка, 4 – висока оцінка, 5 – дуже висока оцінка. Збираємо результати оцінок експертів в таблицю 7.1.

Таблиця 7.1 – Зведені результати експертних оцінок

Критерії	**Експерт 1**	**Експерт 2**	**Експерт 3**	**Середній бал**
Технічна реалізованість	4	5	4	4.33
Економічна ефективність	3	4	4	3.67
Інноваційність	5	4	4	4.33
Конкурентоспроможність	4	4	5	4.33
Ризики	2	3	2	2.33

Визначте загальну оцінку для проекту, враховуючи ваги критеріїв, які присвоєно наступні: технічна реалізованість – 30%, економічна ефективність – 25%, інноваційність – 20%, конкурентоспроможність – 15%, ризики – 10%.

Загальна оцінка = (4.33×0.3)+(3.67×0.25)+(4.33×0.2)+(4.33×0.15)+(2.33×0.1).

Отримана загальна оцінка дозволить вам зробити висновок про привабливість проекту. Якщо оцінка перевищує певний поріг (3.5), проект може вважатися привабливим для реалізації. Цей процес експертних оцінок дозволяє не лише систематизувати думки фахівців, а й забезпечити об’єктивність у прийнятті рішень щодо доцільності реалізації проекту програмної реалізації системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи програматора верстатів ЧПК на базі контролера AT90USB647-AU можна використовувати кілька методів, в залежності від специфіки проекту, наявних даних і цілей оцінки.

Метод аналізу витрат полягає в розрахунку всіх витрат, пов’язаних із розробкою програмного забезпечення. Включає: прямі витрати (заробітна плата розробників, витрати на обладнання та програмне забезпечення), непрямі витрати (витрати на адміністративні витрати, оренду, електроенергію тощо.

Метод оцінки за аналогією базується на оцінці вартості проекту шляхом порівняння з подібними проектами, які вже були реалізовані. Важливо, щоб аналоги були максимально схожі за характеристиками і масштабами. Це дозволяє визначити орієнтовну вартість на основі історичних даних.

Метод оцінки на основі функцій ґрунтується на розрахунку вартості залежно від кількості функціональних одиниць (функцій, модулів) програми. Часто використовується в методах FPA (Function Point Analysis), де оцінюється обсяг програмного забезпечення на основі його функцій.

У методі експертних оцінок залучаються експерти, які надають свої оцінки вартості проекту на основі власного досвіду та знань. Експертні оцінки можуть бути кількісними (оцінки у фінансових одиницях) або якісними (наприклад, висока/середня/низька вартість).

Метод вартості життєвого циклу всі етапи життєвого циклу продукту, включаючи розробку, тестування, впровадження, експлуатацію та підтримку. Це дозволяє отримати повну картину витрат на проект у довгостроковій перспективі.

Для програмної реалізації системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU доцільно комбінувати декілька методів. Комбінувати метод аналізу витрат з методом оцінки за аналогією: це дозволить отримати більш точну оцінку вартості на основі попередніх проектів, враховуючи специфіку нового проекту.

Використовувати метод функцій для точнішої оцінки: якщо проект має чітко визначені функції, це дозволить розрахувати вартість більш детально.

Залучати експертів: оцінка вартості за допомогою експертних оцінок може надати цінну інформацію, особливо на етапах, коли відсутні точні дані.

Цей підхід дозволить отримати об'єктивну і всебічну оцінку вартості проекту, що в свою чергу допоможе у прийнятті рішень щодо його реалізації.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність впровадження системи програматора верстатів з числовим програмним управлінням (ЧПК) на базі контролера AT90USB647-AU може бути проаналізована за кількома параметрами, такими як зниження витрат, підвищення продуктивності та рентабельність інвестицій (ROI).

Таблиця 7.2 – Економічна ефективність від реалізації проєкту для клієнта

Збільшення продуктивності	Економічні вигоди
Підвищення продуктивності: +20% (за рахунок автоматизації процесів) Збільшення кількості оброблених деталей на місяць: з 1000 до 1200 деталей. Дохід від продажу	Збільшення доходу щомісяця Рентабельність інвестицій (ROI) – 25,6%

Впровадження системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU може принести значну економічну ефективність через зниження витрат, підвищення продуктивності та збільшення доходів. На прикладі видно, що через кілька місяців інвестиції починають приносити прибуток, і проект стає рентабельним. Цей підрахунок може служити аргументом для прийняття рішення про впровадження подібних технологій на виробництві.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Алгоритм просування проєкту програмної реалізації системи програматора верстатів ЧПК на базі контролера AT90USB647-AU, який включає кілька етапів, від аналізу ринку до впровадження та моніторингу запропоновано на рисунку 7.2.

1. Аналіз ринку	Дослідження конкурентів: проаналізуйте існуючі рішення на ринку, їхні сильні та слабкі сторони.
	Визначення цільової аудиторії: визначте потенційних користувачів системи (виробництва, що використовують ЧПК, навчальні заклади тощо).
	Аналіз потреб: з'ясуйте специфічні потреби та вимоги цільової аудиторії до програмного забезпечення.
2. Розробка стратегії	Позиціювання продукту: визначте, як ваша система буде позиціонуватися на ринку (цінова стратегія, унікальні особливості).
	Формулювання цілей: визначте короткострокові та довгострокові цілі просування.
	Бюджет: складіть бюджет на маркетинг та просування.
3. Розробка та тестування	Розробка програмного забезпечення: створіть прототип програми на базі контролера AT90USB647-AU.
	Тестування: проведіть тестування програмного забезпечення на різних верстатах ЧПК.
	Збір зворотного зв'язку: залучіть користувачів для тестування та збору відгуків про програму.
4. Маркетинг та просування	Створення веб-сайту: розробіть сайт з інформацією про продукт, можливостями та перевагами.
	SEO-оптимізація: оптимізуйте сайт для пошукових систем, щоб підвищити видимість.
	Соціальні мережі: створіть профілі в соціальних мережах для залучення уваги до продукту.
	Реклама: використовуйте контекстну рекламу, банери та рекламні кампанії для залучення клієнтів.
5. Взаємодія з клієнтами	Презентації та демонстрації: організуйте вебінари, семінари або демонстрації продукту для потенційних клієнтів.
	Виставки та конференції: беріть участь у виставках, конференціях та галузевих заходах.
	Співпраця з дистриб'юторами: знайдіть партнерів і дистриб'юторів, які можуть допомогти в просуванні вашого продукту.
6. Впровадження	Пілотне впровадження: проведення пілотного впровадження у вибраних підприємствах для тестування.
	Навчання користувачів: розробіть навчальні матеріали та проведення тренінгів для кінцевих користувачів.
7. Моніторинг та підтримка	Збір зворотного зв'язку: регулярно збирайте відгуки користувачів для вдосконалення програмного забезпечення.
	Оновлення продукту: впроваджуйте оновлення та нові функції на основі зворотного зв'язку.
	Технічна підтримка: забезпечте наявність технічної підтримки для користувачів.
8. Оцінка ефективності	Аналіз результатів: оцініть досягнуті результати порівняно з поставленими цілями.
	Коригування стратегії: на основі отриманих результатів коригуйте стратегію просування та вдосконалення продукту.

Рисунок 7.2 – Алгоритм просування проєкту

Цей алгоритм просування проєкту дозволяє систематично підійти до реалізації програми для програматора верстатів ЧПК. Важливо адаптувати його до конкретних умов і вимог ринку, а також активно реагувати на зміни в потребах клієнтів і конкурентному середовищі.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту програмного забезпечення для програматора верстатів ЧПК з використанням контролера AT90USB647-AU варто розглянути кілька стратегій, спрямованих на досягнення максимальної ефективності та охоплення цільової аудиторії.

Стратегії оптимізації каналів збуту:

1. Прямі продажі підприємствам (прямі комунікації з виробничими компаніями: звернення до компаній, що використовують ЧПК-верстати, з демонстрацією продукту та його переваг; персоналізовані пропозиції: створення персональних пакетів або знижок для клієнтів, які готові впровадити рішення для кількох ліній виробництва; пілотні проєкти: пропонування безкоштовного тестування системи на невеликій партії обладнання, що допоможе побудувати довіру з боку підприємств).

2. Співпраця з дистриб’юторами і реселерами (пошук партнерів-дистриб'юторів: знайдіть компанії, які вже реалізують продукцію для ЧПК-верстатів і можуть включити вашу систему до свого асортименту; програма партнерської співпраці: розробка програм для дистриб'юторів, що передбачає знижки, бонуси або технічну підтримку; підтримка технічних консультантів: забезпечте технічну підготовку дистриб'юторів для ефективного просування продукту та консультації з клієнтами).

3. Онлайн-канали продажу (власний веб-сайт: створення сайту з докладною інформацією про продукт, його функціональність, інструкціями та підтримкою; маркетплейси та інтернет-магазини: розміщення продукту на

спеціалізованих платформах для промислового обладнання; соціальні мережі та професійні платформи (LinkedIn, Xing): регулярне публікування оглядів продукту, кейсів впровадження, а також відгуків клієнтів для створення бренду і довіри).

4. Консалтингові компанії (взаємодія з компаніями, які впроваджують ЧПК-рішення: залучення консалтингових компаній, що спеціалізуються на впровадженні новітніх технологій на виробництві, для рекомендації продукту клієнтам, програма для консультантів: стимулювання консультантів до просування продукту через спеціальні бонуси або комісії за залучення нових клієнтів).

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовими факторами успіху проєкту програмної реалізації системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU є такі (рис. 7.3):

Рисунок 7.3 – Ключові фактори успіху проєкту

Ці фактори є основою для досягнення успіху проєкту, оскільки вони забезпечують надійність, економічність, задоволеність клієнтів і конкурентну перевагу на ринку систем для ЧПК.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Зі становленням та розвитком інформаційного суспільства спостерігається масове впровадження комп’ютерних технологій в усіх сферах життя і діяльності людини. Застосування персональних комп’ютерів і ЕОМ дозволило значно підвищити продуктивність праці, змінити характер і зміст праці.

Впровадження комп'ютерних технологій принципово змінило характер праці різних категорій фахівців. Програмісти у процесі роботи отримують негативний вплив на органи зору, а також мають значну розумову напругу і нервово-емоційне навантаження. Руки (м’язи рук та суглоби пальців) при роботі з клавіатурою мають теж істотне навантаження. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій спеціалісти відносять високочастотні електромагнітні коливання роботи апаратної частини ЕОМ та виділення шкідливих газів [1, 2].

Ці шкідливі фактори можуть привести до професійних захворювань.

До недоліків умов праці користувачів комп’ютерної техніки можна віднести:

– недостатню площу і обсяг виробничого приміщення;

– недотримання вимог, мікроклімату на робочих місцях;

– низький рівень освітленості у приміщеннях і на робочих поверхнях апаратури;

– підвищений рівень низькочастотних магнітних полів від моніторів;

– порушення вимог організації робочих місць;

– недотримання вимог до режимам праці та відпочинку;

– надмірне виробничу навантаження працівників;

– відсутність навичок зниження впливу психоемоційного напруги.

Відповідно до ст. 14 Закону «Про охорони праці» [3] на роботодавця покладено обов'язок забезпечити: безпеку працівників при експлуатації устаткування; застосування коштів індивідуальної захисту працівників; відповідні вимоги охорони праці, умови праці в кожному робоче місце; дотримання режиму праці та відпочинку працівників; навчання безпечним методам і прийомам виконання; інструктаж з охорони праці; організацію контролю над станом умов праці в робочих місць; проведення атестації робочих місць в умовах праці.

Максимально зменшити кількість шкідливих впливів на людину при високій продуктивності праці, створити комфортні умови для роботи людей – ось одна з головних задач охорони праці [5].

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальні машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98 [2].

Шкідливими факторами при роботі з персональним комп'ютером є неонізуюче випромінювання промислової частоти, збільшене нервово-емоційне навантаження на оператора, збільшення навантаження на органи зору та дрібні стереостатичні рухи кінцівок.

Ці фактори можуть викликати у працівника певні розлади здоров'я, зокрема підвищення артеріального тиску, кон'юктивіти, тендовагініти ті інші захворювання.

Комп'ютер, як і будь-який електричний прилад, особливо при його неправильному підключенні, може бути джерелом ураження оператора

електричним струмом. Саме тому всі працівники, які працюють з персональним комп'ютером, повинні мати першу(або другу) групу допуску з електробезпеки.

Через наявність зазначених факторів працівники, які працюють з персональними комп'ютерами, підлягають попередньому та періодичному медичному огляду згідно з пунктом 6.2.3 додатку 4 до наказу Міністерства охорони здоров'я України "Про затвердження Порядку проведення медичних оглядів працівників певних категорій" від 21 травня 2007 року №246 [8].

8.3 Аналіз умов праці

Приміщення розташовано на третьому поверсі п'ятиповерхового будинку. У приміщенні розташовано 3 робочих місць з комп'ютерами (далі ПК). Відповідно до норм «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [4] площа, що відводиться для робочого місця з комп'ютером повинна бути не менше 6 м^2, об'єм не менше 20 м^3. Розміри даного приміщень складають: довжина – 6 м, ширина – 4,5 м, висота – 3,5 м, тобто загальна фактична площа складає 27 м^2. Необхідна площа на 3 робочих місця із установленими ПК складає 18 м^2, що не перевищує фактичну. Обсяг кабінету на одного працюючого складає 31,5м^3, отже відповідає нормі ДСанПіН 3.3.2-007-98 – не менше 20 м^3.

При роботі з ПК людина може піддатися впливу шкідливих та небезпечних факторів. Під шкідливими виробничими факторами розуміють фактори, тривалий вплив яких викликає розвиток професійних захворювань. Небезпечні виробничі фактори – вплив яких на працюючого викликає травму, тобто пошкодження організму. Шкідливі і небезпечні чинники, з якими стикається бібліограф при роботі з ПК, приведені в таблиці 8.1.

По категорії вибухо– і пожежонебезпеки, згідно дане приміщення відноситься до категорії В – пожежонебезпечне, тому що присутні тверді матеріали, що горять, такі як дерев'яні столи, папір і інше. Виходячи з категорії

пожежонебезпеки і поверховості будинку, ступінь вогнестійкості будівлі II. Згідно з ДБН В 1.1-7-2016 «Пожежна безпека об'єктів будівництва» [5] ЕОМ повинні розташовуватись в будівлі не менше ніж II ступню вогнестійкості.

Таблиця 8.1 – Перелік шкідливих та небезпечних виробничих факторів

Найменування факторів	Можливі джерела їх виникнення	Характер дії
Небезпека ураження електричним струмом	Мережа живлення	Небезпечний
Пожежонебезпечність приміщень	Наявність матеріалів, що згорають і джерел запалення (електроапаратура)	Небезпечний та шкідливий
Іонізація повітря	Статична електрика випромінювання	Шкідливий
Підвищений рівень шуму	Шум створюється перетворювачем напруги ЕОМ, її технічною периферією, а також людьми, що працюють в приміщенні	Шкідливий
Несприятлива освітленість	Недостатнє штучне і природне освітлення	Шкідливий
Незадовільні параметри мікроклімату	Незадовільний стан системи опалення і вентиляції	Шкідливий
Психофізіологічні напруження	Монотонність праці, перенапруженість зорових аналізаторів, розумова напруженість, незручність і статичність пози	Шкідливий

По ступені небезпеки поразки людей електричним струмом відділ, згідно, класифікується як приміщення з підвищеною небезпекою, тому що не виключена можливість одночасного дотику людини до маючих з'єднання з землею конструкціям будинку, з одного боку, і до металевих корпусів електроустаткування, що можуть виявити під напругою – з іншого.

Для забезпечення вищевказаних оптимальних метеорологічних умов у помешканні передбачена система опалення (загальне парове) в холодному періоді, та вентиляція і кондиціювання в теплий період року, згідно ДБН2.5–67–2013 «Опалення, вентиляція та кондиціювання» [6]. При виконанні замірів параметрів мікроклімату, значення їх відповідали оптимальним та допустимим параметрам відповідно до ДСанПіН3.3.2.007 – 98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно – обчислювальних машин».

Припустимий рівень іонізації повітря помешкання відповідно до СН 21.52-80 повинен складати 1500 – 3000 один./м^3.

Нормування освітлення здійснюється відповідно до ДБН В.2.5 – 28 – 2006 «Природне та штучне освітлення». [7]

Відділ забезпечений комбінованим освітленням. В темний час доби передбачається загальне і/або місцеве рівномірне штучне, а в світлий – бокове одностороннє природне освітлення два віконних прорізи.

Одним з найбільш поширеніших чинників зовнішнього середовища, який несприятливо впливає на людину, є шум. Вплив шуму на організм людини залежить від рівня звукового тиску, частотних характеристик, тривалості дії, а також індивідуальних особливостей людини.

При тривалій дії шуму у оператора ЕОМ виявляються симптоми утомленості, нервового збудження, що сприяють погіршенню працездатності і допущенні помилок при роботі. Для уникнення шкідливої дії шуму на організм працюючого, необхідне дотримання нормованих параметрів, які не повинні перевищувати допустимих величин. При роботі на комп'ютері рівень шуму не

повинен перевищувати 50 дБА. Приміщення розташоване вікнами у двір і знаходиться далеко від проїжджої частини вулиці. Основними джерелами шуму в приміщенні є устаткування і люди. Розглянута кімната не призначена для прийому відвідувачів і тому в ній не спостерігається великого скупчення людей. Тому основним джерелом шуму є комп'ютерна техніка.

Джерелами шуму при роботі ЕОМ є механічні частини принтера, що рухаються, і вентилятори(Lпк = 35 дБА, , Lprn = 48 дБА.) При роботі вентиляційної системи, що забезпечує оптимальний температурний режим електронних блоків ЕОМ і вмонтована в задню панель, створюється аеродинамічний шум. Шум, створюваний працюючим комп'ютером, може бути охарактеризований як широко смуговий постійний з аперіодичним посиленням при роботі принтера. Час роботи ПЕОМ – 6 - 8 год. за добу; принтери працюють не більш 1,5-2 год. за добу.

При наявності великої кількості джерел шуму еквівалентне значення шуму LЭКВ, дБА розраховують по наступній формулі:

$$L_{ЭКВ} = 10 \cdot \lg\left(\frac{1}{T}\sum_{i=1}^{n}\left(t_i \cdot 10^{0.1 \cdot L_i}\right)\right) \quad (8.1)$$

де:

L_i – рівень шуму i-го джерела (пристрою),

ti – час роботи i-го джерела (пристрою),

T – загальний час роботи,

n – кількість джерел шуму даного типу;

Для даного приміщення необхідні змінні складають:

Загальний час роботи – робітник день, тобто T=8 годин.

Для фонового шуму (вентиляторів):

L1 = 35 дБА, T1 = 8 годин, n1=15 (5x3);

Для лазерного принтера Lexmark Jet:

L2 = 48дБА, T2 = 2 години, n2=1, для сканера L3 = 46дБА, T3 = 2 години.

Підставляємо отримані величини у формулу (8.1):

$$L_{экв} = 10 \cdot \lg\left(\frac{1}{8} \cdot \left(15 \cdot 8 \cdot 10^{0.1 \cdot 35} + 1 \cdot 2 \cdot 10^{0.1 \cdot 48} + 1 \cdot 2 \cdot 10^{0.1 \cdot 46}\right)\right) = 46{,}3 \text{ дБА}$$

Таким чином, еквівалентний рівень шуму в приміщенні за робітник день $L_{экв}$ = 46,3 дБА , тобто не перевищує норму 50 дБА.

8.4 Техніка безпеки та протипожежна профілактика

Відповідно ДБН В 1.1-7-2016 «Пожежна безпека об’єктів будівництва» будинок можна віднести до II групи по ступені вогнестійкості й до категорії Д по ступені пожежонебезпеки.

Від розподільного щита по праву й ліву сторони встановлені кондиціонери, зовнішня електропроводка, поміщена в ізольований кабель. Висота проводки становить 2,2 м від рівня підлоги, її кріплення здійснюється за допомогою металевих власників. Біля кожного стола організований розподільний щит, розташований на текстолітовій пластинці, закріпленої на стіні на рівні 1м від підлоги. Усього до складу входять п'ять розеток і дві клеми заземлення. Всі обчислювальні машини з'єднані із клемами заземлення. Чотири з п'яти розеток забезпечують подачу напруги 220 V, а одна, забезпечує подачу напруги в 36 V. Про це є відповідні написи на кожному розподільному щиті.

Робота обслуговуючого персоналу полягає в інсталяції необхідного програмного забезпечення й наступному його використанні в діалоговому режимі роботи з ЕОМ. Іноді може виникати необхідність написання допоміжних програм для поліпшення роботи вузла або для зниження витрат. З погляду забезпечення умов праці й вимог техніки безпеки для роботи програміста необхідно наступне: достатнє висвітлення екрана дисплея й робочого місця; повна технічна справність устаткування, його електробезпечність; достатня пожежобезпечність приміщення; оптимальний мікроклімат, що сприяє продуктивній роботі; відповідність робочого місця вимогам ергономіки. До небезпечних і шкідливих факторів, дії яких піддається програміст, можна віднести: можливість поразки

електричним струмом, при електроні справності встаткування, порушенні заземлення або техніки безпеки; робота в мікрокліматі з неприпустимими параметрами; робота при недостатній освітленості екрана дисплея й робочого місця.

Відповідно НПАОП 40.1-1.21-98 “Правил безпечної експлуатації електроустановок споживачів” [8], приміщення можна віднести до приміщень без підвищеної небезпеки, оскільки це приміщення, сухе, з нормальною температурою й ізолюючими підлогами, що не має заземлених металоконструкцій.

Персональні ЕОМ можна віднести до першого класу електротехнічних виробів по способі захисту людини від поразки електричним струмом, оскільки їхні корпуси зроблені з ізолюючої пластмаси й кожен пристрій має заземлення. Відповідно правилам пристрою електроустановок ЕОМ можна віднести до електроустановок з робочою напругою до 1000 В.

Однієї з достовірних причин пожежі в приміщенні з обчислювальною технікою може бути коротке замикання, що спричиняє спалах електропроводки. Для його попередження вся обчислювальна техніка, а також інші електричні пристрої повинні бути обладнані плавкими запобіжниками, а на вході електромережі повинен бути передбачений автомат захисту. Не слід користуватися електричними подовжувачами й трійниками, що не мають сертифікатів відповідності вимогам безпеки.

Необхідно передбачити наявність у межах досяжності первинних засобів гасіння пожежі (вогнегасників) для локалізації вогню власними засобами до приїзду команди пожежної охорони. Повинен бути розроблений план екстреної евакуації персоналу при виникненні загоряння. Кількість евакуаційних виходів повинне бути не менш двох. Допускається використання одного евакуаційного виходу, якщо відстань найбільш віддаленого робочого місця до цього виходу не перевищує 25 м.

8.5 Розрахункова частина

Запорукою безпечної роботи є виконання вимог електричної безпеки, оскільки все офісне обладнання заживлюється від електричної мережі. Одним з необхідних засобів електричної безпеки є встановлення захисного заземлення.

Початкові дані, необхідні для розрахунку захисного заземлення:

– допустимий опір розповсюдженню струму в землі від заземлювального пристрою $R_{зн}$ = 10 Ом;

– питомий опір грунту в місці встановлення заземлювача $\rho_з$ = 100 Ом/м;

– тип грунту – суглинок;

– тип заземлювача – труба, діаметром d=0.06 м і довжиною l =1.7 м;

– конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [7].

Визначимо розрахунковий опір землі:

$$\rho_{рз} = \phi \cdot \rho_з$$

де ϕ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні вологості ґрунту протягом року; при використанні заземлювача довжиною l = 1.7 м при глибині закладання від вершини h = 0.65 м ϕ = 1.1 для четвертої кліматичної зони.

Схема розташування заземлювачів показана на рисунку 8.1.

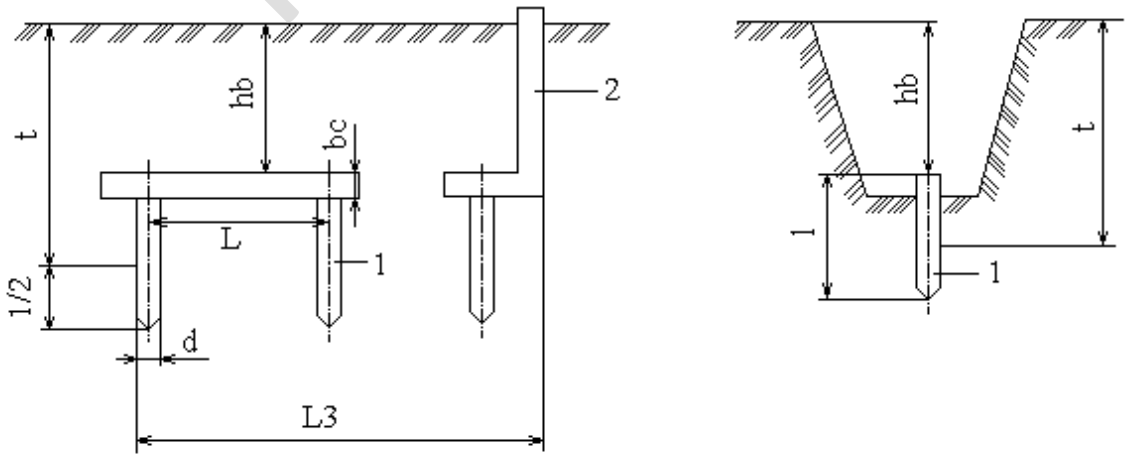

Рисунок 8.1 - Схема розташування заземлювачів

Опір землі:

$$\rho_{рз} = 1{,}1 \cdot 100 = 110 \text{ Ом·м}$$

Опір Rв, розповсюдженню струму в землі від одного вертикального заземлювача:

$$R_B = \frac{\rho_{рз}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0.5\ln\frac{4t + l}{4t - l}\right)$$

де l – довжина заземлювача (l = 1.7 м);

d = 0.06 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.65 + 1.7/2 = 1.5 \text{ м.}$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 1.7}\left(\ln\left(\frac{2 \cdot 1.7}{0.06}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.5 + 1.7}{4 \cdot 1.5 - 1.7}\right)\right) = 45.17 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

$$n' = \frac{R_в}{R_{зн}} = \frac{45.17}{10} = 4.5 \approx 5 \text{ шт.}$$

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 2.4 м.

Таким чином коефіцієнт використання вертикального заземлювача за довідковими даними дорівнює $\eta_В = 0{,}6$.

Кількість вертикальних заземлювачів з урахуванням коефіцієнту використання $\eta_В$ приблизно складає

$$n = \frac{R_в}{R_{зн} \cdot \eta_в} = \frac{45.17}{10 \cdot 0.6} = 7.53 \approx 8 \text{ шт.}$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 2.4 \cdot 8 = 19.2 \text{ м}$$

Опір горизонтального заземлювача $R_Г$, Ом, прокладеного на глибині h = 0.65 м від поверхні землі буде

$$R_г = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 19.2} \cdot \ln \frac{2 \cdot 19.2^2}{0.06 \cdot 0.65} = 10.61 \text{ Ом}$$

де b = 0.04 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_з = \frac{R_в \cdot R_г}{n \cdot R_г \cdot \eta_в + R_в \cdot \eta_г} = \frac{45.17 \cdot 10.61}{6 \cdot 10.61 \cdot 0.6 + 45.17 \cdot 0.34} = 8.33 \text{ Ом}$$

де $\eta_Г$ - коефіцієнт використання горизонтального заземлювача ($\eta_Г = 0.34$).

Маємо 8.33 < 10 Ом (за потужності генераторів та трансформаторів 100 кВт і менше), отже нормативне обмеження $R_з < R_{з.норм}$ виконується.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

– Досліджена система програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

– На основі отриманих результатів досліджень створена програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CAST-128.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лисенко Д.С. Дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

2. І.В.Чихіра, А.Г. Микитишин Конспект лекцій з дисципліни «Програмування систем реального часу» / Укладачі : Чихіра І.В., Микитишин А.Г., – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя , 2016. – 76 с.

3. Jack Ganssle and Michael Barr. 2003. Embedded Systems Dictionary. CMP Books.

4. Технології інтернету речей. Навчальний посібник [Електронний ресурс]: / Б. Ю. Жураковський, І.О. Зенів; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 12,5 Мбайт). – Київ: КПІ ім. Ігоря Сікорського, 2021. – 271 с.

5. Greg Dunko, Joydeep Misra, Josh Robertson, Tom Snyder “A reference guide to the Internet of Things” / 2017 Bridgera LLC, RIoT.

6. Donald Norris “Programming with STM32. Getting started with the Nucleo Board and C/C++” 416 p. 2018.

7. Neil Kolban “Kolban’s book on ESP32”. Texas, USA. 951 p.

8. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

9. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

10. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

11. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

12. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

13. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

14. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

15. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

16. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

17. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In:

Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

18. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

19. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

20. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

21. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

22. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

23. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

24. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications,*

Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

25. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

26. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

27. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

28. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

29. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

30. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

31. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

32. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

33. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

35. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

36. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

37. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

38. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

39. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

40. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

41. О.А. Смірнов, П.С. Усік, «дослідження перспектив використання технологічних рішень в мережах 5g» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

42. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

43. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

44. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

45. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

46. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

47. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

48. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

49. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

50. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

51. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

52. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 07.08.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати маркетингове та економічне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2024 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 97 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Маркетингове та економічне обґрунтування ІТ-проєкту.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 16.12.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Улічев О.С.

Дослідження та програмна реалізація системи програматора верстатів ЧПК з використанням контролера AT90USB647-AU

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 19

Літера: РП

Кропивницький – 2024 року

Основна програма

```
import serial
import time

# Функція ініціалізації з'єднання з контролером AT90USB647-AU
def initialize_connection(port, baud_rate):
    ser = serial.Serial(port, baud_rate)
    time.sleep(2)
    return ser

# Функція відправки команди на контролер
def send_command(ser, command):
    ser.write(command.encode())
    time.sleep(0.1)

# Функція читання відповіді від контролера
def read_response(ser):
    if ser.in_waiting > 0:
        response = ser.readline().decode('utf-8').strip()
        return response
    return None

# Функція закриття з'єднання з контролером
def close_connection(ser):
    ser.close()

# Функція переміщення осі X на задану відстань
def move_x_axis(ser, distance):
    command = f"G01 X{distance}"
    send_command(ser, command)
    response = read_response(ser)
    return response

# Функція переміщення осі Y на задану відстань
def move_y_axis(ser, distance):
    command = f"G01 Y{distance}"
    send_command(ser, command)
    response = read_response(ser)
    return response

# Функція переміщення осі Z на задану відстань
def move_z_axis(ser, distance):
    command = f"G01 Z{distance}"
    send_command(ser, command)
    response = read_response(ser)
    return response

# Функція запуску програми ЧПК
def start_cnc_program(ser):
    command = "M03"# Запуск шпинделя
    send_command(ser, command)
    return read_response(ser)

# Функція зупинки програми ЧПК
def stop_cnc_program(ser):
    command = "M05"# Зупинка шпинделя
    send_command(ser, command)
    return read_response(ser)

# Функція встановлення швидкості подачі
def set_feed_rate(ser, rate):
    command = f"F{rate}"
    send_command(ser, command)
    return read_response(ser)

# Функція встановлення швидкості обертання шпинделя
```

```
def set_spindle_speed(ser, speed):
    command = f"S{speed}"
    send_command(ser, command)
    return read_response(ser)

# Функція зворотнього калібрування інструменту по осі Z
def calibrate_tool_z(ser):
    command = "G28 Z0"
# Калібрування осі Z
    send_command(ser, command)
    return read_response(ser)

# Основна програма для керування верстатом ЧПК
def main():
    port = '/dev/ttyUSB0'
# Порт підключення контролера
    baud_rate = 115200
# Швидкість передачі даних
    ser = initialize_connection(port, baud_rate)

    if ser.is_open:
        print("Підключення встановлено успішно")

 # Приклад роботи програми ЧПК
        set_spindle_speed(ser, 1500)
# Встановлення швидкості обертання шпинделя
        set_feed_rate(ser, 300)
# Встановлення швидкості подачі
        move_x_axis(ser, 100)
# Переміщення по осі X
        move_y_axis(ser, 50)
# Переміщення по осі Y
        move_z_axis(ser, -10)
# Переміщення по осі Z
        start_cnc_program(ser)
# Запуск програми

        time.sleep(5)# Очікування завершення руху

        stop_cnc_program(ser)# Зупинка програми
        calibrate_tool_z(ser)# Калібрування інструменту

        close_connection(ser)# Закриття з'єднання

    else:
        print("Не вдалося підключитися до контролера")

# Виклик головної функції
if __name__ == "__main__":
    main()

# Функція ініціалізації нульової точки верстата
def initialize_zero_point(ser):
    command = "G92 X0 Y0 Z0"# Встановлення нульової точки
    send_command(ser, command)
    return read_response(ser)

# Функція різки по прямій лінії
def cut_line(ser, x_distance, y_distance):
    command = f"G01 X{x_distance} Y{y_distance} F300"
# Переміщення по прямій лінії
    send_command(ser, command)
    return read_response(ser)

# Функція переміщення до початкової точки
def move_to_start(ser):
    command = "G00 X0 Y0"
# Швидке переміщення до нульової точки
    send_command(ser, command)
```

```
    return read_response(ser)

# Функція активації охолодження
def activate_cooling(ser):
    command = "M08"
# Включення охолодження
    send_command(ser, command)
    return read_response(ser)

# Функція деактивації охолодження
def deactivate_cooling(ser):
    command = "M09"# Вимкнення охолодження
    send_command(ser, command)
    return read_response(ser)

# Функція зупинки програми при помилці
def emergency_stop(ser):
    command = "M112"# Екстрена зупинка
    send_command(ser, command)
    return read_response(ser)

# Функція встановлення кутового руху
def set_angular_motion(ser, angle):
    command = f"G02 I{angle} J{angle}"# Кутовий рух
    send_command(ser, command)
    return read_response(ser)

# Функція переміщення до певної координати
def move_to_coordinate(ser, x, y, z):
    command = f"G01 X{x} Y{y} Z{z} F200"
# Переміщення до координати
    send_command(ser, command)
    return read_response(ser)

# Функція перезавантаження контролера
def reset_controller(ser):
    command = "M999"# Перезавантаження
    send_command(ser, command)
    return read_response(ser)

# Функція увімкнення лазера
def turn_on_laser(ser):
    command = "M106"# Увімкнення лазера
    send_command(ser, command)
    return read_response(ser)

# Функція вимкнення лазера
def turn_off_laser(ser):
    command = "M107"# Вимкнення лазера
    send_command(ser, command)
    return read_response(ser)

# Функція перевірки стану контролера
def check_status(ser):
    command = "M114"
# Запит поточного стану
    send_command(ser, command)
    return read_response(ser)
```

Файл multi_axis_control.py

```
# file: multi_axis_control.py

import time

# Функція переміщення осі A на задану відстань
def move_a_axis(ser, distance):
    command = f"G01 A{distance}"
    send_command(ser, command)
    response = read_response(ser)
    return response

# Функція переміщення осі B на задану відстань
def move_b_axis(ser, distance):
    command = f"G01 B{distance}"
    send_command(ser, command)
    response = read_response(ser)
    return response

# Функція переміщення осі C на задану відстань
def move_c_axis(ser, distance):
    command = f"G01 C{distance}"
    send_command(ser, command)
    response = read_response(ser)
    return response

# Функція переміщення всіх осей одночасно
def move_all_axes(ser, x, y, z, a, b, c):
    command = f"G01 X{x} Y{y} Z{z} A{a} B{b} C{c}"
    send_command(ser, command)
    response = read_response(ser)
    return response
```

Файл controller_support.py

```
# file: controller_support.py

# Функція вибору типу контролера та ініціалізації відповідного зв'язку
def initialize_controller(controller_type, port, baud_rate):
    if controller_type == "AT90USB647-AU":
        ser = initialize_connection(port, baud_rate)
    elif controller_type == "GRBL":
        ser = initialize_grbl_connection(port, baud_rate)
    elif controller_type == "Smoothieboard":
        ser = initialize_smoothieboard_connection(port, baud_rate)
    else:
        raise ValueError("Невідомий тип контролера")
    return ser

# Функція ініціалізації з'єднання для контролера GRBL
def initialize_grbl_connection(port, baud_rate):
    ser = serial.Serial(port, baud_rate)
    time.sleep(2)
    send_command(ser, "\r\n\r\n")
    return ser

# Функція ініціалізації з'єднання для Smoothieboard
def initialize_smoothieboard_connection(port, baud_rate):
    ser = serial.Serial(port, baud_rate)
    time.sleep(2)
    return ser
```

Файл sensor_calibration.py

```
# file: sensor_calibration.py

import time

# Функція автоматичного калібрування осей з використанням сенсорів
def auto_calibrate(ser):
    command = "G28"# Стандартна команда калібрування на кінцеві вимикачі
    send_command(ser, command)
    response = read_response(ser)
    return response

# Функція перевірки даних з сенсорів для калібрування
def check_sensors(ser):
    sensor_data = {}
    sensor_data['X'] = read_sensor_data(ser, "X")
    sensor_data['Y'] = read_sensor_data(ser, "Y")
    sensor_data['Z'] = read_sensor_data(ser, "Z")
    return sensor_data

# Функція зчитування даних з сенсорів для осі
def read_sensor_data(ser, axis):
    command = f"M119 {axis}"# Читання стану кінцевого вимикача для осі
    send_command(ser, command)
    response = read_response(ser)
    return response
```

Файл trajectory_optimization.py

```
# file: trajectory_optimization.py

import math

# Функція лінійної інтерполяції для оптимізації траєкторії
def linear_interpolation(start, end, steps):
    delta = (end - start) / steps
    trajectory = [start + i * delta for i in range(steps + 1)]
    return trajectory

# Функція сплайн-інтерполяції для плавних рухів
def spline_interpolation(points):
# Простий сплайн алгоритм для гладкої траєкторії
    trajectory = []
    for i in range(1, len(points) - 1):
        p0 = points[i - 1]
        p1 = points[i]
        p2 = points[i + 1]
        mid_point = (p0 + p1) / 2
        trajectory.append(mid_point)
    return trajectory

# Функція оптимізації траєкторії руху на основі обчислення кривизни
def optimize_trajectory_by_curvature(points):
    optimized_points = []
    for i in range(1, len(points) - 1):
        curvature = compute_curvature(points[i - 1], points[i], points[i + 1])
        if curvature < threshold:
            optimized_points.append(points[i])
    return optimized_points

# Функція обчислення кривизни
def compute_curvature(p0, p1, p2):
    return abs((p1[0] - p0[0]) * (p2[1] - p1[1]) - (p1[1] - p0[1]) * (p2[0] -
p1[0])) / math.dist(p0, p2)
```

Файл tool_visualization.py

```
# file: tool_visualization.py

import matplotlib.pyplot as plt

# Функція відображення поточного положення інструменту
def plot_tool_position(x, y, z):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(x, y, z, c='r', marker='o')
    ax.set_xlabel('X Coordinate')
    ax.set_ylabel('Y Coordinate')
    ax.set_zlabel('Z Coordinate')
    plt.show()

# Функція оновлення графіка в реальному часі
def update_tool_visualization(ser):
    positions = []
    while True:
        x, y, z = get_current_position(ser)
        positions.append((x, y, z))
        plot_tool_position(x, y, z)

# Функція отримання поточного положення інструменту з контролера
def get_current_position(ser):
    command = "M114"# Запит поточного стану інструменту
    send_command(ser, command)
    response = read_response(ser)
    position = parse_position_response(response)
    return position['X'], position['Y'], position['Z']

# Функція розбору відповіді про позицію
def parse_position_response(response):
    parts = response.split(' ')
    position = {}
    for part in parts:
        axis, value = part.split(':')
        position[axis] = float(value)
    return position
```

Файл remote_control.py

```
# file: remote_control.py

import socket

# Функція налаштування сервера для дистанційного керування
def setup_remote_server(host, port):
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(1)
    return server

# Функція обробки запитів дистанційного керування
def handle_remote_control(server, ser):
    conn, addr = server.accept()
    print(f"З'єднано з {addr}")
    while True:
        data = conn.recv(1024).decode('utf-8')
        if not data:
            break
        send_command(ser, data)
        response = read_response(ser)
        conn.send(response.encode('utf-8'))
    conn.close()

# Функція підключення до сервера для відправки команд
def connect_to_remote_server(host, port):
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((host, port))
    return client

# Функція відправки команди на сервер
def send_command_to_server(client, command):
    client.send(command.encode('utf-8'))
    response = client.recv(1024).decode('utf-8')
    return response
```

Файл gui_control.py

```
# file: gui_control.py

import tkinter as tk

# Функція створення графічного інтерфейсу для керування верстатом ЧПК
def create_gui(ser):
    root = tk.Tk()
    root.title("CNC Controller")

# Поле для введення команд
    command_entry = tk.Entry(root, width=50)
    command_entry.pack()

# Кнопка для відправки команди
    send_button = tk.Button(root, text="Send Command", command=lambda:
send_gui_command(ser, command_entry.get()))
    send_button.pack()

# Поле для виводу результатів
    result_label = tk.Label(root, text="")
    result_label.pack()

# Кнопка для завершення роботи програми
    exit_button = tk.Button(root, text="Exit", command=root.quit)
    exit_button.pack()

    root.mainloop()

# Функція відправки команди з GUI
def send_gui_command(ser, command):
    send_command(ser, command)
    response = read_response(ser)
    result_label.config(text=response)
```

Файл tool_recognition.py

```
# file: tool_recognition.py

# Словник з типами інструментів і їх характеристиками
tool_library = {
    "drill": {"diameter": 5.0, "type": "drill", "length": 100},
    "milling_cutter": {"diameter": 10.0, "type": "milling", "length": 120},
    "lathe_tool": {"diameter": 12.0, "type": "lathe", "length": 150},
}

# Функція розпізнавання інструменту за введеними даними
def recognize_tool(tool_id):
    if tool_id in tool_library:
        return tool_library[tool_id]
    else:
        return "Інструмент не знайдено в базі"

# Функція додавання нового інструменту в бібліотеку
def add_tool(tool_id, diameter, tool_type, length):
    tool_library[tool_id] = {"diameter": diameter, "type": tool_type, "length":
length}

# Функція виводу інформації про інструмент
def get_tool_info(tool_id):
    tool_info = recognize_tool(tool_id)
    return tool_info
```

Файл gcode_formats.py

```
# file: gcode_formats.py

# Функція завантаження G-коду з файлу
def load_gcode(file_path, format_type="standard"):
    with open(file_path, 'r') as file:
        gcode = file.readlines()

    if format_type == "standard":
        return parse_standard_gcode(gcode)
    elif format_type == "extended":
        return parse_extended_gcode(gcode)
    else:
        raise ValueError("Непідтримуваний формат G-коду")

# Функція парсингу стандартного формату G-коду
def parse_standard_gcode(gcode):
    parsed_code = []
    for line in gcode:
        parsed_code.append(line.strip())
    return parsed_code

# Функція парсингу розширеного формату G-коду
def parse_extended_gcode(gcode):
    parsed_code = []
    for line in gcode:
        if line.startswith(';'):
            continue# Ігнорувати коментарі
        parsed_code.append(line.strip())
    return parsed_code

# Функція збереження G-коду у файл
def save_gcode(file_path, gcode):
    with open(file_path, 'w') as file:
        for line in gcode:
            file.write(line + '\n')
```

Файл ai_correction.py

```
# file: ai_correction.py

import random

# Симуляція алгоритму машинного навчання для корекції помилок
def ai_error_correction(gcode_lines):
    corrected_gcode = []
    for line in gcode_lines:
        if has_error(line):
            corrected_line = correct_gcode_line(line)
            corrected_gcode.append(corrected_line)
        else:
            corrected_gcode.append(line)
    return corrected_gcode

# Функція перевірки на помилки в рядку G-коду
def has_error(gcode_line):
    return random.choice([True, False])# Симуляція виявлення помилки

# Функція виправлення рядка G-коду
def correct_gcode_line(gcode_line):
    return gcode_line.replace("X", "Y")# Проста корекція для прикладу

# Функція обробки всього файлу G-коду
def process_gcode_with_ai(gcode_file):
    gcode_lines = load_gcode(gcode_file)
    corrected_gcode = ai_error_correction(gcode_lines)
    save_gcode("corrected_" + gcode_file, corrected_gcode)
```

Файл logging_analytics.py

```
# file: logging_analytics.py

import time

# Функція логування виконаних команд
def log_command(command, status):
    with open("command_log.txt", "a") as log_file:
        log_file.write(f"{time.strftime('%Y-%m-%d %H:%M:%S')} - {command} -
{status}\n")

# Функція аналізу виконаних команд
def analyze_logs():
    command_count = {}
    with open("command_log.txt", "r") as log_file:
        for line in log_file:
            command = line.split(" - ")[1]
            if command in command_count:
                command_count[command] += 1
            else:
                command_count[command] = 1
    return command_count

# Функція виводу аналітики
def print_analytics():
    analytics = analyze_logs()
    for command, count in analytics.items():
        print(f"Команда: {command}, Використана: {count} разів")
```

Файл material_instructions.py

```
# file: material_instructions.py

# Бібліотека матеріалів та інструкцій до них
material_library = {
    "aluminum": {"speed": 1000, "feed_rate": 200, "cooling": True},
    "steel": {"speed": 800, "feed_rate": 150, "cooling": True},
    "plastic": {"speed": 1200, "feed_rate": 250, "cooling": False},
}

# Функція вибору інструкцій для матеріалу
def get_instructions_for_material(material):
    if material in material_library:
        return material_library[material]
    else:
        return "Інструкції для матеріалу не знайдено"

# Функція додавання нового матеріалу в бібліотеку
def add_material_instructions(material, speed, feed_rate, cooling):
    material_library[material] = {"speed": speed, "feed_rate": feed_rate,
"cooling": cooling}

# Функція виводу інструкцій
def print_material_instructions(material):
    instructions = get_instructions_for_material(material)
    print(f"Матеріал: {material}")
    print(f"Швидкість: {instructions['speed']}")
    print(f"Швидкість подачі: {instructions['feed_rate']}")
    print(f"Охолодження: {instructions['cooling']}")
```

Файл simulation_3d_modeling.py

```
# file: simulation_3d_modeling.py

from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

# Функція для 3D-візуалізації моделі
def plot_3d_model(tool_path):
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')

    x_data = [point[0] for point in tool_path]
    y_data = [point[1] for point in tool_path]
    z_data = [point[2] for point in tool_path]

    ax.plot(x_data, y_data, z_data, label="Tool Path")
    ax.set_xlabel('X axis')
    ax.set_ylabel('Y axis')
    ax.set_zlabel('Z axis')
    plt.show()

# Функція запуску симуляції
def run_simulation(gcode):
    tool_path = []
    for line in gcode:
        if "G01" in line:
   # Простий розбір для координат X, Y, Z
            coords = parse_gcode_for_coords(line)
            tool_path.append(coords)
    plot_3d_model(tool_path)

# Функція розбору рядка G-коду для отримання координат
def parse_gcode_for_coords(gcode_line):
    coords = {"X": 0, "Y": 0, "Z": 0}
    parts = gcode_line.split()
    for part in parts:
        if part.startswith("X"):
            coords["X"] = float(part[1:])
        elif part.startswith("Y"):
            coords["Y"] = float(part[1:])
        elif part.startswith("Z"):
            coords["Z"] = float(part[1:])
    return coords["X"], coords["Y"], coords["Z"]
```

Файл voice_control.py

```
# file: voice_control.py

import speech_recognition as sr

# Ініціалізація розпізнавача
recognizer = sr.Recognizer()

# Функція отримання голосової команди
def listen_for_command():
    with sr.Microphone() as source:
        print("Слухаю команду...")
        audio = recognizer.listen(source)
    try:
        command = recognizer.recognize_google(audio, language="uk-UA")
        return command
    except sr.UnknownValueError:
        return "Невдалося розпізнати голосову команду"
    except sr.RequestError:
        return "Помилка підключення до сервера розпізнавання"

# Функція виконання голосової команди
def execute_voice_command(command):
    if "рух" in command:
        move_x_axis(ser, 100)# Простий приклад виконання
    elif "зупинка" in command:
        stop_cnc_program(ser)
    else:
        print("Невідома команда")
```

Файл multitasking_support.py

```
# file: multitasking_support.py

import threading

# Функція для запуску окремої програми в новому потоці
def run_program_in_thread(gcode_file):
    thread = threading.Thread(target=run_program, args=(gcode_file,))
    thread.start()

# Функція виконання програми ЧПК
def run_program(gcode_file):
    gcode_lines = load_gcode(gcode_file)
    for line in gcode_lines:
        send_command(ser, line)
        read_response(ser)
```