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
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Василенко Д.М.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

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
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Василенку Дмитру Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Лисенко І.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Василенко Д.М.
(прізвище та ініціали)

АНОТАЦІЯ

Василенко Д.М. Програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи автоматичного керування сушаркою на основі ARM STM32F429.

Метою розробки є програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429.

Результат роботи – програмна реалізація системи автоматичного керування сушаркою на основі ARM STM32F429.

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, автоматичне керування, сушарка, ARM STM32F429

ABSTRACT

Vasylenko D.M. Software for the automatic dryer control system based on ARM STM32F429. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the automatic dryer control system based on ARM STM32F429.

The purpose of the development is the software for the automatic dryer control system based on ARM STM32F429.

The result of the work is the software implementation of the automatic dryer control system based on ARM STM32F429.

In the process of working on the software model, an analysis of existing hardware and software tools 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.

The program is developed in Python.

Keywords: computer engineering, automatic control, dryer, ARM STM32F429

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БП – блок перетворення

БРІУ – блок вимірювання, керування та індикації

ЗВЧ – зверх високі частоти

ІВТМ – вимірювач вологості та температури.

КПК – карманий персональний комп’ютер

МК – мікроконтролер

ПЗ – програмне забезпечення

ПК – персональний комп’ютер

ПІ – перетворювач інтерфейсу

ЯМР – ядерний магнітний резонанс

GPRS – стандарт бездротової передачі даних

GSM – стандарт мобільного зв’язку

IP – рівень мережної моделі передачі даних

PIC – мікроконтролер

RAD – набір візуальних інструментів для швидкісної розробки додатків

RS-232 – протокол обміну даними

RS-485 – протокол обміну даними

USB – universal serial bus

VCL – бібліотека візуальних компонентів

ВСТУП

Актуальність теми. Інтелектуальне програмне забезпечення для керування моніторингом зерна – це все, що надійне обладнання та нове інтелектуальне програмне забезпечення можуть зробити для вас. Скористайтеся всіма перевагами новітніх технологій і дозвольте системі моніторингу зерна надати вам потужні інструменти, необхідні для постійного контролю над вашим зерном і посівами.

Захистіть свої врожаї, а також забезпечте швидке та надійне повернення інвестицій

1. Швидке повернення інвестицій. Захищаючи свої врожаї та зменшуючи відходи, ви максимізуєте свої прибутки.

2. Максимізація прибутку. Програмне забезпечення допоможе вам керувати кількома об’єктами та контролювати їх на одній унікальній інформаційній панелі.

3. Кілька об'єктів на одній платформі. Системи моніторингу зернових бункерів захищають урожай, який зберігається, і зводять до мінімуму відходи.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429.

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

– Огляд існуючих систем автоматичного керування сушаркою на основі ARM STM32F429.

– Дослідження системи автоматичного керування сушаркою на основі ARM STM32F429.

– Програмна реалізація системи автоматичного керування сушаркою на основі ARM STM32F429.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі автоматичного керування сушаркою на основі ARM STM32F429.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Наведемо переваги сільськогосподарської сушарки:

– максимальна енергоефективність: сушильні системи характеризуються мінімальним енергоспоживанням, що не тільки знижує експлуатаційні витрати, але й захищає навколишнє середовище. Завдяки інноваційним технологіям рекуперації тепла ми гарантуємо, що енергія не витрачається даремно.

– якість і довговічність: сушильні системи стабільні і довговічні, виготовлені з високоякісних матеріалів і відповідають суворим вимогам.

– гнучкість і адаптивність: наші системи сушіння гнучко розроблені та можуть бути адаптовані до індивідуальних потреб вашого бізнесу. Ми пропонуємо вам відповідне, індивідуальне рішення – незалежно від того, чи потрібні вам сільськогосподарські сушарки для стаціонарного чи мобільного використання .

– зручність у використанні: наші сушильні системи характеризуються інтуїтивно зрозумілою роботою та низькими вимогами до обслуговування . Завдяки найсучаснішим системам управління від нашої власної компанії моніторинг і контроль процесів сушіння не вимагає зусиль.

– стійкість: ми надаємо велике значення захисту навколишнього середовища та економному використанню ресурсів. Наші системи сушіння дозволяють зменшити викиди CO_2.

1.2 Область застосування

Сільськогосподарські сушарки використовуються в широкому діапазоні сільськогосподарських застосувань:

– Сушіння зерна: оптимальне сушіння для максимальної надійності врожаю. Наші мобільні та стаціонарні зерносушарки забезпечують максимальну ефективність.

– Сушіння кукурудзи: щоб забезпечити термін зберігання та якість кукурудзи, технологія забезпечує ефективне та дбайливе сушіння, що максимізує енергоефективність.

– Сушіння насіння: сушарки періодичної дії та безперервні сушарки забезпечують схожість і якість вашого насіння.

– Бобові: високоефективне сушіння бобів, гороху та сочевиці. Сушарки для особливих потреб, наприклад для сої, також доступні.

– Рисові сушарки для рису-педі та пропареного рису: щоб переробити зерно, яке найбільше споживається у світі, рисові зерна слід обережно висушити після збирання.

– Сушарки безперервної дії: для безперервного сушіння сипучих культур у компактній та компактній конструкції.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Керування АГРОбізнесом для України

Компанією «Кварц» розроблений програмний продукт обліку на елеваторі "Керування АГРОбізнесом для України", призначений для автоматизації бухгалтерського обліку, податкового обліку, кількісно-якісного й управлінського обліку на хлібоприймальних, заготівельних і зернопереробних підприємствах України. Дане рішення створює єдиний інформаційний простір на підприємстві для контролю й керування всіма необхідними процесами.

Програма "Керування АГРОбізнесом для України" дозволить Вам автоматизувати бухгалтерський і податковий облік, включаючи підготовку обов'язкової (регламентованої) звітності. Контур програми «Елеватор, млин, комбікормовий завод» дає можливість ведення обліку для елеватора, як для окремого суб'єкта господарювання, так і для елеватора, що входить у групу різноманітних по виду діяльності підприємств сільськогосподарського напрямку.

Облік діяльності елеватора в програмному продукті «Керування Агробізнесом для України», ведеться в строгій відповідності чинному законодавству в цій сфері.

Даний контур дозволяє автоматизувати повний цикл роботи:

– кількісно – якісний облік хлібоприймального підприємства (елеватора, борошномельного й комбікормового заводу), що включає надходження зерна автомобільним і залізничним транспортом (форма 1 – СГ);

– виявлення й відображення ПТЛ якісних показників зерна, що надійшло, у картці аналізу зерна (форми № 47);

– рецепти, проведення чищення/сушіння зерна за формою № 34 (з можливістю формування друкованих форм, акту розподілу відходів, акту розподілу умовного сушіння, акту знищення непридатних відходів і звіту про використання зрідженого газу);

– формування акту – розрахунку й попереднього акту – розрахунку;

– операції переоформлення (друковані форми акту на передачу хлібопродуктів, пропуску, акт на переоформлення (прийому – передачі);

– формування реєстрів ТТН на ввіз автомобільним і залізничним транспортом (форми ЗХС – 3);

– вивіз зерна з території оформляється наказом на вивіз, формуються реєстри ТТН на вивіз;

– створення квитанцій зі складу, простих і подвійних складських свідчень із можливістю їхнього вивантаження в програму «ЕРЗС – Реєстратор ІІІ»;

– автоматизована робота вагаря, можливий автоматичний розрахунок вартості послуг (зберігання, очищення, сушіння, зважування й т.д.), ведення форми 36 і форми 37;

– проведення зачищення виробничих приміщень (Акт зачищення) і спеціалізована звітність (аналіз валового збору, журнал лабораторних аналізів, журнал вагаря, журнал квитанцій зі складу й ін.);

– реалізований облік виробництва й переробки: форма № 117, акт списання, прихід з виробництва, фасовка, спеціалізована звітність (аналіз розрахункового випуску продукції, розрахунки із власниками, виробничий звіт і ін.).

При використанні даного контуру Ви зможете систематизувати документообіг, упорядкувати основні бізнес – процеси ХПП, спростити взаємодію ХПП із Державною хлібною інспекцією.

Системи моніторингу AgroLog для бункерів і силосів

TMS2000 Для вас, кому потрібно знати температуру всередині ваших силосів

Кожен силос потребує точності та надійної системи моніторингу температури. Наш моніторинг зернових бункерів TMS2000 – це базовий професійний портативний термінал і блок зчитування.

- TMS2000 + обладнання.
- Контроль безпеки.
- Портативний пристрій.
- Легко встановити.
- Проста у використанні система.

Ця система є гнучкою і може бути розширена в будь-який момент. Ідеально підходить як надійна та проста стартова система для моніторингу температури у ваших силосах або насипних сховищах.

TMS2500 Для вас, кому потрібно мати можливість стежити за розвитком

Наша система моніторингу зернового бункера TMS2500 – це компактний портативний ручний термінал із вбудованою пам'яттю для документованого моніторингу температури. Він також містить програмне забезпечення для ПК, щоб надати вам швидкий огляд.

- TMS2500 + обладнання + програмне забезпечення для ПК.
- Контроль безпеки. Портативний пристрій.
- Збережені дані та журнал температури.
- Легко встановити.
- Проста у використанні система.

Ця система надає детальну історичну інформацію. Температурні діаграми та таблиці генеруються, щоб увімкнути температурні тенденції, які аналізуються з часом.

TMS6000 Для вас, кому потрібна повністю автоматизована система

Ця система моніторингу зерна повністю використовує переваги нової технології. Наша система моніторингу зерна AgroLog надає вам потужні інструменти, необхідні для постійного контролю над вашим зерном і врожаєм.

- TM6000 + апаратне забезпечення + індивідуальне програмне забезпечення.
- Контроль безпеки.
- Автоматичні функції.
- Автоматичний журнал і історичні дані.
- Налаштування будильника.
- Аерація – ручна або автоматична.
- Проста у використанні система та доступ звідусіль.

AgroLog від Supertech для вас, кому потрібна повністю автоматизована система моніторингу температури та вологості, щоб зробити ваш щоденний робочий процес простішим і набагато безпечнішим.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також

містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи автоматичного керування сушаркою на основі ARM STM32F429.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) 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

Перед аграрно-промисловим комплексом України в цей час коштують складні завдання по вдосконалюванню керування виробництвом, підвищенню його якості й ефективності, забезпеченню широкого застосування автоматизованих систем керування, побудованих на базі мікро- і міні ЕОМ.

Одним з основних факторів підвищення якості зерна й насіння є сушіння – обов'язковий етап технологічного процесу сільськогосподарського виробництва в умовах зон підвищеного зволоження. Зерновий матеріал, висушений до кондиційної вологості, довгостроково й безпечно зберігається, що рівноцінно додатковому виробництву й збереженню продукції, дозволяє використовувати частину посівних площ під інші культури й дає значний економічний ефект у масштабах країни.

Розробка системи керування для технологічного процесу сушіння зернових матеріалів включає кілька етапів.

Метою першого етапу є дослідження технологічного процесу як об'єкта автоматизації, визначення принципів побудови системи керування. Основні завдання першого етапу: аналіз технологічного процесу й існуючих систем керування, виявлення їхніх особливостей і недоліків; формування мети створення й функцій системи керування, вибір состава комплексу технічних засобів; удосконалювання й розробка методів керування нестаціонарними об'єктами й систем автоматичного контролю й керування окремими технологічними параметрами.

При рішенні зазначених завдань варто враховувати, що інтенсифікація сушіння зернових матеріалів у камерах періодичної дії, спрямована на підвищення ефективності діючих і проектованих систем керування в сушильних

камерах, приводить до ускладнення технологічних процесів сушіння як об'єктів керування [2].

Отримавши широке поширення релейні й лінійні системи автоматичного регулювання не можуть забезпечити заданої якості й надійності керування через відсутність необхідної апріорної інформації про об'єкт [1]. Тому в пропоновані нами системах автоматизованого керування (САУ) як регулятори вологості, температури (для позоних зерносушарок і експозиції) використовуються безконтактні логічні пристрої на основі тиристорів і сімісторов [2]. У зв'язку із цим для поліпшення якості керування в системах керування процесом сушіння зерна доцільно використовувати адаптивні системи керування, що дозволяють пристосовуватися до умов, що змінюються, за рахунок одержання, обробки й аналізу за допомогою адаптивного керуючого пристрою відсутньої інформації про керований процес [3].

Використовуючи адаптивні системи, можна вирішувати широке коло завдань, у який входять не тільки завдання регулювання, але також і завдання знаходження оптимальних умов роботи системи в цілому, керування об'єктами при параметричних збурюваннях і при наявності перешкод.

З'являється усе більше робіт з використання адаптивних систем керування й адаптивних регуляторів для об'єктів керування в різних галузях промисловості, що дозволяє сподіватися на розширення сфери їхнього застосування. Цьому сприяють постійне вдосконалювання структури адаптивних систем керування, використовуваних методів ідентифікації [4], і застосування нових типів керуючих міні- і мікроЕОМ. [4].

Інтенсифікація режимів сушіння зерна й підвищення вимог до якості матеріалів, що висушуються, приводять до необхідності оснащення зерносушильних установок новими контролюючими приладами й системами автоматичного керування, що дозволяють вирішити питання контролю й керування процесами сушіння в камерах періодичної дії.

До теперішнього часу вивчені особливості сушильних камер періодичної дії як об'єктів автоматичного керування, однак створені системи автоматичного регулювання температури й психрометрічної різниці сушильного агента не мають корекції режимів по параметрах матеріалу, що висушується, (по середній поточній вологості й температурі), що не дозволяє здійснити комплексну автоматизацію процесу сушіння. Ці системи не реалізують негативні зворотні зв'язки по параметрах стану процесу (тобто поточним параметрам матеріалу, що висушується,) і, по суті, є стабілізуючими системами автоматичного регулювання, що дозволяють автоматично підтримувати задані параметри агента сушіння на вході в певних діапазонах значень. Створено систему й прилади для виміру середньої поточної вологості зернового матеріалу під час камерного сушіння, однак вона має потребу в удосконалюванні, особливо при використанні в рамках АСУ ТП. Основою для створення автоматизованих систем служить система контролю, що забезпечує повну наблюдаємість об'єктів керування.

Існуюча в цей час система контролю процесу сушіння реалізована на локальних засобах автоматизації, у якій багато важливі з погляду керування процесами параметри або не змінюються, або змінюються зі значними помилками й великими часовими запізнюваннями. Це не дозволяє об'єктивно й оперативно оцінювати стан процесів, що в остаточному підсумку погіршує якість керування ними.

Крім того, існуюча система, у принципі, не може забезпечити виконання наступних функцій:

– оперативну діагностику стану встаткування;

– автоматичну сигналізацію про порушення технологічного режиму;

– оперативний розрахунок техніко-економічних показників роботи відділень прийому й підготовки до сушіння;

– документування технологічної й техніко-економічної інформації.

Зазначені обставини спричиняються актуальність роботи, спрямованої на розробку системи керування на базі мікроЕОМ технологічним процесом сушіння

зерна в зерносушарках сільськогосподарського призначення з камерами періодичної й безперервної дії.

Погрішності визначення середньої вологості доходять до 5...8%, що обумовлено помилками у визначенні "сухого" ваги зразка й дискретністю контролю. Зниження якості зерна (коагуляція білків, термічне й механічне травмування й ін.) при сушінні значною мірою викликається порушеннями раціональних режимів роботи через неточний контроль вологості.

Результати досліджень, проведених у рамках першого етапу, викладені в справжній роботі. У наступних статтях передбачається опублікувати результати, отримані при реалізації завдання побудови системи централізованого контролю, розробки структури комплексу технічних засобів, а також її інформаційного, алгоритмічного й програмного забезпечення (другий етап). Наступний (третій) етап був присвячений розробці алгоритмічного забезпечення системи керування технологічним процесом сушіння зерна й рішенню завдань, пов'язаних з розробкою адаптивної системи автоматичного керування й створенням програмного забезпечення системи керування технологічним процесом, що дозволяють більш повно задовольняти системі принципів [7], що забезпечують ефективність, інтенсивність і оптимальність досліджуваного технологічного процесу.

3.2 Розробка структурної схеми

Розглянемо типи датчиків, які використовуються для сушіння зерна.

Типи датчиків

Моніторинг температури

Уникайте псування зерна, виявляючи гарячі точки. Регулярно перевіряйте температуру, доки не буде досягнуто цільової температури. Більш низькі температури дозволяють зберігати зерно при більш високому вмісті вологи. Комахи, кліщі, грибки та мікротоксини розвиваються при вищих температурах.

Моніторинг вологості

Для кращої якості та безпеки зберігання. Наші системи моніторингу зерна пропонують контроль вологості та усадки шляхом вимірювання вмісту вологи та температури в кількох точках по всьому силосу.

Моніторинг рівня

Важливим компонентом є відстеження запасів у силосах. Використання програмного забезпечення і нашого автоматичного механічного датчика рівня дозволяє легко дізнатися, де знаходяться матеріали та скільки їх є.

Контроль аерації

Автоматичне керування вентилятором оптимізує якість зерна. Оптимізація вологості та температури зерна. Уникайте усадки. Наш моніторинг температури та вологості в поєднанні з даними про температуру та вологість навколишнього середовища автоматично запускає/зупиняє роботу вентиляторів.

Моніторинг CO2

Датчики CO2 можуть забезпечити раннє виявлення псування. Комахи та цвіль – це організми, які дихають і виділяють CO2. Газ буде рухатися до верхнього простору силосу завдяки внутрішньому висхідному потоку повітря

Моніторинг головного простору

Уникайте конденсації та вологого зерна на поверхні зерна. Вентилятори на даху керуються даними датчика простору, а точка роси обчислюється на основі температури та вологості

Моніторинг навколишнього середовища

Моніторинг погодних умов забезпечує чудовий контроль за аерацією. Метеостанція постійно стежить за умовами навколишнього середовища. Переконайтеся, що для аерації використовується лише повітря з відповідними властивостями

STM32F429

STM32F4 серія випускається в корпусі, починаючи з 64 ніжок або 64 пінів, що не цілком достатньо. Якщо відкинути виводи живлення, виводи підключення

кварцового резонатора, виводи, необхідні для забезпечення обміну з комп'ютером і з огляду на можливість підключення екрана, то виявиться що для паралельного обміну ніжок у мікроконтролера швидше за все не вистачить.

Тому вибір припав на STM32F429, що має 100 ніжок і до того ж використовується в відладочній платі STM32F4DISCOVERY. Із цих 100 ніжок як виводи порту уведення/виводу можна використовувати тільки 82. Інші 18 потрібні для забезпечення роботи мікроконтролера. Ці 18 ніжок потрібні тільки для того щоб подати живлення на мікроконтролер і підключити кварцовий резонатор.

Практично всі виводи мікроконтролера STM32 можуть виконувати альтернативну функцію. Той самий вивід може використовуватися, наприклад, як вивід порту уведення/виводу, як вивід зовнішнього переривання, як уведення аналого-цифрового перетворювача, як вивід порту USB або чого-небудь ще. Яку альтернативну функцію буде виконувати вивід мікроконтролера, визначається вмістом спеціальних регістрів. Це дозволяє на тому самому мікроконтролері створювати різні по призначенню пристрою. Досить визначитися з конфігурацією й записати її в спеціальні регістри.

Порти уведення/виводу в мікроконтролері є 16-ти розрядними й мають літерне позначення PA(15..0) – це GPIO PORT A, PB(15..0) – це GPIO PORT B і т.д. Використовувати кожної з них не вийти, тому що ті самі виводи мікроконтролера можуть використовуватися для різних убудованих пристроїв і якщо цей пристрій використовується, то використовувати ті ж ніжки для іншого вже не рекомендується, а іноді це й неможливо.

Для мене це означає, що якщо буде задіяний для обміну з комп'ютером порт USB або який-небудь із USART, те мені не бажано використовувати відповідний порт уведення/виводу. Наприклад, USART1 частково використовує виводи порту уведення/виводу "A" (GPIO PORT A) і тому якщо я буду використовувати USART1, то мені не бажано використовувати виводи, що залишилися, GPIO PORT A як шина обміну даними. Мені треба мінімум 26

розрядів і добре, якщо це будуть два вільних порти уведення/виводу (GPIO PORT). Це спростить програмування алгоритму роботи.

Можливості мікроконтролера STM32F4xxx і його швидкодію дозволяють використовувати практично будь-які набори виводів для організації обміну даними. Але для цього необхідно буде щораз конфігурувати кожний вивід або невелику групу виводів. Якщо ж використовувати виводи одного порту, то зміна конфігурації, на мій погляд, і досвід програмування мікроконтролера AT91SAM7X, робити буде простіше й легше.

Напруга живлення мікроконтролера STM32F4xxx становить 3,3 вольти. Старе встаткування харчувалося в основному від 5 вольтів. Це обставина мало-мало не змусила мене відмовитися від мікроконтролера STM32F4xxx, але виявилося що порти уведення/виводу (GPIO PORT) сумісні або толерантні з напругою 5У. Тобто, тут перешкод для використання немає. Єдине потрібно буде врахувати, що виходи старого встаткування працюють за схемою відкритий колектор, тобто потрібно буде забезпечити навантаження при прийманні даних.

Пам'яті для програм і ОЗП для виконання цих програм цілком достатньо.

Система виконує наступні завдання:

1. Керування транспортуванням зерна:

а) керування норіями, формування маршрутів транспортування (у тому числі рециркуляційних);

б) автоматичне завантаження зерна в надсушильний бункер відповідно до показань дискретних датчиків рівня зерна;

в) вивантаження зерна відповідно до заданого часового циклу, або – як можливий варіант – по досягненню заданої вологості (при наявності можливості виміру вихідної вологості зерна);

г) взаємодія із глобальною транспортною системою елеватора шляхом збору й передачі зовнішніх сигналів керування й блокування комплексу.

2. Вимір вологості зерна.

3. Вимір температури зерна (3 зони в кожній з веж зерносушарки).

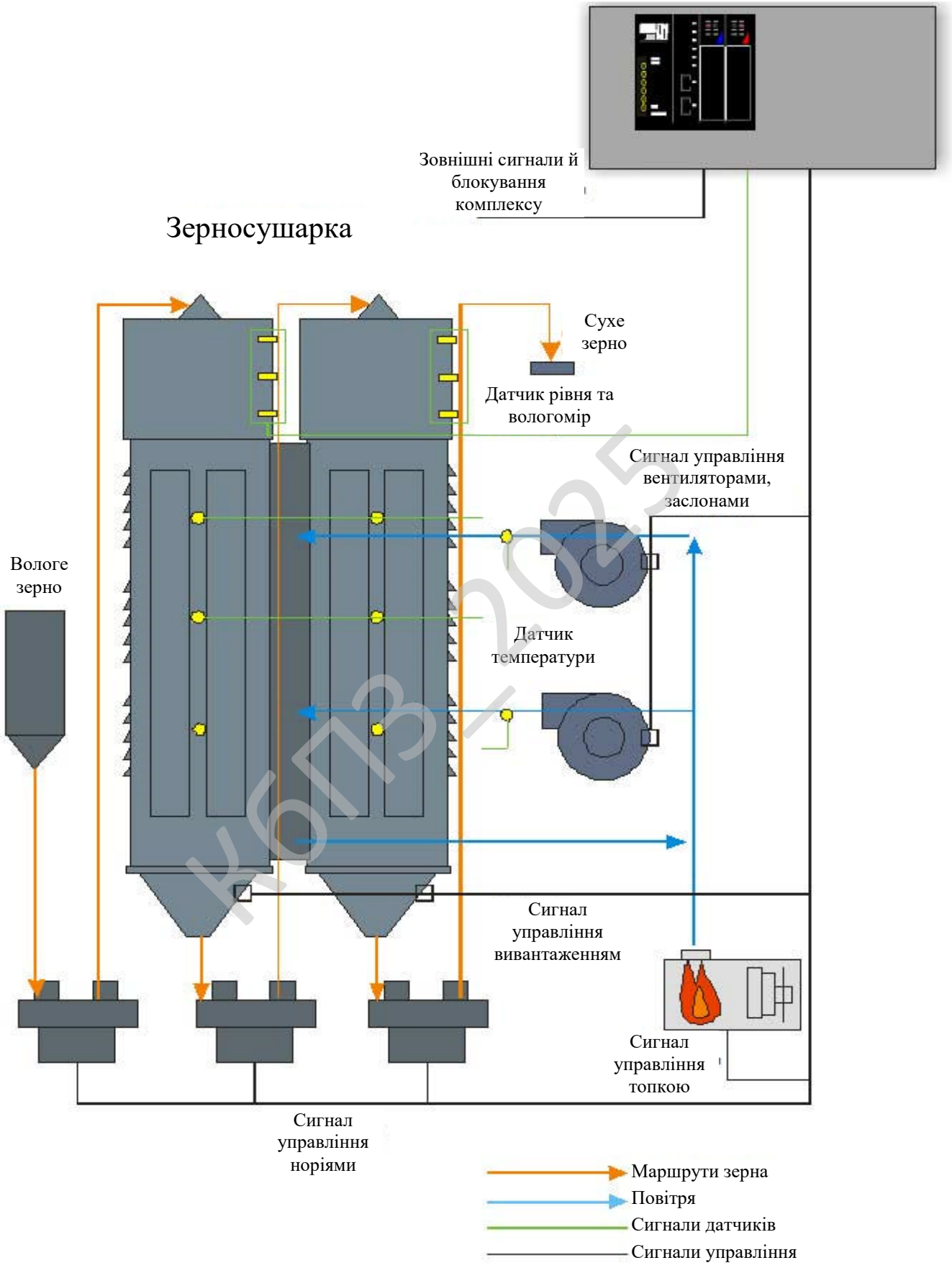

Рисунок 3.1 – Структурна схема системи

4. Керування процесом сушіння:

а) регулювання температури повітря (шляхом завдання температури казанів і за допомогою повітряних заслінок, можлива рециркуляція повітря);

б) ведення системи рецептів із заздалегідь заданими технологічними параметрами сушіння, маршрутами, часовими циклами й т.п.

5. Візуалізація поточного стану системи на сенсорній панелі оператора (можлива інтеграція в систему верхнього рівня автоматизації, або використання вилученої диспетчеризації):

а) відображення поточного стану системи на екрані панелі у вигляді мнемосхеми;

б) візуалізація показань датчиків;

в) видача текстових повідомлень про виниклі аварійні ситуації, блокування й т.п. із записом в архівний журнал.

6. Ручне керування окремими вузлами системи із кнопкового пульта на передній панелі шафи автоматики.

У запропоновану систему автоматизації не входить безпосереднє керування газовим пальником – передбачається використання готової системи, з можливістю подачі загальних сигналів керування (пуск/останов пальника, завдання температури, сигнали стану, т.д.)

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена розроблена функціональна схема програмної частини дипломного проекту. З неї видно як розроблена програма взаємодіє з апаратною частиною. Розроблена дипломна програма розбита на такі частини:

1. Взаємодія з комунікаційним інтерфейсним модулем на основі ARM STM32F429.

2. Взаємодія з зовнішнім модулем АЦП/ЦАП.

3. Інтерфейс ПЗ.

4. Захист ПЗ.

5. Обробка даних.

6. Налагодження ПЗ.

7. Довідкова інформація.

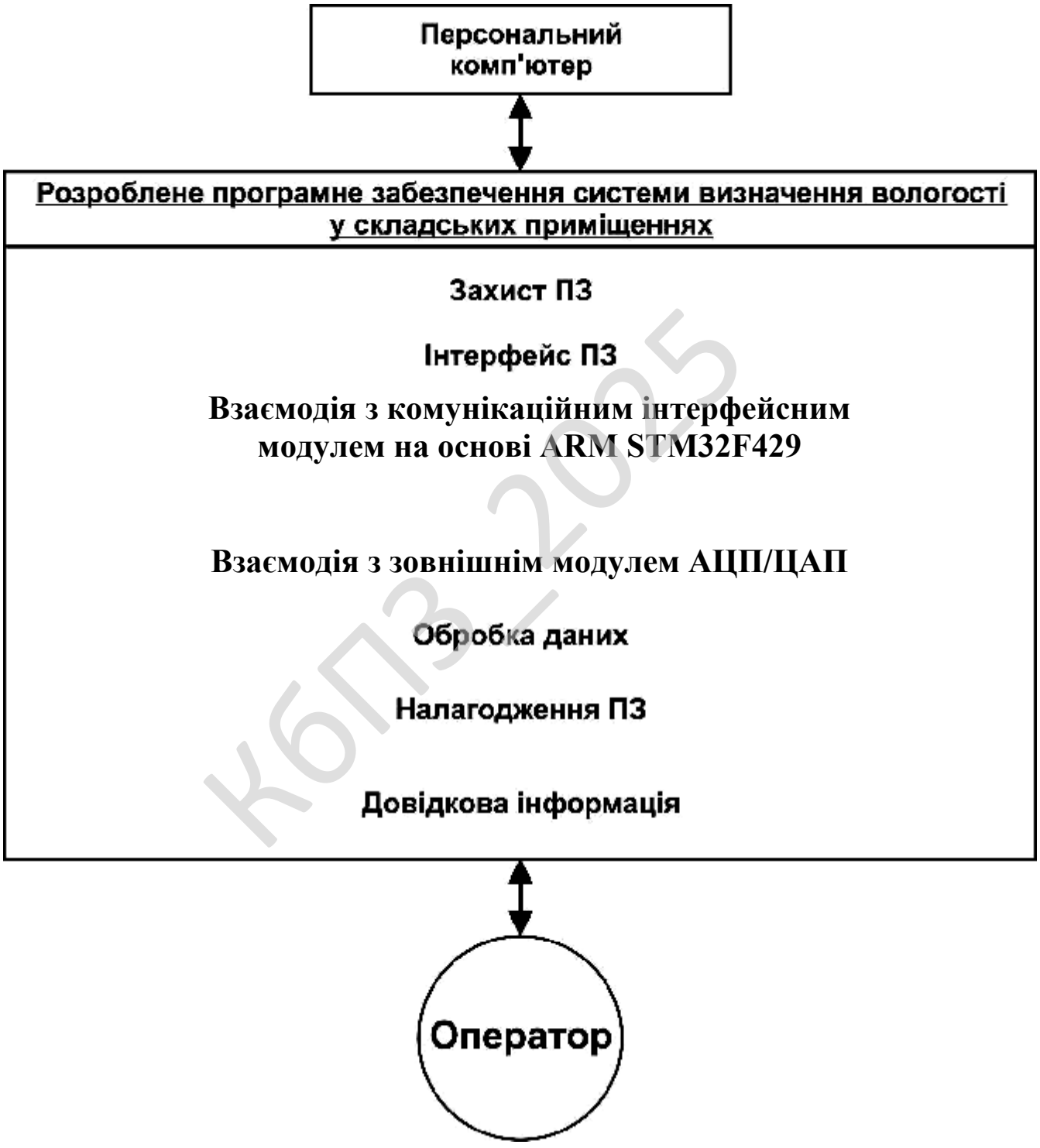

Рисунок 3.2 – Функціональна схема програмної частини

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.

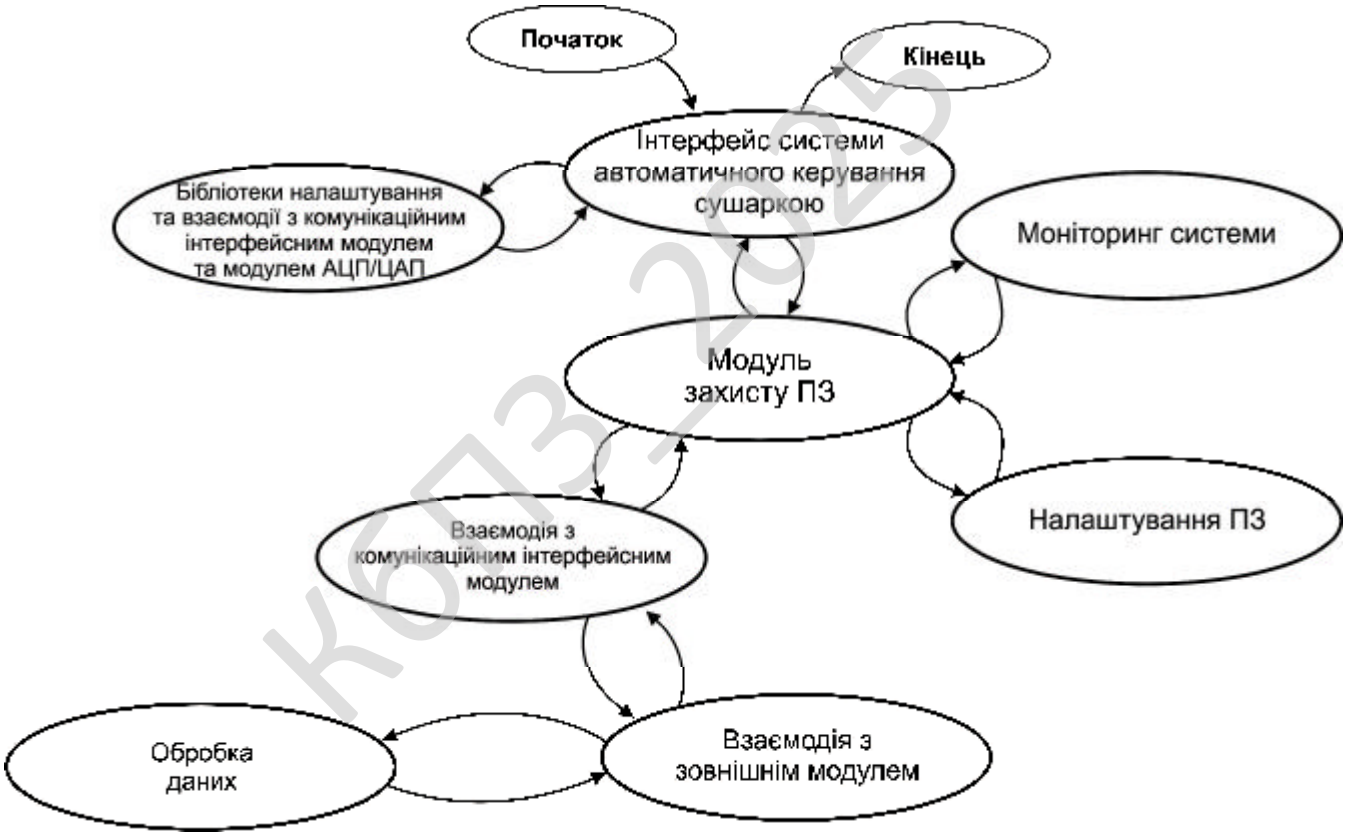

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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

Під час роботи над бакалаврською 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.

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 системи автоматичного керування сушаркою на основі ARM STM32F429.

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.

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.

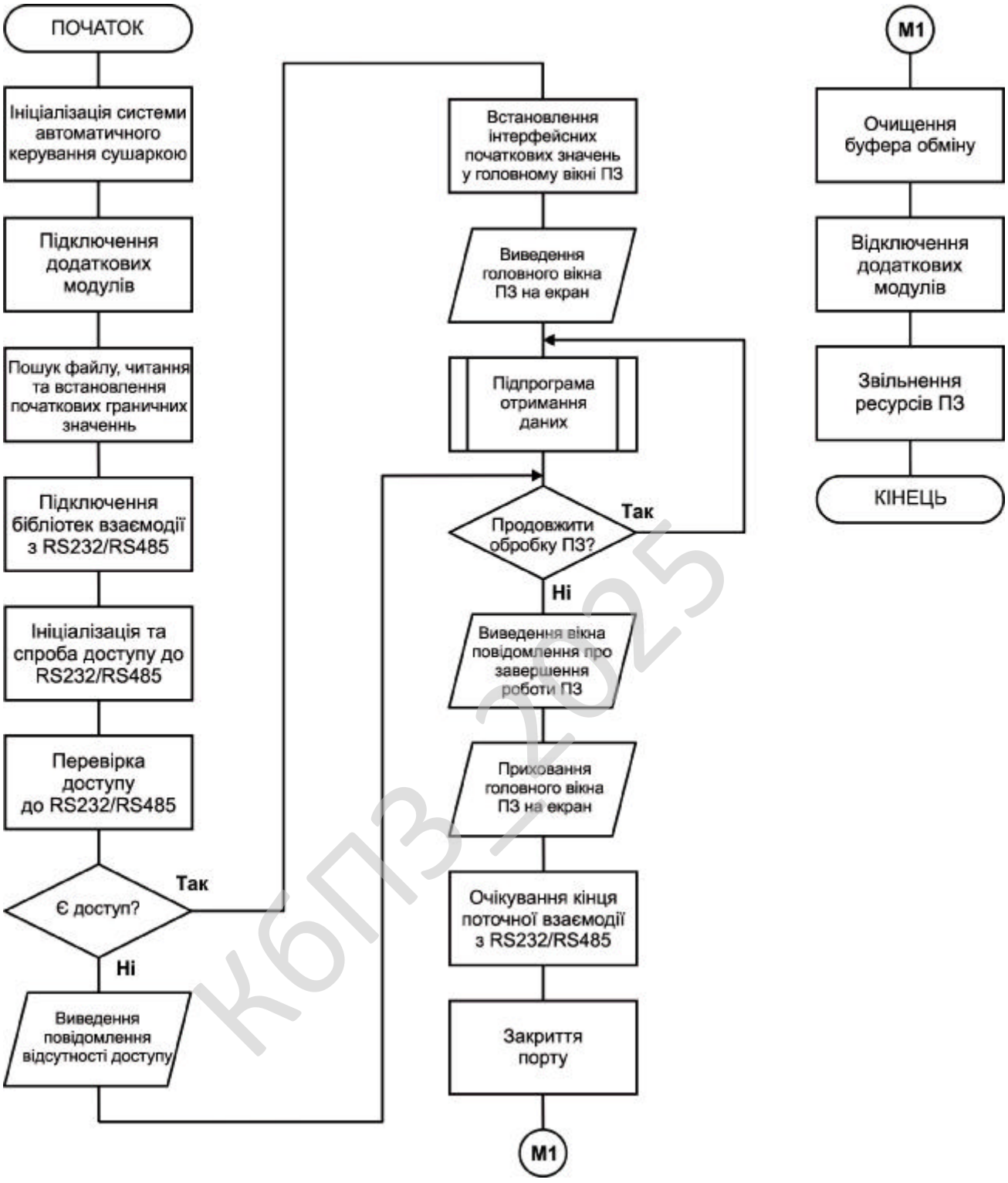

Рисунок 4.1 – Блок-схема основної програми

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.

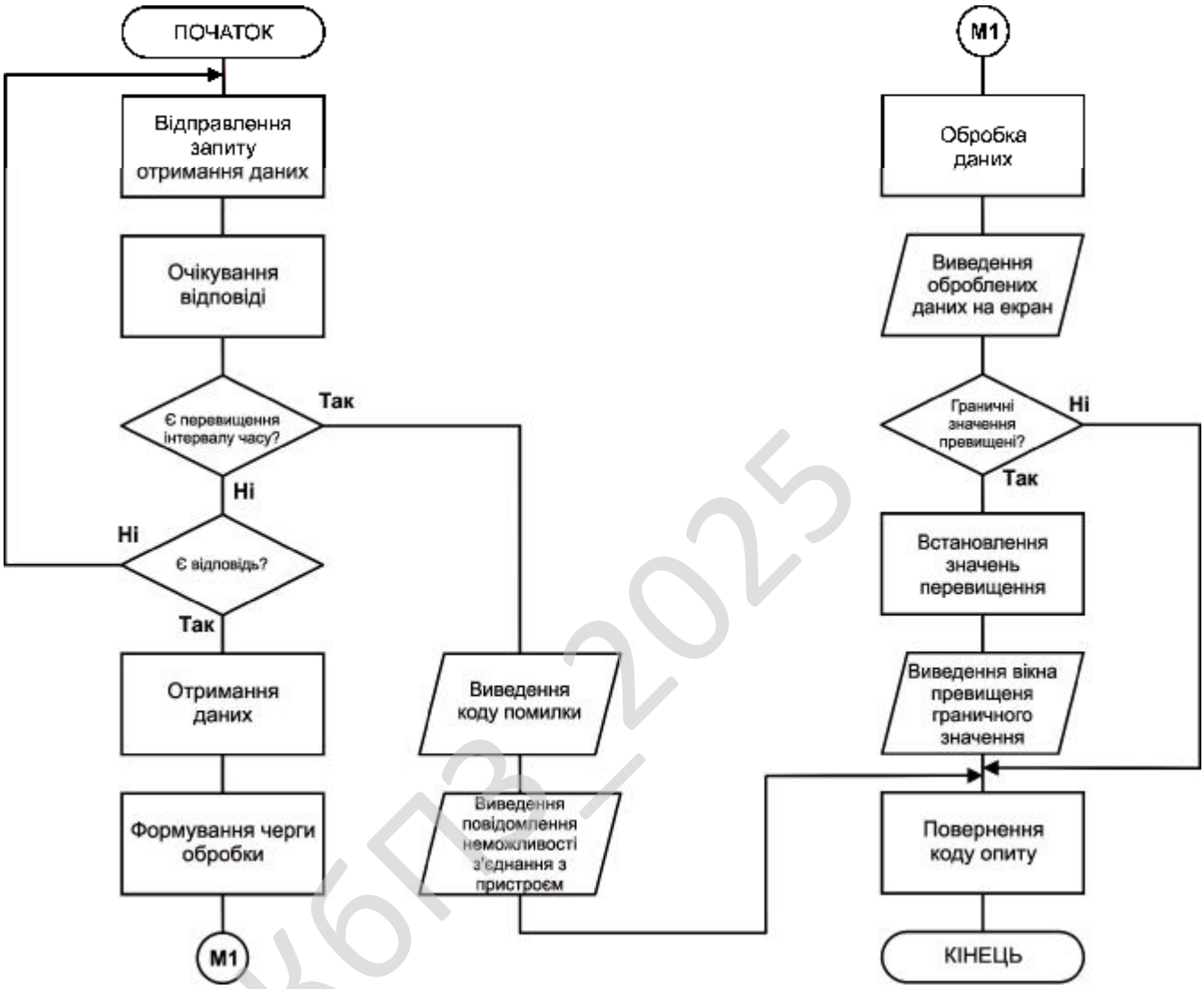

Рисунок 4.2 – Блок-схема роботи підпрограми

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.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма об'єктів; Діаграма розгортання.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

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.

При розробці й взаємодії програмної частини з апаратної виникли труднощі взаємодії зовнішнього АЦП/ЦАП з персональним комп'ютером тому що підключення до нього відбувається через спеціальний порт RS485.

Для рішення цієї проблеми й перетворення сигналу використовувався інтерфейсний модуль перетворення з взаємодією ARM STM32F429.

Опис системи

Система автоматичного керування сушаркою на основі ARM STM32F429 використовує мікроконтролер STM32F429 для збору даних від датчиків, обробки інформації та керування виконавчими механізмами. Архітектура включає кілька основних модулів, які взаємодіють між собою.

Основні модулі системи

1. Модуль збору даних використовує датчики температури, вологості та потоку повітря. Дані зчитуються через інтерфейси I2C або SPI, що забезпечує швидку передачу інформації.

2. Модуль обробки даних виконує аналіз отриманих значень, нормалізацію та визначення необхідного режиму роботи сушарки. Реалізує алгоритм PID-регулювання для підтримки оптимальних параметрів.

3. Модуль керування виконавчими механізмами включає електронні реле та ШІМ-регулятори для управління нагрівальними елементами, вентиляторами та системою подачі повітря. Використовує GPIO та PWM виходи STM32F429.

4. Комунікаційний модуль забезпечує зв’язок з користувачем через дисплей TFT-LCD та сенсорний інтерфейс. Передбачена можливість підключення до Wi-Fi для віддаленого керування.

5. Модуль живлення включає стабілізатори напруги та перетворювачі, що забезпечують надійне живлення всіх компонентів системи.

Принцип роботи системи

Датчики реєструють параметри середовища, після чого мікроконтролер аналізує отримані дані та визначає необхідні дії для підтримки заданих умов. Система використовує PID-регулятор, що коригує потужність нагрівальних елементів та швидкість вентиляторів для забезпечення оптимального процесу сушіння.

Опис алгоритму керування

1. Система зчитує поточні значення температури та вологості за допомогою датчиків.

2. Отримані дані передаються у модуль обробки, де відбувається нормалізація значень.

3.PID-регулятор аналізує різницю між поточним значенням температури та заданим рівнем.

4. На основі розрахунків PID-регулятора система визначає необхідну потужність нагрівального елемента.

5. Контролер керує нагрівальним елементом через ШІМ для точного регулювання температури.

6. Система циклічно перевіряє значення датчиків та коригує параметри, забезпечуючи стабільність процесу.

7. У разі відхилень від норми система може перейти в аварійний режим та повідомити користувача.

Опис безпеки та відмовостійкості

1. Захист від перегріву. Вбудовані температурні датчики контролюють рівень нагріву. У разі перевищення безпечного порогу система автоматично відключає нагрівальні елементи.

2. Виявлення збоїв датчиків. Система періодично перевіряє коректність роботи датчиків. Якщо датчик виходить з ладу або передає некоректні дані, активується аварійний режим з відключенням нагріву.

3. Аварійне вимкнення. У разі критичних відхилень від заданих параметрів система відключає живлення нагрівальних елементів і сповіщає користувача.

4. Живлення та резервування. Передбачено стабілізоване живлення, що захищає від стрибків напруги. Для безперервної роботи можна підключити резервне джерело енергії (акумулятор).

5. Логування подій. Всі критичні події та відхилення реєструються в пам’яті системи або передаються через Wi-Fi для аналізу та профілактичного обслуговування.

Розрахунки та обґрунтування системи

Використання мікроконтролера STM32F429 обумовлено його високою продуктивністю та підтримкою апаратного прискорення обчислень, що зменшує затримки при обробці сигналів. PID-регулювання дозволяє досягти високої стабільності процесу сушіння.

Оптимальна частота ШІМ для нагрівальних елементів становить 1 кГц, що забезпечує ефективне управління температурою без значних втрат енергії. Система живлення включає стабілізатори на 3.3В та 5В, що необхідно для роботи мікроконтролера та периферії.

Завдяки модульній архітектурі система легко масштабується та адаптується під різні типи сушильних установок. Використання Wi-Fi дозволяє дистанційно контролювати процес сушіння, що підвищує зручність експлуатації.

Як висновок проектована система забезпечує автоматизоване керування процесом сушіння з використанням сучасних методів регулювання та збору даних. Використання STM32F429 гарантує високу продуктивність та енергоефективність, що робить систему надійною та гнучкою в експлуатації.

Розглянемо частину коду:

```
import time
import random
class Sensor:
    def __init__(self, sensor_type):
        self.sensor_type = sensor_type
    def read_value(self):
        return round(random.uniform(20.0, 100.0), 2)

class PIDController:
    def __init__(self, kp, ki, kd):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.prev_error = 0
        self.integral = 0
    def compute(self, setpoint, measured_value):
        error = setpoint - measured_value
        self.integral += error
        derivative = error - self.prev_error
        output = self.kp * error + self.ki * self.integral + self.kd *
derivative
        self.prev_error = error
        return max(0, min(100, output))
class Dryer:
    def __init__(self):
        self.temperature_sensor = Sensor("temperature")
        self.humidity_sensor = Sensor("humidity")
        self.pid_controller = PIDController(1.2, 0.01, 0.5)
        self.heater_power = 0
    def control_loop(self, setpoint):
        while True:
```

```
                    temperature = self.temperature_sensor.read_value()
                    humidity = self.humidity_sensor.read_value()
                    self.heater_power    =    self.pid_controller.compute(setpoint,
temperature)
                    print(f"Temperature:  {temperature}C,  Humidity:  {humidity}%,
Heater Power: {self.heater_power}%")
                    time.sleep(2)
        if __name__ == "__main__":
            dryer = Dryer()
            dryer.control_loop(50.0)
```

Клас Sensor для зчитування значень датчиків температури і вологості.

Клас PIDController для розрахунку необхідної потужності нагрівального елемента.

Клас Dryer, який використовує датчики та PID-регулятор для керування сушаркою.

Функцію control_loop, яка імітує цикл керування з періодичним зчитуванням даних та коригуванням нагріву.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Blowfish, який є симетричним алгоритмом шифрування, тобто таким, у якому ключ шифрування дорівнює ключу дешифрування. Він є мережею Фейштеля, у якій кількість ітерацій дорівнює 16. Довжина блоку дорівнює 64 бітам, ключ може мати будь-яку довжину в межах 448 біт. Хоча перед початком будь-якого шифрування виконується складна фаза ініціалізації, саме шифрування даних виконується досить швидко.

Алгоритм призначений в основному для додатків, у яких ключ міняється нечасто, до того ж існує фаза початкового рукостискання, під час якої відбувається автентифікація сторін і узгодження загальних параметрів і секретів. При реалізації на 32-бітних мікропроцесорах з більшим кешем даних Blowfish значно швидше DES.

Алгоритм складається із двох частин: розширення ключа й шифрування даних. Розширення ключа перетворює ключ довжиною, принаймні, 448 біт у кілька масивів підключів загальною довжиною 4168 байт.

В основі алгоритму лежить мережа Фейштеля з 16 ітераціями. Кожна ітерація складається з перестановки, що залежить від ключа, і підстановки, що залежить від ключа й даних. Операціями є XOR і додавання 32-бітних слів.

Blowfish використовує велику кількість підключів. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або дешифрування даних. Елементи алгоритму:

1. P – масив, що складається з вісімнадцяти 32-бітних підключів:

$$P_1, P_2, ..., P_{18}.$$

2. Чотири 32-бітних S-boxes с 256 входами кожний. Перший індекс означає номер S-box, другий індекс – номер входу.

$$S_{1,0}, S_{1,1}, \ldots S_{1,255};$$
$$S_{2,0}, S_{2,1}, \ldots S_{2,255};$$
$$S_{3,0}, S_{3,1}, \ldots S_{3,255};$$
$$S_{4,0}, S_{4,1}, \ldots S_{4,255};$$

Шифрування

Входом є 64-бітний елемент даних X, що ділиться на дві 32-бітні половини, X_l і X_r.

$$X_\text{l} = X_\text{l} \text{ XOR } P_\text{i}$$
$$X_\text{r} = F(X_\text{l}) \text{ XOR } X_\text{r}$$
$$\text{Swap } X_\text{l} \text{ and } X_\text{r}$$

Функція F

Розділити X_l на чотири 8-бітних елементи A, B, C, D.

$F(X_\text{l}) = ((S_{1,A} + S_{2,B} \bmod 2^{32}) \text{ XOR } S_{3,C}) + S_{4,D} \bmod 2^{32}$

Дешифрування відрізняється від шифрування тим, що P_i використовуються у зворотному порядку.

Генерація підключів

Підключи обчислюються з використанням самого алгоритму Blowfish.

1. Ініціалізувати перший *P*-масив і чотири *S*-boxes фіксовані рядки.

2. Виконати операцію XOR P_1 з першими 32 бітами ключа, операцію XOR P_2 із другими 32 бітами ключа й т.д. Повторювати цикл доти, поки весь *P*-масив не буде побітово складний з усіма бітами ключа. Для коротких ключів виконується конкатенація ключа із самим собою.

3. Зашифрувати нульовий рядок алгоритмом Blowfish, використовуючи підключи, описані в пунктах (1) і (2).

4. Замінити P_1 і P_2 виходом, отриманим на кроці (3).

5. Зашифрувати вихід кроку (3), використовуючи алгоритм Blowfish з модифікованими підключами.

6. Замінити P_3 і P_4 виходом, отриманим на кроці (5).

7. Продовжити процес, заміняючи всі елементи *P*-масиву, а потім всі чотири *S*-boxes, виходами відповідним чином модифікованого алгоритму Blowfish.

Для створення всіх підключів потрібна 521 ітерація.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429 складається з наступних функціональних блоків:

– Блоку інформаційно-навігаційного відображення.

– Блоку функцій представлених у графічному вигляді.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

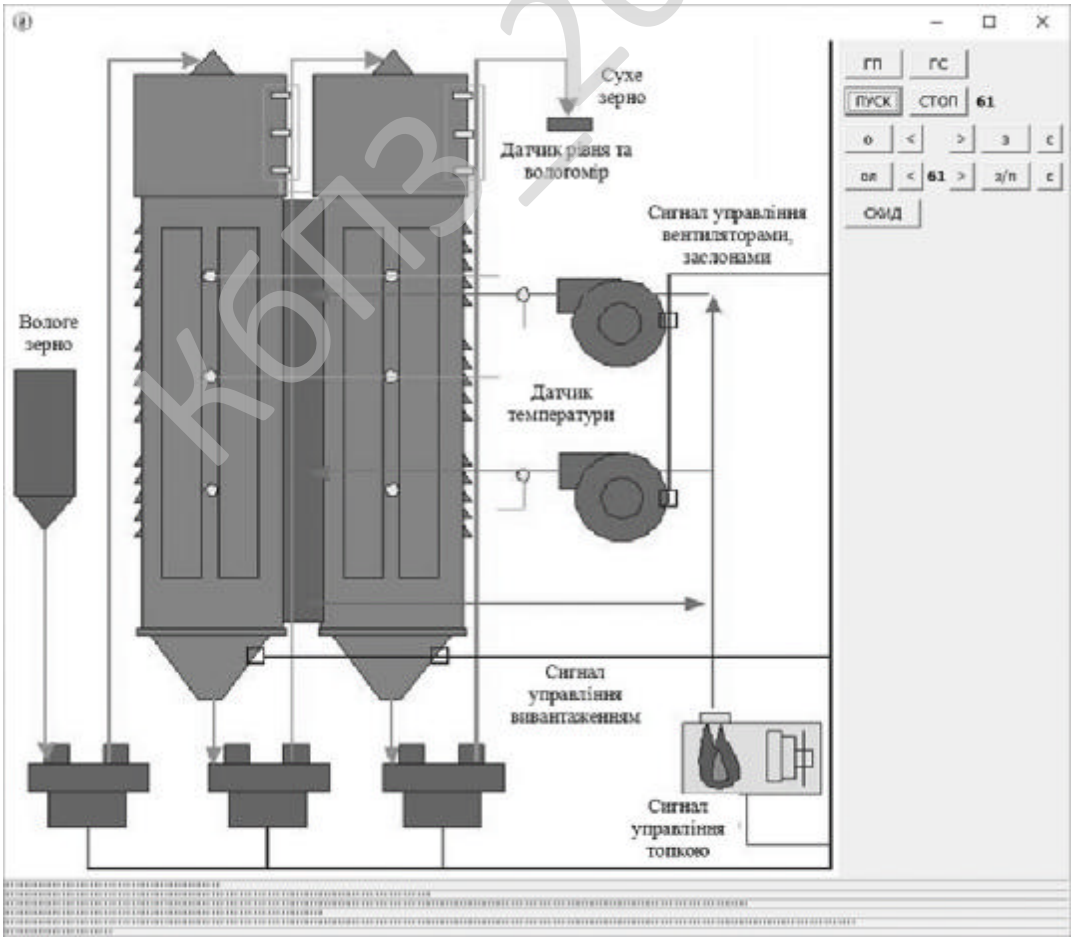

Рисунок 5.1 – Головне вікно розробленого ПЗ

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.

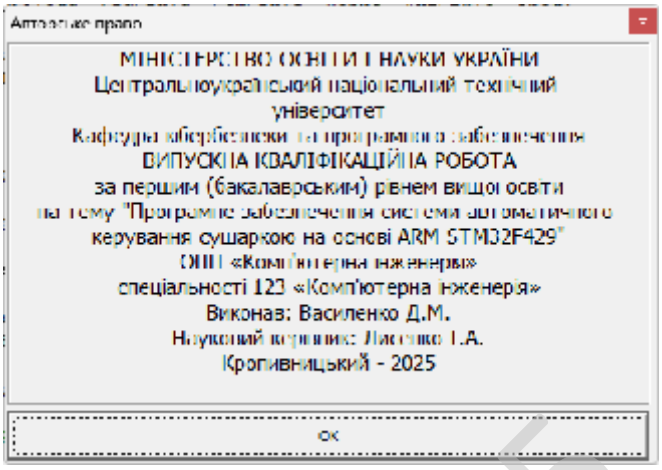

Рисунок 5.2 – Вікно розробника ПЗ

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 – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи автоматичного керування сушаркою на основі ARM STM32F429.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем автоматичного керування сушаркою на основі ARM STM32F429.

– Досліджена система автоматичного керування сушаркою на основі ARM STM32F429.

– На основі отриманих результатів досліджень створена програмна реалізація системи автоматичного керування сушаркою на основі ARM STM32F429.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання автоматичного керування сушаркою на основі ARM STM32F429.

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 призначені для системи автоматичного керування сушаркою на основі ARM STM32F429. 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 Blowfish.

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.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

2. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

3. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

4. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

5. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

6. Kuznetsov O., Ilchenko O., Kryvinska N., Buravchenko K., Smirnov O., Savchenko Iu. «An Empirical Assessment of Leading Blockchain Financial Services». *2023 IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF),* Kyiv, Ukraine, 2023, pp. 1-6,

7. Smirnov O., Fedorov E., Neskorodieva A., Neskorodieva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

8. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

9. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems,* 2023, 729 LNNS, pp. 104–112.

10. 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.

11. 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.

12. 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.

13. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

14. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

15. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

16. 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.

17. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

19. 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.

20. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

21. 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.

22. 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.

23. 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.

24. 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.

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. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

27. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

28. 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.

29. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк , О., Смірнова , Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об'єкту критичної інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 6-26.

30. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

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. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

33. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

34. 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.

35. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

36. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

37. 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.

38. 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.

39. 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.

40. 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).

41. 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.

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

43. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

44. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

45. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції “Інформаційна безпека та комп’ютерні технології”, м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

46. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces “Alexandru cel Bun”. 2017. P7.

47. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA - 2017.

48. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

49. Смірнов О.А., Смірнов С.А., Рябой Д.К., Рябая О.В. Модель вузла комутації з відносними пріоритетами, резервуванням ресурсів і обліком реальної надійності обслуговуючих приладів .Збірник тез всеукраїнської науково-практичної інтернет-конференції «Автоматика та комп’ютерно-інтегровані технології у промисловості, телекомунікаціях, енергетиці та транспорті». м. Кропивницький. 16-17 листопада 2017 р. – Кропивницький: ЦНТУ. – 2017. – С. 198-199.

50. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи автоматичного керування сушаркою на основі ARM STM32F429.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи автоматичного керування сушаркою на основі ARM STM32F429;

– 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 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 53 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 2.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Лисенко І.А.

Програмне забезпечення системи автоматичного керування сушаркою на основі ARM STM32F429

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import time
import random
import logging
import threading

# Глобальні константи для системи
MAX_TEMPERATURE = 120.0
# Максимально допустима температура
MIN_TEMPERATURE = 20.0
# Мінімально допустима температура для початку сушіння
TARGET_TEMPERATURE = 80.0
# Цільова температура сушіння
HUMIDITY_THRESHOLD = 50.0
# Поріг вологості для визначення режиму сушіння

# Клас PID-регулятора для керування температурою
class PIDController:
    def __init__(self, kp, ki, kd):
        # Ініціалізація коефіцієнтів PID-регулятора
        self.kp = kp
        self.ki = ki
        self.kd = kd
        # Ініціалізація попередньої помилки
        self.previous_error = 0.0
        # Ініціалізація інтегрального значення
        self.integral = 0.0

    def compute(self, setpoint, measurement):
        # Розрахунок помилки між встановленою і виміряною температурою
        error = setpoint - measurement
        # Накопичення інтегральної помилки
        self.integral += error
        # Розрахунок похідної від помилки
        derivative = error - self.previous_error
        # Обчислення вихідного сигналу за формулою PID
        output = self.kp * error + self.ki * self.integral + self.kd *
derivative
        # Оновлення попередньої помилки
        self.previous_error = error
        # Повернення розрахованого вихідного сигналу
        return output

# Клас симуляції датчиків системи
class SensorSimulator:
    def __init__(self):
        # Ініціалізація датчика температури випадковим значенням
        self.temperature = random.uniform(25.0, 30.0)
        # Ініціалізація датчика вологості випадковим значенням
        self.humidity = random.uniform(40.0, 60.0)
        # Ініціалізація датчика тиску випадковим значенням
        self.pressure = random.uniform(990.0, 1020.0)
        # Прапорець відмови датчиків
        self.fault = False

    def update_sensors(self):
        # Оновлення температури з невеликим випадковим коливанням
        self.temperature += random.uniform(-0.5, 0.5)
        # Оновлення вологості з невеликим випадковим коливанням
        self.humidity += random.uniform(-0.5, 0.5)
        # Оновлення тиску з невеликим випадковим коливанням
        self.pressure += random.uniform(-0.2, 0.2)
        # Імітація випадкової відмови датчиків
        if random.randint(0, 1000) > 995:
            self.fault = True
        # Повернення оновлених значень датчиків
        return self.temperature, self.humidity, self.pressure
```

```
    def get_temperature(self):
        # Повернення поточного значення температури
        return self.temperature

    def get_humidity(self):
        # Повернення поточного значення вологості
        return self.humidity

    def get_pressure(self):
        # Повернення поточного значення тиску
        return self.pressure

# Клас симуляції виконавчих пристроїв системи
class ActuatorSimulator:
    def __init__(self):
        # Ініціалізація стану нагрівача
        self.heater_state = False
        # Ініціалізація стану вентилятора
        self.fan_state = False
        # Ініціалізація стану двигуна/механізму
        self.motor_state = False

    def control_heater(self, state):
        # Встановлення стану нагрівача
        self.heater_state = state

    def control_fan(self, state):
        # Встановлення стану вентилятора
        self.fan_state = state

    def control_motor(self, state):
        # Встановлення стану двигуна/механізму
        self.motor_state = state

# Клас контролера сушарки, який керує процесом сушіння
class DryerController:
    def __init__(self):
        # Створення екземпляру симулятора датчиків
        self.sensor = SensorSimulator()
        # Створення екземпляру симулятора виконавчих пристроїв
        self.actuator = ActuatorSimulator()
        # Створення екземпляру PID-регулятора з параметрами
        self.pid = PIDController(1.2, 0.01, 0.05)
        # Встановлення цільової температури для процесу сушіння
        self.target_temperature = TARGET_TEMPERATURE
        # Встановлення початкового стану системи
        self.system_state = "IDLE"
        # Ініціалізація прапорця помилки
        self.error_state = False
        # Ініціалізація системного журналу подій
        self.log = []
        # Ініціалізація лічильника циклів
        self.cycle_count = 0
        # Прапорець необхідності обслуговування
        self.maintenance_required = False

    def update(self):
        # Оновлення показників датчиків
        temp, hum, pres = self.sensor.update_sensors()
        # Запис показників датчиків до журналу
        self.log.append("Cycle " + str(self.cycle_count) + ": Temperature=" +
"{:.2f}".format(temp) + ", Humidity=" + "{:.2f}".format(hum) + ", Pressure=" +
"{:.2f}".format(pres))
        # Перевірка на відмову датчиків та фіксація події
        if self.sensor.fault:
            self.log.append("Sensor fault detected.")
            self.error_state = True
```

```
        # Розрахунок керуючого сигналу за допомогою PID-регулятора
        control_signal = self.pid.compute(self.target_temperature, temp)
        # Прийняття рішення щодо керування виконавчими пристроями
        if control_signal > 0:
            self.actuator.control_heater(True)
            self.actuator.control_fan(False)
            self.log.append("Heater ON, Fan OFF. Control signal: " +
"{:.2f}".format(control_signal))
        else:
            self.actuator.control_heater(False)
            self.actuator.control_fan(True)
            self.log.append("Heater OFF, Fan ON. Control signal: " +
"{:.2f}".format(control_signal))
        # Оцінка стану системи за отриманими даними
        self.evaluate_state(temp, hum)
        # Збільшення лічильника циклів
        self.cycle_count += 1

    def evaluate_state(self, temp, hum):
        # Перевірка, чи не виходить температура за допустимі межі
        if temp > MAX_TEMPERATURE or temp < MIN_TEMPERATURE:
            self.system_state = "ERROR"
            self.error_state = True
            self.log.append("Temperature out of safe range: " +
"{:.2f}".format(temp))
        else:
            # Визначення режиму роботи залежно від рівня вологості
            if hum > HUMIDITY_THRESHOLD:
                self.system_state = "DRYING"
            else:
                self.system_state = "COOLING"
            self.log.append("System state updated to: " + self.system_state)

    def run(self):
        # Виконання циклічного керування протягом фіксованої кількості циклів
        for i in range(150):
            self.update()
            time.sleep(0.1)
            # Перевірка на наявність помилок
            if self.error_state:
                self.log.append("Error state encountered, breaking control
loop.")
                break
            # Планове обслуговування кожні 50 циклів
            if self.cycle_count % 50 == 0 and self.cycle_count != 0:
                self.perform_maintenance()
        # Завершення роботи системи після циклів
        self.shutdown()

    def shutdown(self):
        # Безпечне відключення всіх виконавчих пристроїв
        self.actuator.control_heater(False)
        self.actuator.control_fan(False)
        self.actuator.control_motor(False)
        self.log.append("System shutdown completed.")

    def perform_maintenance(self):
        # Імітація операцій з обслуговування системи
        self.log.append("Performing scheduled maintenance.")
        time.sleep(0.2)
        self.maintenance_required = False
        self.log.append("Maintenance completed successfully.")

    def reset(self):
        # Скидання параметрів і стану системи
        self.log.append("System reset initiated.")
```

```
        self.cycle_count = 0
        self.error_state = False
        self.system_state = "IDLE"
        self.sensor.fault = False
        self.pid.previous_error = 0.0
        self.pid.integral = 0.0
        self.log.append("System reset completed.")

    def get_log(self):
        # Повернення журналу системних подій
        return self.log

# Функція для калібрування системи
def calibrate_system(controller):
    # Початок процесу калібрування датчиків
    controller.log.append("Calibration process started.")
    for step in range(10):
        # Корекція показників температури під час калібрування
        controller.sensor.temperature += random.uniform(-0.2, 0.2)
        # Корекція показників вологості під час калібрування
        controller.sensor.humidity += random.uniform(-0.2, 0.2)
        # Запис інформації про крок калібрування
        controller.log.append("Calibration step " + str(step+1) + " completed.")
        time.sleep(0.05)
    controller.log.append("Calibration process completed.")

# Функція для проведення діагностики системи
def run_diagnostics(controller):
    # Запис початку діагностики в журнал
    controller.log.append("Diagnostics started.")
    for check in range(5):
        # Діагностична перевірка датчика температури
        if controller.sensor.get_temperature() < MIN_TEMPERATURE or
controller.sensor.get_temperature() > MAX_TEMPERATURE:
            controller.log.append("Temperature sensor diagnostic failed at check
" + str(check+1) + ".")
            controller.error_state = True
        else:
            controller.log.append("Temperature sensor diagnostic passed at check
" + str(check+1) + ".")
        # Діагностична перевірка датчика вологості
        if controller.sensor.get_humidity() < 0 or
controller.sensor.get_humidity() > 100:
            controller.log.append("Humidity sensor diagnostic failed at check "
+ str(check+1) + ".")
            controller.error_state = True
        else:
            controller.log.append("Humidity sensor diagnostic passed at check "
+ str(check+1) + ".")
        time.sleep(0.1)
    controller.log.append("Diagnostics completed.")
    return not controller.error_state

# Функція для проведення технічного огляду системи
def maintenance_check(controller):
    # Запис початку технічного огляду в журнал
    controller.log.append("Maintenance check started.")
    for phase in range(3):
        # Виконання етапу технічного огляду
        controller.log.append("Maintenance check phase " + str(phase+1) + "
completed.")
        time.sleep(0.1)
    controller.log.append("Maintenance check completed.")
    return True
```

```
# Функція для імітації зовнішнього мережевого запиту параметрів системи
def network_query():
    # Імітація затримки мережевого запиту
    time.sleep(0.05)
    # Повернення випадкових зовнішніх параметрів
    external_parameters = {
        "ambient_temperature": random.uniform(15.0, 35.0),
        "wind_speed": random.uniform(0.0, 15.0)
    }
    return external_parameters

# Функція для оновлення налаштувань системи з отриманих мережевих даних
def update_settings_from_network(controller):
    # Виконання мережевого запиту
    params = network_query()
    # Запис результату мережевого запиту в журнал
    controller.log.append("External network query result: " + str(params))
    # Оновлення цільової температури залежно від зовнішніх умов
    ambient_temp = params.get("ambient_temperature", TARGET_TEMPERATURE)
    if ambient_temp < TARGET_TEMPERATURE:
        controller.target_temperature = TARGET_TEMPERATURE - (TARGET_TEMPERATURE
- ambient_temp) * 0.1
    else:
        controller.target_temperature = TARGET_TEMPERATURE
    # Запис оновленої цільової температури
    controller.log.append("Updated target temperature: " +
"{:.2f}".format(controller.target_temperature))
    time.sleep(0.05)

# Функція для розширеного журналювання подій
def extended_logging(controller):
    # Генерація детальних записів журналу для моніторингу системи
    for entry in range(20):
        # Запис детальної інформації з номером циклу та поточним станом
        controller.log.append("Extended log entry " + str(entry+1) + ": Cycle "
+ str(controller.cycle_count) + ", State: " + controller.system_state)
        time.sleep(0.03)

# Функція для резервного копіювання стану системи у файл
def backup_system_state(controller):
    # Спроба резервного копіювання журналу системи у файл
    try:
        # Відкриття файлу для запису резервної копії
        with open("system_backup.log", "w") as backup_file:
            for log_entry in controller.get_log():
                backup_file.write(log_entry + "\n")

        # Запис успішного резервного копіювання
        controller.log.append("System state backup completed successfully.")
    except Exception as error:

        # Запис невдалого резервного копіювання з деталями помилки
        controller.log.append("System state backup failed: " + str(error))
    time.sleep(0.05)

# Функція для відновлення стану системи з файлу резервної копії
def restore_system_state(controller):
    # Спроба відновлення журналу системи з файлу резервної копії
    try:
        with open("system_backup.log", "r") as backup_file:
            restored_logs = backup_file.readlines()
        # Запис кількості відновлених записів журналу
        controller.log.append("System state restored with " +
str(len(restored_logs)) + " log entries.")
    except Exception as error:
```

```
            # Запис невдалого відновлення з деталями помилки
            controller.log.append("System state restoration failed: " + str(error))
        time.sleep(0.05)

# Функція для імітації процедури аварійного відключення системи
def emergency_shutdown(controller):
    # Запис початку процедури аварійного відключення
    controller.log.append("Emergency shutdown procedure initiated.")
    # Негайне відключення всіх виконавчих пристроїв
    controller.actuator.control_heater(False)
    controller.actuator.control_fan(False)
    controller.actuator.control_motor(False)
    # Встановлення прапорця помилки
    controller.error_state = True
    # Запис завершення процедури аварійного відключення
    controller.log.append("Emergency shutdown completed.")
    time.sleep(0.05)

# Функція для виконання повного циклу роботи системи, включаючи оновлення
налаштувань та журналювання
def full_system_cycle(controller):
    # Запис початку повного циклу роботи системи
    controller.log.append("Full system cycle started.")
    # Оновлення показників датчиків та керування пристроями
    controller.update()
    # Оновлення налаштувань системи з мережевих даних
    update_settings_from_network(controller)
    # Виконання розширеного журналювання для поточного циклу
    extended_logging(controller)
    # Проведення технічного огляду, якщо лічильник циклів кратний 30
    if controller.cycle_count % 30 == 0 and controller.cycle_count != 0:
        maintenance_check(controller)
    # Резервне копіювання стану системи наприкінці циклу
    backup_system_state(controller)
    # Запис завершення повного циклу роботи
    controller.log.append("Full system cycle completed.")
    time.sleep(0.05)

# Функція для послідовного виконання декількох циклів роботи системи
def run_repeated_cycles(controller, num_cycles):
    # Запуск декількох повних циклів роботи системи
    for cycle in range(num_cycles):
        full_system_cycle(controller)
        time.sleep(0.1)

# Головна функція для запуску симуляції автоматичного керування сушаркою
def main():
    # Створення екземпляру контролера сушарки
    controller = DryerController()

    # Запис завершення ініціалізації системи
    controller.log.append("Dryer control system initialization complete.")
    # Проведення калібрування системи
    calibrate_system(controller)

    # Виконання початкової діагностики системи та запис результату
    if run_diagnostics(controller):
        controller.log.append("Initial diagnostics passed.")
    else:
        controller.log.append("Initial diagnostics failed.")
    # Проведення технічного огляду системи
    maintenance_check(controller)

    # Запуск окремого потоку для виконання повторних циклів роботи системи
    cycle_thread = threading.Thread(target=run_repeated_cycles,
args=(controller, 30))
    cycle_thread.start()
```

```
    # Запуск окремого потоку для розширеного журналювання подій
    log_thread = threading.Thread(target=extended_logging, args=(controller,))
    log_thread.start()

    # Очікування завершення виконання обох потоків
    cycle_thread.join()
    log_thread.join()

    # Виконання додаткового оновлення налаштувань системи з мережевих даних
    update_settings_from_network(controller)

    # Резервне копіювання стану системи після циклів роботи
    backup_system_state(controller)

   # Відновлення стану системи для перевірки цілісності
    restore_system_state(controller)

    # Імітація аварійного відключення системи за умов випадкового вибору
    if random.choice([True, False]):
        emergency_shutdown(controller)

    # Грейсфул завершення роботи системи
    controller.shutdown()

    # Виведення всіх записів журналу на консоль
    for entry in controller.get_log():
        print(entry)

# Точка входу в програму
if __name__ == "__main__":
    main()
```

Файл GraphicalDataVisualizationLibrary.py

```
import matplotlib.pyplot as plt
import numpy as np
import random
import time
import math
import logging
import statistics

class GraphicalDataVisualizationLibrary:
    def __init__(self):
        self.figures = []
    def plot_line_graph(self, x, y, title="Line Graph"):
        plt.figure()
        plt.plot(x, y, marker='o')
        plt.title(title)
        plt.xlabel("X-axis")
        plt.ylabel("Y-axis")
        self.figures.append(plt.gcf())
        plt.close()
    def plot_scatter_graph(self, x, y, title="Scatter Graph"):
        plt.figure()
        plt.scatter(x, y)
        plt.title(title)
        plt.xlabel("X-axis")
        plt.ylabel("Y-axis")
        self.figures.append(plt.gcf())
        plt.close()
    def plot_bar_chart(self, categories, values, title="Bar Chart"):
        plt.figure()
        plt.bar(categories, values)
        plt.title(title)
        plt.xlabel("Categories")
        plt.ylabel("Values")
        self.figures.append(plt.gcf())
        plt.close()
    def generate_multiple_graphs(self, data_dict):
        for key in data_dict:
            values = data_dict[key]
            x = list(range(len(values)))
            self.plot_line_graph(x, values, title=key + " Line Graph")
            self.plot_scatter_graph(x, values, title=key + " Scatter Graph")
            categories = [str(i) for i in x]
            self.plot_bar_chart(categories, values, title=key + " Bar Chart")
    def get_all_figures_count(self):
        return len(self.figures)

class ArtificialIntelligenceDiagnosticModule:
    def __init__(self):
        self.model_weights = [random.random() for _ in range(10)]
    def preprocess_data(self, data):
        norm_data = []
```

```
        min_val = min(data)
        max_val = max(data)
        range_val = max_val - min_val + 1e-6
        for d in data:
            norm_data.append((d - min_val) / range_val)
        return norm_data
    def run_inference(self, processed_data):
        score = 0
        for i in range(len(processed_data)):
            score += processed_data[i] * self.model_weights[i %
len(self.model_weights)]
        return score
    def analyze_sensor_data(self, sensor_data):
        processed = self.preprocess_data(sensor_data)
        inference_score = self.run_inference(processed)
        if inference_score > 0.7:
            result = "Optimal"
        elif inference_score > 0.4:
            result = "Warning"
        else:
            result = "Critical"
        return {"inference_score": inference_score, "diagnosis": result}
    def generate_random_sensor_data(self, size=50):
        return [random.uniform(0, 100) for _ in range(size)]
    def run_diagnostic_cycle(self):
        data = self.generate_random_sensor_data()
        result = self.analyze_sensor_data(data)
        return result

class DetailedLogAnalysisFramework:
    def __init__(self, logs):
        self.logs = logs
    def count_occurrences(self, keyword):
        count = 0
        for log in self.logs:
            if keyword in log:
                count += 1
        return count
    def get_error_logs(self):
        error_logs = []
        for log in self.logs:
            if "error" in log.lower():
                error_logs.append(log)
        return error_logs
    def get_warning_logs(self):
        warning_logs = []
        for log in self.logs:
            if "warning" in log.lower():
                warning_logs.append(log)
        return warning_logs
    def analyze_log_statistics(self):
        total = len(self.logs)
        error_count = self.count_occurrences("error")
```

```
        warning_count = self.count_occurrences("warning")
        info_count = total - error_count - warning_count
        return {"total": total, "errors": error_count, "warnings":
warning_count, "info": info_count}
    def generate_report(self):
        stats = self.analyze_log_statistics()
        report = "Total logs: " + str(stats["total"]) + "\n"
        report += "Errors: " + str(stats["errors"]) + "\n"
        report += "Warnings: " + str(stats["warnings"]) + "\n"
        report += "Info: " + str(stats["info"]) + "\n"
        detailed_errors = "\n".join(self.get_error_logs())
        report += "Detailed Error Logs:\n" + detailed_errors
        return report
    def filter_logs_by_date(self, date_str):
        filtered = []
        for log in self.logs:
            if date_str in log:
                filtered.append(log)
        return filtered
    def sort_logs_alphabetically(self):
        return sorted(self.logs)

class ModularTestDataGeneratorLibrary:
    def generate_uniform_data(self, size, lower, upper):
        return [random.uniform(lower, upper) for _ in range(size)]
    def generate_integer_data(self, size, lower, upper):
        return [random.randint(lower, upper) for _ in range(size)]
    def generate_normal_data(self, size, mean, std):
        return [random.gauss(mean, std) for _ in range(size)]
    def generate_poisson_data(self, size, lam):
        data = []
        for _ in range(size):
            L = math.exp(-lam)
            k = 0
            p = 1
            while p > L:
                k += 1
                p *= random.random()
            data.append(k - 1)
        return data
    def generate_exponential_data(self, size, lambd):
        return [random.expovariate(lambd) for _ in range(size)]
    def generate_data_matrix(self, rows, cols, generator_function, *args):
        matrix = []
        for _ in range(rows):
            row = []
            for _ in range(cols):
                row.append(generator_function(*args))
            matrix.append(row)
        return matrix
    def generate_random_string(self, length):
        letters = "abcdefghijklmnopqrstuvwxyz"
        return "".join(random.choice(letters) for _ in range(length))
```

```
    def generate_string_list(self, size, string_length):
        return [self.generate_random_string(string_length) for _ in range(size)]
    def generate_mixed_data(self, size):
        data = []
        for i in range(size):
            if i % 3 == 0:
                data.append(random.uniform(0, 100))
            elif i % 3 == 1:
                data.append(random.randint(0, 100))
            else:
                data.append(self.generate_random_string(5))
        return data
    def generate_large_dataset(self, size):
        dataset = []
        for _ in range(size):
            entry = {"sensor1": random.uniform(0, 100), "sensor2":
random.uniform(0, 100), "status": random.choice(["OK", "FAIL", "WARN"])}
            dataset.append(entry)
        return dataset

class SystemPerformanceOptimizationToolkit:
    def benchmark_function(self, func, *args, **kwargs):
        start = time.time()
        result = func(*args, **kwargs)
        end = time.time()
        return {"result": result, "time": end - start}
    def optimize_parameters(self, function, param_range):
        best_param = None
        best_score = float('inf')
        for param in param_range:
            score = function(param)
            if score < best_score:
                best_score = score
                best_param = param
        return best_param, best_score
    def simulate_load(self, iterations):
        total = 0
        for i in range(iterations):
            total += math.sqrt(i) * math.sin(i) * math.cos(i)
        return total
    def run_benchmark_suite(self):
        benchmarks = {}
        for i in range(1, 6):
            result = self.benchmark_function(self.simulate_load, 10000 * i)
            benchmarks["load_test_" + str(i)] = result
        return benchmarks
    def tune_system(self, initial_param, adjustment_function, iterations):
        param = initial_param
        history = []
        for i in range(iterations):
            new_param = adjustment_function(param)
            score = self.simulate_load(int(new_param))
            history.append((new_param, score))
```

```
            param = new_param
        return history
    def dummy_adjustment(self, param):
        return param * random.uniform(0.95, 1.05)
    def perform_optimization_cycle(self, iterations):
        history = self.tune_system(1000, self.dummy_adjustment, iterations)
        best = min(history, key=lambda x: x[1])
        return best, history

def main():
    vis = GraphicalDataVisualizationLibrary()
    x = list(range(50))
    y = [random.uniform(0, 100) for _ in x]
    vis.plot_line_graph(x, y, "Line Graph Example")
    vis.plot_scatter_graph(x, y, "Scatter Graph Example")
    vis.plot_bar_chart([str(i) for i in x], y, "Bar Chart Example")
    data_dict = {"Dataset1": [random.uniform(0, 50) for _ in range(30)],
"Dataset2": [random.uniform(50, 100) for _ in range(30)]}
    vis.generate_multiple_graphs(data_dict)
    print("Total figures generated:", vis.get_all_figures_count())
    ai_module = ArtificialIntelligenceDiagnosticModule()
    sensor_data = ai_module.generate_random_sensor_data(100)
    diagnosis = ai_module.analyze_sensor_data(sensor_data)
    diag_cycle = ai_module.run_diagnostic_cycle()
    print("Diagnosis Result:", diagnosis)
    print("Diagnostic Cycle Result:", diag_cycle)
    logs = []
    for i in range(100):
        log_entry = "Log entry " + str(i) + " status " + random.choice(["OK",
"Warning", "Error", "Info"])
        logs.append(log_entry)
    log_analyzer = DetailedLogAnalysisFramework(logs)
    error_logs = log_analyzer.get_error_logs()
    warning_logs = log_analyzer.get_warning_logs()
    stats = log_analyzer.analyze_log_statistics()
    report = log_analyzer.generate_report()
    sorted_logs = log_analyzer.sort_logs_alphabetically()
    print("Log Stats:", stats)
    print("Report:\n", report)
    test_data_gen = ModularTestDataGeneratorLibrary()
    uniform_data = test_data_gen.generate_uniform_data(50, 10, 20)
    integer_data = test_data_gen.generate_integer_data(50, 5, 15)
    normal_data = test_data_gen.generate_normal_data(50, 0, 1)
    poisson_data = test_data_gen.generate_poisson_data(50, 3)
    exponential_data = test_data_gen.generate_exponential_data(50, 1)
    data_matrix = test_data_gen.generate_data_matrix(5, 5, random.uniform, 0,
100)
    string_list = test_data_gen.generate_string_list(20, 8)
    mixed_data = test_data_gen.generate_mixed_data(30)
    large_dataset = test_data_gen.generate_large_dataset(20)
    print("Uniform Data Sample:", uniform_data[:5])
    print("Integer Data Sample:", integer_data[:5])
    print("Normal Data Sample:", normal_data[:5])
```

```
    print("Poisson Data Sample:", poisson_data[:5])
    print("Exponential Data Sample:", exponential_data[:5])
    print("Data Matrix Sample:", data_matrix)
    print("String List Sample:", string_list)
    print("Mixed Data Sample:", mixed_data)
    print("Large Dataset Sample:", large_dataset)
    perf_toolkit = SystemPerformanceOptimizationToolkit()
    benchmark_results = perf_toolkit.run_benchmark_suite()
    best_param, best_score = perf_toolkit.optimize_parameters(lambda p:
perf_toolkit.simulate_load(int(p)), range(1000, 1100))
    optimization_history = perf_toolkit.perform_optimization_cycle(10)
    benchmark = perf_toolkit.benchmark_function(perf_toolkit.simulate_load,
50000)
    print("Benchmark Results:", benchmark_results)
    print("Best Parameter:", best_param, "Score:", best_score)
    print("Optimization History:", optimization_history)
    print("Single Benchmark:", benchmark)

if __name__ == "__main__":
    main()
```

Файл IoTRemoteMonitoringModule.py

```
import time
import random
import math
import threading

class IoTRemoteMonitoringModule:
    def __init__(self):
        self.data_queue = []
        self.server_address = "192.168.1.100"
        self.connected = False
    def connect(self):
        self.connected = True
        self.connection_attempts = 0
        while not self.connected and self.connection_attempts < 5:
            self.connection_attempts += 1
            time.sleep(0.5)
            if random.random() > 0.2:
                self.connected = True
            else:
                self.connected = False
        return self.connected
    def disconnect(self):
        self.connected = False
    def send_data(self, data):
        if self.connected:
            self.data_queue.append(data)
            time.sleep(0.1)
            return True
        else:
            return False
    def simulate_remote_monitoring(self, sensor_data):
        self.connect()
        for data in sensor_data:
            success = self.send_data(data)
            if not success:
                self.connect()
                self.send_data(data)
        self.disconnect()
        return self.data_queue
    def generate_sensor_data(self, count):
        data = []
        for i in range(count):
            data.append({"timestamp": time.time(), "value": random.uniform(0,
100), "id": i})
            time.sleep(0.01)
        return data

class ComprehensiveDiagnosticsSuite:
    def __init__(self):
        self.results = {}
```

```
    def diagnostic_cpu(self):
        values = []
        for i in range(20):
            values.append(random.uniform(0, 100))
            time.sleep(0.01)
        avg = sum(values)/len(values)
        self.results["cpu"] = avg
        return avg
    def diagnostic_memory(self):
        values = []
        for i in range(20):
            values.append(random.uniform(100, 200))
            time.sleep(0.01)
        avg = sum(values)/len(values)
        self.results["memory"] = avg
        return avg
    def diagnostic_sensors(self):
        sensor_checks = {}
        for sensor in range(5):
            sensor_checks["sensor_" + str(sensor)] = random.choice(["OK",
"FAIL", "WARN"])
            time.sleep(0.02)
        self.results["sensors"] = sensor_checks
        return sensor_checks
    def diagnostic_network(self):
        latency = []
        for i in range(10):
            latency.append(random.uniform(10, 100))
            time.sleep(0.01)
        avg_latency = sum(latency)/len(latency)
        self.results["network"] = avg_latency
        return avg_latency
    def run_full_diagnostics(self):
        self.diagnostic_cpu()
        self.diagnostic_memory()
        self.diagnostic_sensors()
        self.diagnostic_network()
        overall = 0
        for key in self.results:
            if isinstance(self.results[key], dict):
                overall += len(self.results[key])
            else:
                overall += self.results[key]
        self.results["overall"] = overall
        return self.results
    def generate_diagnostic_report(self):
        report_lines = []
        diagnostics = self.run_full_diagnostics()
        for key, value in diagnostics.items():
            report_lines.append(str(key) + ": " + str(value))
        report = "\n".join(report_lines)
        return report
```

```
class CustomizableControlAlgorithmsLibrary:
    def __init__(self):
        self.algorithms = {}
    def register_pid(self, kp, ki, kd):
        def pid_control(setpoint, measurement, prev_error, integral):
            error = setpoint - measurement
            integral_new = integral + error
            derivative = error - prev_error
            output = kp * error + ki * integral_new + kd * derivative
            return output, error, integral_new
        self.algorithms["PID"] = pid_control
        return pid_control
    def register_fuzzy_logic(self):
        def fuzzy_control(input_value):
            if input_value < 30:
                return "Low"
            elif input_value < 70:
                return "Medium"
            else:
                return "High"
        self.algorithms["Fuzzy"] = fuzzy_control
        return fuzzy_control
    def register_neural_network(self):
        def neural_control(inputs):
            weights = [random.uniform(-1, 1) for _ in range(len(inputs))]
            output = sum(i*w for i, w in zip(inputs, weights))
            return output
        self.algorithms["Neural"] = neural_control
        return neural_control
    def execute_algorithm(self, algorithm_name, *args, **kwargs):
        if algorithm_name in self.algorithms:
            return self.algorithms[algorithm_name](*args, **kwargs)
        else:
            return None
    def list_algorithms(self):
        return list(self.algorithms.keys())
    def simulate_control_cycle(self, setpoint, measurement):
        pid = self.register_pid(1.0, 0.1, 0.05)
        output, error, integral = pid(setpoint, measurement, 0, 0)
        fuzzy = self.register_fuzzy_logic()
        fuzzy_result = fuzzy(measurement)
        neural = self.register_neural_network()
        neural_result = neural([setpoint, measurement])
        return {"PID": output, "Fuzzy": fuzzy_result, "Neural": neural_result}

class DistributedControlNetworkIntegration:
    def __init__(self):
        self.nodes = {}
        self.network_status = {}
    def add_node(self, node_id, state):
        self.nodes[node_id] = state
        self.network_status[node_id] = "active"
    def remove_node(self, node_id):
```

```
            if node_id in self.nodes:
                del self.nodes[node_id]
                del self.network_status[node_id]
        def update_node_state(self, node_id, state):
            if node_id in self.nodes:
                self.nodes[node_id] = state
        def broadcast_message(self, message):
            responses = {}
            for node_id in self.nodes:
                responses[node_id] = self.receive_message(node_id, message)
            return responses
        def receive_message(self, node_id, message):
            delay = random.uniform(0.01, 0.1)
            time.sleep(delay)
            return "Node " + str(node_id) + " received: " + message
        def simulate_network_activity(self):
            for i in range(10):
                self.add_node(i, "state_" + str(i))
            responses = self.broadcast_message("Sync")
            for node_id in self.nodes:
                self.update_node_state(node_id, "updated_state_" + str(node_id))
            for i in range(5):
                node_to_remove = random.choice(list(self.nodes.keys()))
                self.remove_node(node_to_remove)
            return responses
        def get_network_status(self):
            return self.network_status
        def perform_network_diagnostics(self):
            diagnostics = {}
            for node_id in self.nodes:
                diagnostics[node_id] = {"status": self.network_status[node_id],
"state": self.nodes[node_id]}
                time.sleep(0.01)
            return diagnostics

    class MultiLayerDataRedundancySystem:
        def __init__(self):
            self.primary_storage = {}
            self.secondary_storage = {}
            self.tertiary_storage = {}
        def store_data_primary(self, key, value):
            self.primary_storage[key] = value
        def replicate_to_secondary(self):
            for key, value in self.primary_storage.items():
                self.secondary_storage[key] = value
        def replicate_to_tertiary(self):
            for key, value in self.secondary_storage.items():
                self.tertiary_storage[key] = value
        def verify_data_integrity(self):
            integrity = True
            for key in self.primary_storage:
                if key not in self.secondary_storage or key not in
    self.tertiary_storage:
```

```
                integrity = False
                break
        return integrity
    def simulate_data_backup_cycle(self, data_items):
        for key, value in data_items.items():
            self.store_data_primary(key, value)
            time.sleep(0.01)
        self.replicate_to_secondary()
        time.sleep(0.05)
        self.replicate_to_tertiary()
        time.sleep(0.05)
        return self.verify_data_integrity()
    def retrieve_data(self, key):
        if key in self.primary_storage:
            return self.primary_storage[key]
        elif key in self.secondary_storage:
            return self.secondary_storage[key]
        elif key in self.tertiary_storage:
            return self.tertiary_storage[key]
        else:
            return None
    def generate_data_items(self, count):
        data_items = {}
        for i in range(count):
            data_items["item_" + str(i)] = random.randint(0, 1000)
            time.sleep(0.005)
        return data_items
    def perform_full_backup(self):
        data_items = self.generate_data_items(50)
        result = self.simulate_data_backup_cycle(data_items)
        return result, data_items

def main():
    iot_module = IoTRemoteMonitoringModule()
    sensor_data = iot_module.generate_sensor_data(30)
    iot_results = iot_module.simulate_remote_monitoring(sensor_data)
    print("IoT Remote Monitoring Results:")
    print(iot_results)
    diagnostics_suite = ComprehensiveDiagnosticsSuite()
    diagnostics_report = diagnostics_suite.generate_diagnostic_report()
    print("Comprehensive Diagnostics Report:")
    print(diagnostics_report)
    control_lib = CustomizableControlAlgorithmsLibrary()
    control_results = control_lib.simulate_control_cycle(75, 65)
    print("Control Algorithms Results:")
    print(control_results)
    network_integration = DistributedControlNetworkIntegration()
    network_integration.simulate_network_activity()
    network_status = network_integration.get_network_status()
    diagnostics = network_integration.perform_network_diagnostics()
    print("Distributed Network Status:")
    print(network_status)
    print("Network Diagnostics:")
```

```
    print(diagnostics)
    redundancy_system = MultiLayerDataRedundancySystem()
    backup_result, data_items = redundancy_system.perform_full_backup()
    print("Multi-Layer Data Redundancy Backup Result:")
    print(backup_result)
    print("Data Items:")
    print(data_items)

if __name__ == "__main__":
    main()
```