Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, факультету)
Автоматизації та інтелектуальних інформаційних технологій
(повна назва кафедри)

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Синтез нечітких регуляторів в системах автоматичного керування»

Виконав: студент 2-го курсу, групи
1АКІТ-22М
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Іван ГНАТЮК
(ім'я та прізвище)
Керівник: д.т.н., доцент каф. АІІТ
Володимир ГАРМАШ
(ім'я та прізвище)
«4» грудня 2023 р.
Опонент: д.т.н., доцент каф. АІІТ
Володимир ОЗЕРАНСЬКИЙ
(ім'я та прізвище)
«11» грудня 2023 р.

Допущено до захисту
Завідувач кафедри АІІТ
д.т.н., проф.
Олег БІСІКАЛО
(ім'я та прізвище)
«14» грудня 2023 р.

Вінниця ВНТУ - 2023 рік

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 15 Автоматизація та приладобудування
Спеціальність 151 Автоматизація та комп'ютерно-інтегровані технології
Освітня програма Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
д.т.н., проф.
_____ Олег Бісікало
«20» вересня 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Гнатюку Івану Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи
Синтез нечітких регуляторів в системах автоматичного керування
_____
керівник роботи д.т.н., доцент каф. АІІТ Гармаш В.В.

Затверджені наказом ВНТУ від "18" вересня 2023 року № 247
2. Термін подання студентом роботи "05" грудня 2023 року
3. Вихідні дані до роботи:
Мова програмування Js, операційна система Windows, 8ГБ оперативної пам'яті, багатоядерний процесор з тактовою частотою до 4ГГЦ, 8 ядер
4. Зміст текстової частини: Вступ, Аналіз та порівння ринку аналогів ссистем, Проектування системи, Програмна реалізація веб сервісу, Тестування та дослідна експлуатація застосунку
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): Діаграма варіантів використання для програмної реалізації, діаграма варіантів використання для користувача , архітектура програмної системи, діаграма класів програмної системи, таблиці бази даних, ER діаграма бази даних, реалізація бази даних.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Володимир ГАРМАШ	21.09.2023	05.11.2023
5	Володимир КОЗЛОВСЬКИЙ	27.09.2023	01.12.2023

7. Дата видачі завдання "____" _____2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської дипломної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз та специфікації вимог системи	06.09 – 13.09.2023	Виконано
2	Проектування системи	13.09.2023 – 22.10.2023	Виконано
3	Програмна реалізація	23.10.2023 – 30.10.2023	Виконано
4	Тестування та дослідна експлуатація	31.10.2023 – 18.11.2023	Виконано
5	Оформлення пояснювальної записки	19.11.2023 – 24.11.2023	Виконано
6	Попередній захист роботи	26.11.2023	Виконано
7	Остаточний захист роботи	05.12.2023 – 14.12.2023	Виконано

Студент _____ Іван ГНАТЮК
(підпис)

Керівник роботи _____ Володимир ГАРМАШ
(підпис)

Анотація

УДК 004.45

Гнатюк І.Ю. Створення Web - застосунку нечіткого регулятора в системі автоматичного регулювання температури водонагрівача. Магістерська кваліфікаційна робота зі спеціальності 151- Автоматизація та комп'ютерно інтегровані технології, освітня програма – Інтелектуальні комп'ютерні системи. Вінниця: ВНТУ, 2023. 95 с.

На укр. мові. Бібліограф.. 31 назва; рис.: 18; таб.: 7.

Синтез нечітких регуляторів є актуальною та ефективною проблемою в сучасних системах автоматичного керування. Ця дослідницька робота присвячена розробці та оптимізації нечітких регуляторів для підвищення якості та надійності системи керування. У роботі досліджуються основні принципи нечіткої логіки та їх застосування в контексті автоматичного керування. таким чином, розглядається використання нечітких множин та правил для моделювання нечіткого регулятора. Для підвищення ефективності системи впроваджуються методи оптимізації та адаптації параметрів нечітких регуляторів. Результати експериментів підтверджують високу ефективність запропонованого підходу. Нечіткі регулятори демонструють здатність адаптуватися до змінних умов роботи системи та забезпечувати стабільність та точність керування. Отримані висновки можуть бути використані в розробці сучасних систем автоматичного керування для різноманітних виробничих процесів, таких як промислові процеси, транспортні системи та робототехніка.

Ключові слова: нечіткі регулятори, оптимізація, адаптація параметрів, автоматичне керування, ефективність системи.

Abstract

Hnatiuk I.Yu. Creating a Web - the use of a fuzzy regulator in the system of automatic temperature control of a water heater. Master's thesis on specialty 151- Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2023. 95 p.

In Ukrainian language Bibliographer.. 31 titles; Fig.: 18; tab.: 7.

The synthesis of fuzzy controllers is a relevant and effective problem in modern automatic control systems. This research work is devoted to the design and optimization of fuzzy controllers to improve the quality and reliability of the control system. The paper examines the basic principles of fuzzy logic and their application in the context of automatic control. thus, the use of fuzzy sets and rules for fuzzy controller modeling is considered. In order to improve the efficiency of the system, methods of optimization and adaptation of parameters of fuzzy regulators are introduced. The results of the experiments confirm the high efficiency of the proposed approach. Fuzzy controllers demonstrate the ability to adapt to changing system operating conditions and provide control stability and accuracy. The obtained conclusions can be used in the development of modern automatic control systems for various production processes, such as industrial processes, transport systems and robotics.

Keywords: fuzzy controllers, optimization, parameter adaptation, automatic control, system efficiency.

Зміст

ВСТУП

Актуальність дослідження нечітких регуляторів в системах автоматичного керування залишається актуальним і в ряду напрямків наукових досліджень і практичного застосування. Ці регулятори вперше були запропоновані як інструмент для вирішення проблеми управління системами, які мають нечіткі, невизначені або змінні параметри, які не можуть бути точно описані класичними математичними моделями.

Актуальність дослідження нечітких регуляторів проявляється у багатьох аспектах. По-перше, нечіткі регулятори не дозволяють моделювати системи, які мають нечіткість та невизначеність, що є реальністю для багатьох сучасних додатків. Вони також можуть адаптуватися до змін у середовищі та системі, що є критичним для управління системами зі змінними умовами.

Крім того, нечіткі регулятори не втрачають експертних знань та вимог до системи керування, що має важливе значення в багатьох галузях, включаючи медицину, промисловість та фінанси. Вони також підвищують стійкість системи до шуму та випадкових впливів, які можуть виникати в реальних умовах.

Застосування нечітких регуляторів особливо корисно для складних систем, таких як робототехніка та транспортні системи, де складно створити точні математичні моделі. Результати досліджень у цьому напрямку сприяють розробці нових методів та техніки для підвищення ефективності та надійності нечітких регуляторів, що робить цей напрямок актуальним і в галузі наукової спільноти та промисловості.

Мета удосконалення нечітких регуляторів у системах автоматичного керування полягає у покращенні ефективності та точності керування у різноманітних додатках. Нечіткі регулятори використовують нечіткі множини для моделювання та управління системами, які можуть бути складними, нелінійними та невизначеними. Ось кілька конкретних мет цього

покращення: Підвищення стійкості системи: Нечіткі регулятори можуть допомогти збільшити стійкість системи до змін параметрів, завад та зовнішніх впливів. Покращення стійкості системи дозволяє їй працювати більш надійно та ефективно.

Об’єкт дослідження – процес керування в системі автоматичного керування, де застосовуються нечіткі регулятори. дослідження щодо нечітких регуляторів у системах автоматичного керування є оптимізація нечітких ПІД-регуляторів (регуляторів зі зворотними зв'язками за пропорційним, інтегральним та диференціальним впливом). такі регулятори є важливою складовою багатьох автоматичних систем керування та широкого використання в промисловості та різних галузях.

Предметом дослідження – дослідження є методи "Оптимізації нечітких ПІД-регуляторів в системах автоматичного керування" або "Адаптація нечітких ПІД-регуляторів до системи зі змінними параметрами в автоматичному керуванні". У цьому випадку дослідження спрямоване на покращення та оптимізацію використання конкретних нечітких ПІД-регуляторів у системах автоматичного керування, з метою підвищення їх точності, швидкості та стійкості в різних умовах експлуатації.

Для досягнення поставленої мети необхідно розв'язати наступні задачі:

Для досягнення мети дослідження нечітких регуляторів у системах автоматичного керування сформульовано наступні задачі:

Розробка нечітких моделей систем: Першою задачею є розробка математичних моделей для системи, які потрібно керувати. Ці моделі повинні отримати нечіткість та невизначеність параметрів системи.

Створення нечітких регуляторів: Розробка алгоритмів нечіткого керування, які базуються на створених моделях і враховують нечіткість вхідних даних та вимог до системи.

Налаштування параметрів регуляторів: Визначення оптимальних параметрів нечітких регуляторів, які забезпечують бажану динаміку системи та забезпечують мінімізацію помилок керування.

Адаптація до змінних умов: Розробка механізмів адаптації нечітких регуляторів до змінних умов роботи системи або до зміни динаміки системи з часом.

Експерименти та валідація: Проведення експериментів і чисельних симуляцій для перевірки ефективності розроблених нечітких регуляторів у реальних або схожих на реальних умовах.

Порівняння з іншими методами керування: Порівняння результатів роботи нечітких регуляторів з іншими методами керування, такими як класичні ПІД-регулятори або інші адаптивні алгоритми.

Оптимізація та покращення результатів: Пошук можливостей для оптимізації та подальшого покращення результатів, враховуючи відгуки з попередніх етапів дослідження.

Методи дослідження. Методи дослідження нечітких регуляторів у системах автоматичного керування включають математичне моделювання та симуляцію систем, експериментальне випробування на реальних пристроях або в імітаційних середовищах, налаштування параметрів регулятора на основі оптимізаційних методів, аналіз стійкості та швидкості відгуків систем, порівняння з іншими методами керування, а також дослідження адаптивності. регулятора до змінних умов та параметрів системи. Усі ці методи спрямовані на покращення роботи нечітких регуляторів та оптимальне їх використання в різних системах автоматичного керування.

Новизна(інноваційність) одержаних результатів:

Наукова новизна отриманих результатів у дослідженні нечітких регуляторів у системах автоматичного керування досягнута в вдосконаленні та розширенні наявних знань та методів у цій області. Основні внески включають:

-розробка нових алгоритмів: Досконалення та створення нових алгоритмів нечіткого керування, які забезпечують кращу стійкість, точність та адаптивність у різних умовах.

-оптимізація налаштувань: Розробка методів оптимальної настройки параметрів нечітких регуляторів, що дозволяє досягти кращої роботи системи та скоротити час налаштування.

-дослідження адаптивності: Вивчення можливостей адаптації нечітких регуляторів до змінних умов та параметрів системи, що покращує їхню придатність до практичних застосувань.

-валідація на реальних системах: Проведення експериментів на реальних об'єктах контролю для перевірки та демонстрації ефективності нечітких регуляторів промисловості в різних сферах, включаючи транспорт, робототехніку та інші галузі.

Практичною цінністю нечітких регуляторів у системах автоматичного керування створюється в їх здатності ефективно вирішувати ряд важливих завдань та вирішувати проблеми, які важко або навіть неможливі за допомогою класичних лінійних методів керування. Основні практичні переваги використання нечітких регуляторів включають:

Адаптивність до невизначеності: Нечіткі регулятори можуть працювати в умовах, де параметри системи або вхідні дані нечіткі або невизначені. Вони в змозі адаптуватися до змін у реальному часі, що робить їх особливо корисними у змінних середовищах.

Інтеграція експертного знання: Нечіткі регулятори не можуть легко включати експертні знання та правила в процес керування. Це дозволяє виконати вимоги до системи водіння та досягти більшого рівня ефективності.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ. СИНТЕЗ НЕЧІТКИХ РЕГУЛЯТОРІВ У СИСТЕМАХ АВТОМАТИЧНОГО КЕРУВАННЯ

1.1 Синтез нечітких регуляторів у системах автоматичного керування

Зростаючий інтерес до нечітких регуляторів у сучасному автоматичному керуванні є важливим явищем, що відображає змінну парадигму в підходах до керування системами. Цей інтерес визначається кількома ключовими факторами[1].

По-перше, нечіткі регулятори стали популярними завдяки своїй здатності працювати в умовах невизначеності та нелінійності, які часто характеризують реальні системи. Вони можуть адаптуватися до змін, забезпечуючи стабільність та високу точність в умовах, де класичні лінійні методи недоцільні.

По-друге, можливість інтеграції експертних знань у вигляді нечітких правил робить ці регулятори зручними для використання у сферах, де важливий інтуїтивний підхід до управління, таких як медицина, фінанси та технічні системи[2].

По-третє, нечіткі регулятори застосовуються у різних галузях, включаючи промисловість, автомобільний сектор, робототехніку та багато інших. Їхні універсальність та ефективність роблять їх популярними у багатьох галузях технологій.

В цілому, зростаючий інтерес до нечітких регуляторів свідчить про їх важливість та перспективи у сучасному автоматичному керуванні та науковому дослідженні[3].

Застосування нечітких регуляторів у різних галузях відображає їх універсальність і значущість у вирішенні різноманітних завдань та проблем. Одним із ключових переваг нечітких регуляторів є їх здатність ефективно

керувати системами в умовах невизначеності та нелінійності, що робить їх популярними в таких галузях як:

В промисловості: Нечіткі регулятори використовують для автоматичного управління виробничими процесами, де системи можуть мати змінні параметри або невизначеність у характеристиках. Вони дозволяють забезпечити стабільність і точність виробничих ліній та машин[4].

У транспорті: Нечіткі регулятори застосовуються в автомобільній промисловості для систем керування автомобілями та в системах автопілоту. Вони допомагають у підтримці безпеки та ефективності на дорозі.

У робототехніці: Нечіткі регулятори відіграють важливу роль в керуванні роботами, особливо в роботах, які взаємодіють з навколишнім середовищем, де нелінійність та невизначеність є типовими[5].

У медицині: Застосування нечітких регуляторів можливе в системах контролю та діагностики пацієнтів, де важливою є точність та стабільність процесу лікування. Simulink-модель для дослідження нечітких регуляторів зображена на рисунку 1.1.

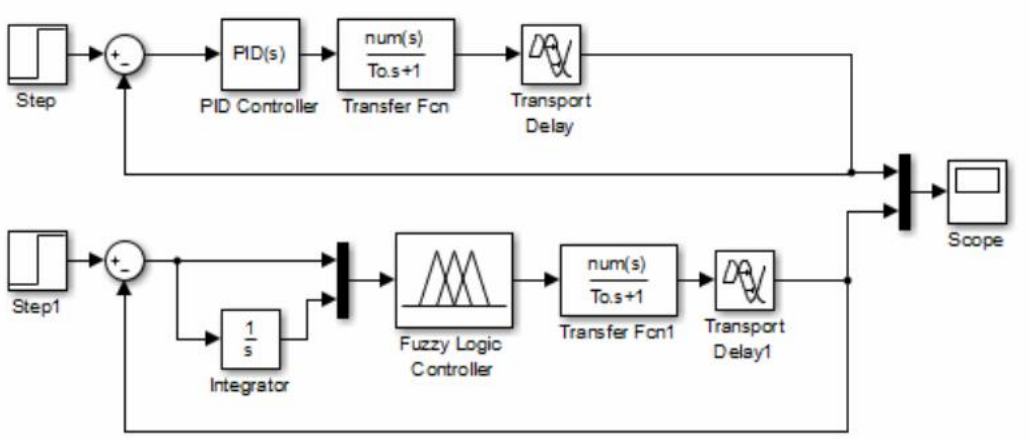

Рисунок 1.1 - Simulink-модель для дослідження нечітких регуляторів

У фінансах: Нечітки регулятори можуть бути використані для прогнозування та управління фінансовими ризиками, де ринки можуть мати нестабільну динаміку.

В екології: У системах охорони навколишнього середовища, нечіткі регулятори можуть допомагати у виконанні завдань моніторингу та збереження природи[6].

Застосування нечітких регуляторів у цих індустріях свідчить про їхнє потужне вплив і різнобічні можливості в різних сферах життя та технологій.

1.2 Автоматизоване тестування нечітких регуляторів

Автоматизоване тестування нечітких регуляторів у системах автоматичного керування є важливим етапом у їх розробці та впровадженні. Цей процес включає кілька етапів:

Планування тестування: Визначення мети тестування, обрання критеріїв ефективності та визначення обсягу та складності тестових сценаріїв. Розробка тестових сценаріїв: Створення набору тестових сценаріїв, що охоплюють різні аспекти роботи нечітких регуляторів, включаючи стійкість до змін параметрів системи, адаптивність та точність керування[7].

Створення тестових об'єктів: Симуляція або створення тестових об'єктів, на яких проводитимуться експерименти та тестування.

Виконання тестів: Запуск тестових сценаріїв та реєстрація результатів. Під час цього етапу, нечіткі регулятори піддаються різним умовам та вхідним даним для оцінки їхньої ефективності[8].

Аналsз результатів: Порівняння результатів тестування з передбачуваними критеріями ефективності та виявлення можливих проблем та недоліків.

Оптимізація та налаштування: Після аналізу результатів, регулятори можуть бути оптимізовані та налаштовані для покращення їхньої роботи.

Повторне тестування: Повторне тестування із внесеними змінами для перевірки покращень та удосконалення нечітких регуляторів.

Документація результатів: Фіксація результатів тестування та підготовка звіту, який може бути використаний для демонстрації якості та

ефективності нечітких регуляторів перед впровадженням у реальну систему[9].

1.3 Інтеграція нечітких регуляторів з іншими методами керування

Інтеграція нечітких регуляторів з іншими методами управління є важливим аспектом розвитку систем автоматичного управління. Цей підхід дозволяє досягти кращої ефективності та надійності в різних сферах застосування[10].

По-перше, інтеграція нечітких регуляторів із класичними ПІД-регуляторами може бути корисною в ситуаціях, коли точність та стійкість до збурень є критичними. Класичні ПІД-регулятори можуть забезпечити точне регулювання у сталому стані, у той час як нечіткі регулятори можуть бути використані для компенсації нелінійності та невизначеності в системі[11].

По-друге, інтеграція з нейромережами дозволяє створювати адаптивні системи управління. Нейромережі можуть навчатися на основі даних та самостійно адаптуватися до змінних умов. Поєднання нечітких регуляторів із нейромережами створює потужний інструмент для управління складними системами[12].

По-третє, генетичні алгоритми можуть бути використані для оптимізації параметрів нечітких регуляторів. Цей підхід дозволяє автоматично підбирати найкращі налаштування для досягнення певних критеріїв управління.

Модульність також є важливою. Розробка нечітких регуляторів як окремих модулів управління робить інтеграцію з іншими системами більш простою та зручною[13].

Загалом, інтеграція нечітких регуляторів з іншими методами керування відкриває широкі можливості для покращення продуктивності та адаптивності систем автоматичного керування в різних сферах, де точність та надійність є важливими.

Математична основа нечітких регуляторів у системах автоматичного керування базується на теорії нечітких множин і логіки нечіткості. Основна ідея полягає в тому, щоб ураховувати невідомі чи нечітко визначені значення параметрів системи та управління у вигляді нечітких множин, які використовуються для опису неоднозначності та невизначеності в системі [14].

Математично, нечіткі регулятори використовують нечіткі правила, які включають умови та висновки, обчислені на основі ступенів належності до нечітких множин. Ці правила використовуються для прийняття рішень та генерації керуючих сигналів. Основні математичні операції включають обчислення ступенів належності, агрегацію правил, інтерполяцію та дефазифікацію. Математична основа нечітких регуляторів дозволяє моделювати та керувати системами, які мають нестаціонарність, нелінійність та невизначеність [15]. Вона забезпечує інструмент для вираження експертних знань та легке введення нових правил за необхідності. Така математична основа робить нечіткі регулятори потужними та універсальними засобами для розв'язання складних завдань автоматичного керування.

1.4 Синтез нечіткого регулятора в системі автоматичного регулювання температури водонагрівача

Синтез нечітких регуляторів в системах автоматичного регулювання температури водонагрівача має велику актуальність у зв'язку з поруч факторів. Крім того, використання нечіткої логіки може допомогти керувати системами в умовах невизначеності, неоднорідності та змінюваності параметрів оточуючого середовища. Це особливо важливо в системах, де зовнішні умови можуть швидко змінюватися, такі як водонагрівачі для домашнього використання.

Переваги:

Адаптабельність: Нечіткі регулятори не можуть адаптуватися до змінних систем роботи та від умов навколишнього середовища, що забезпечує стабільне та ефективне керування.

Робота з невизначеністю: Нечітка логіка дозволяє моделювати та обробляти невизначеність, яка може виникнути внаслідок різних факторів, таких як зміна вхідних параметрів чи нечіткість вихідних даних.

Простота моделювання: Моделювання нечітких регуляторів у системах автоматичного регулювання може бути більш простим та зрозумілим у порівнянні з традиційними методами.

Недоліки:

Складність налаштування: Налаштування параметрів нечітких регуляторів може бути витратним процесом, особливо для складних систем.

Велика кількість правил: У складних системах може знадобитися велика кількість правил нечіткої логіки, що робить їх обслуговування та накладення більших витрат.

Залежність від досвіду: Налагодження нечіткого регулятора часто вимагає досвіду та експертності для досягнення оптимальної ефективності.

В цілому, нечіткі регулятори є потужним інструментом для автоматичного регулювання температури водонагрівача, але їх ефективність та встановлена вартість повинні бути остаточно зважені в контексті конкретних умов та вимог системи. Скорочена функціональна схема САР, зображена на (рис. 1.2).

Рисунок 1.2 - Функціональна схема САР.

1.5 Аналіз конкурентів в сфері розробки нечітких регуляторів та системи автоматичного керування температурою

На ринку розробки нечітких регуляторів та системи автоматичного керування температурою існує кілька компаній, які конкурують за лідерство та постійно вдосконалюють своє рішення. Ось декілька конкретних прикладів конкурентів у цій сфері, на (рис 1.3) зображено регулятор в системі автоматичного регулювання температури водонагрівача компанії Siemens.

Рисунок 1.3 - Регулятор в системі автоматичного регулювання температури водонагрівача компанії Siemens.

1. Siemens пропонує різноманітні рішення в області автоматичного керування, включаючи нечіткі регулятори. Вони надають комплексні системи для промислових та комерційних програм.

Siemens пропонує широкий спектр продуктів у сфері автоматизації та керування, включаючи нечіткі регулятори. Їхні рішення можна

використовувати в різних промислових галузях та комерційних секторах.

Компанія завжди вдосконалює свої технології та впроваджує інноваційні рішення для забезпечення високої ефективності та точності в системах автоматичного керування.

Siemens є одним із ключових гравців на глобальному ринку автоматизації та контролю. Їхні продукти виробляються у великій кількості різних галузей, що показують про високу кількість довіри клієнтів.

Продукти Siemens легко інтегруються з іншими системами автоматизації та управління, що дозволяє створювати комплексні та високоефективні рішення.

Продукти Siemens часто мають високу вартість, що може бути обмежуючим фактором для менших підприємств або проектів з обмеженим бюджетом.

Завдяки широкому спектру функцій і можливостей продукти Siemens можуть бути складними в налаштуваннях, особливо для користувачів без значного досвіду.

Використання продуктів Siemens може зв'язатися з їхньою екосистемою, що може бути невигідним у випадку потреби в інтеграції з продуктами інших виробників.

Якщо великі підприємства зі складними потребами можуть виникнути в ситуації, коли продукти Siemens можуть бути менш гнучкими або менш адаптованими до конкретних вимог .

Загалом, Siemens є визнаним лідером у галузі автоматизації та контролю, але користувачам необхідно вивчити специфікації та врахувати потреби свого проекту перед вибором їхніх рішень.

2. Honeywell володіє обширним досвідом у розробці систем автоматизації та контролю. Вони включають інтелектуальні рішення для керування температурою та іншими параметрами на (рис 1.4) зображено регулятор в системі автоматичного регулювання температури водонагрівача компанії Honeywell.

Рисунок 1.4 - Honeywell регулятор в системі автоматичного регулювання температури водонагрівача

Honeywell пропонує різноманітні рішення в області автоматизації та керування, включаючи продукти для регулювання температури. Це дозволяє їм задовольняти потреби різних галузей та продуктів.

Компанія активно впроваджує інновації у свою продукцію, що дозволяє їм залишатися конкурентоспроможними та відповідати сучасним вимогам.

Продукти Honeywell часто володіють гнучкістю та можливістю налаштування, що сприяє їх використанню в різноманітних умовах та на різних типах об'єктів.

Продукти Honeywell фактично легко інтегруються з іншими системами автоматизації та мають високу ступінь сумісності.

Технології Honeywell можуть бути витратними, що може стати обмеженням для менших підприємств чи проектів з обмеженим бюджетом.

У деяких випадках, залежно від конкретних потреб користувача,

конфігурація продукту Honeywell може бути сумісною з іншими рішеннями.

У різних галузях можна виявляти конкретні вимоги, для яких продукти Honeywell можуть бути менш гнучкими або менш адаптованими.

Деякі користувачі вказують на те, що в окремих випадках продукти Honeywell можуть бути менш конкурентоспроможними в порівнянні з конкретними рішеннями для певних галузей.

Не дивлячись на недоліки, Honeywell залишається визнаним гравцем на ринку автоматизації та систем керування, а їхні технології продовжують використовуватися в різних галузях та застосуваннях.

3. Schneider Electric спеціалізується на системах автоматизації та електротехніки. Вони надають рішення для керування будівлями та промисловими процесами, включаючи нечіткі регулятори, на (рис 1.5) зображено регулятор в системі автоматичного регулювання температури водонагрівача компанії Schneider Electric.

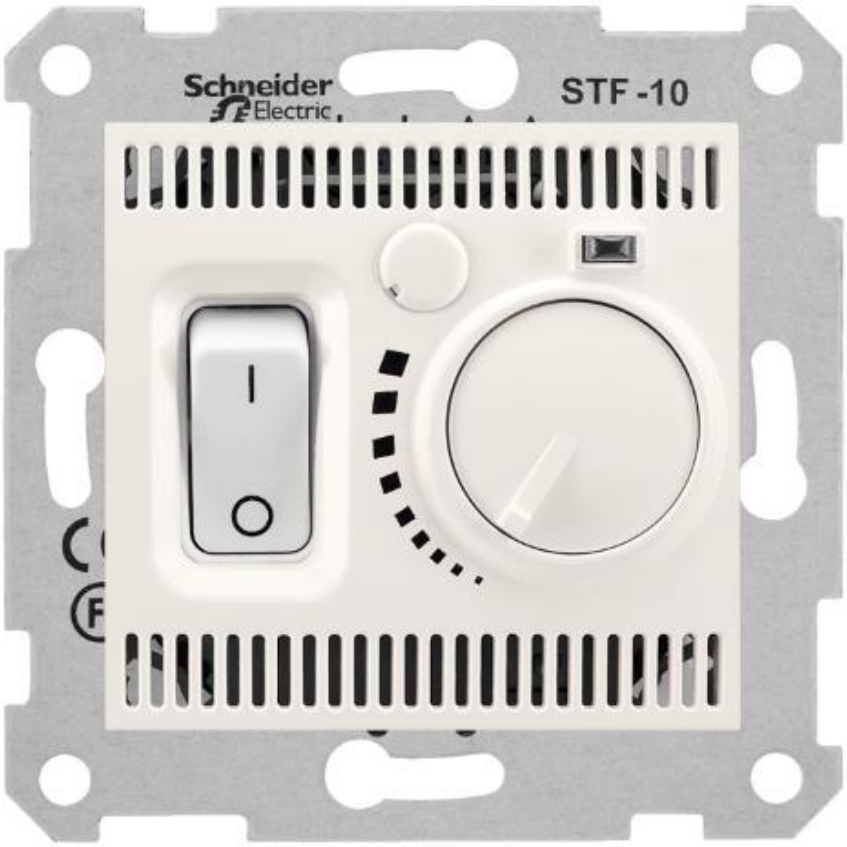

Рисунок 1.5 - Регулятор в системі автоматичного регулювання температури водонагрівача компанії Schneider Electric.

Schneider Electric володіє широким спектром інтегрованих рішень в галузі автоматизації та керування, включаючи системи автоматичного керування температурою. Це дозволяє забезпечити комплексний підхід до вирішення проблем клієнтів.

Schneider Electric акцентує увагу на енергоефективності своїх рішень, які можуть бути основним фактором для підприємств, що залишають перед собою завдання зниження витрат енергії.

Рішення Schneider Electric застосовується в різних галузях, включаючи промисловість, будівництво та інфраструктуру, що робить їх гнучкими та придатними для різноманітних викликів.

Компанія систематично інвестує в дослідження та розробки, щоб залишитися конкурентоспроможною та впроваджувати передові технології у своїх продуктах.

У різних випадках, особливо в комплексних системах, можна виникати складність у налаштуванні та обслуговуванні.

У деяких випадках продукти Schneider Electric можуть виявитися менш сумісними з продуктами інших виробників, які можуть бути присутніми в наявних системах.

Також може бути висока вартість рішень Schneider Electric, що може стати обмеженням для менших підприємств або проектів з обмеженим бюджетом.

Для ефективного використання продукту Schneider Electric може бути наявний досвід у сфері автоматизації та керування транспортними засобами.

Хоча Schneider Electric володіє численними перевагами та залишається гравцем в галузі автоматизації, потрібно забезпечити конкретні потреби та характеристики проекту при виборі ваших рішень.

Ці компанії конкурують, пропонуючи різноманітні інтегровані рішення для автоматичного керування, включаючи температурне регулювання водонагрівачів, та намагаються постійно вдосконалювати свої технології для

забезпечення ефективності та стабільності в різних застосуваннях.

1.6 Висновки до розділу 1

Аналізуючи сучасний стан розробки нечітких регуляторів у системах автоматичного керування та враховуючи попередні запитання, стає очевидним, що ринок насичених конкуренцією представлений такими компаніями, як Siemens, Honeywell і Schneider Electric. Кожен із цих виробників виступає з унікальними рішеннями, пропонуючи високотехнологічні продукти, орієнтовані на енергоефективність та інновації.

У контексті переваг компанії Siemens славиться своїм широким асортиментом продукції та акцентом на інноваціях, Honeywell використовує свою гнучкість та інтеграцію, тоді як Schneider Electric відзначає енергоефективність та комплексні рішення.

Не дивлячись на ці переваги, існують спільні проблеми для користувачів із високою вартістю, можливих складнощів у налаштуваннях системи та залежно від досвіду. Кожен вибір компанії повинен бути обґрунтованим з огляду на конкретні потреби та характеристики проекту, з наданням забезпечення ефективності та сумісності відповідних систем у виробничому середовищі. За всім цим варто зауважити, що ринок системи автоматичного керування еволюціонує у напрямку інновацій та стабільності, підтримуючи важливі аспекти ефективності та відповідальності за використання енергії.

2 ПРОЕКТУВАННЯ СИСТЕМИ «РЕГУЛЯТОРА В СИСТЕМІ АВТОМАТИЧНОГО РЕГУЛЮВАННЯ ТЕМПЕРАТУРИ ВОДОНАГРІВАЧА»

2.1 Моделі розробки автоматизованих тестів для синтезу нечітких регуляторів

Вибір ефективної моделі розробки автоматизованих тестів для синтезу нечітких регуляторів у системах автоматичного управління є критичним завданням, що впливає на якість та надійність регуляторів. Існує кілька підходів до вибору такої моделі[16].

Один із можливих підходів – це модель з етапами валідації. У цьому варіанті розробки тести розглядаються як послідовні етапи валідації системи управління з нечітким регулятором. Починаючи з створення симуляційних моделей системи, розробники визначають тестові кейси та виконують їх, а потім аналізують результати, виявляють помилки та вносять корективи[17]. Цей підхід дозволяє систематично валідувати регулятор на різних етапах розробки, що сприяє виявленню та виправленню проблем .

Інший підхід може включати тестування на реальних об'єктах, якщо така можливість існує. У цьому випадку, тестування проводитиметься безпосередньо на фізичних пристроях або системах, на яких буде застосовуватися нечіткий регулятор. Цей метод може дати більш точні результати та оцінку реальної ефективності, але він може бути обмежений доступом до об'єктів для тестування[18].

Загалом вибір моделі розробки тестів залежить від конкретних умов та ресурсів проекту. Кожен з цих підходів має свої переваги та недоліки, і розробники повинні ретельно розглянути їх, щоб забезпечити надійну та ефективну розробку нечітких регуляторів для систем автоматичного керування[19].

Рисунок 2.1 - Ітераційно-циклічний характер прийняття оптимальних рішень у процесі проектування та створення

Ще однією можливою моделлю розробки тестів для синтезу нечітких регуляторів є модель, яка базується на аналізі статичних та динамічних властивостей системи. На початковому етапі, проводитиметься математичне моделювання системи, включаючи нечіткі регулятори, та аналізується її поведення в різних умовах. На основі цього аналізу визначаються параметри, які потрібно перевірити та налаштувати[20].

Далі, створюються автоматизовані тестові скрипти, які запускаються для перевірки конкретних властивостей регулятора, таких як стійкість до збурень, час реакції, точність та інші. Ці тести можуть включати у собі віртуальне моделювання чи експерименти на реальних об'єктах[21].

Ключовим етапом у цій моделі є аналіз результатів тестів та налаштування параметрів регулятора на основі цього аналізу. Процес повторюється, доки не буде досягнуто заданих характеристик керування.

Ця модель розробки тестів дозволяє більш детально вивчити та оптимізувати властивості регулятора та враховувати вплив змінних умов роботи системи. Однак вона може бути більш час- тою ресурсомісткою порівняно з іншими підходами. Налаштування регулятора є важливим етапом у процесі синтезу нечітких регуляторів у системах автоматичного керування. Воно полягає у визначенні оптимальних значень параметрів регулятора для

досягнення певних критеріїв управління, таких як стійкість, точність та швидкість відгуку системи. Налаштування може виконуватись як експериментально, так і на основі математичних методів[22].

Існують різні методи налаштування нечітких регуляторів, включаючи:

Експертні налаштування: У цьому випадку, досвідчений оператор або інженер вручну визначає значення параметрів регулятора на основі свого досвіду та експертних знань. Цей метод може бути ефективним, коли відсутні точні математичні моделі системи[23].

Методи оптимізації: Використання математичних методів оптимізації для автоматичного підбору параметрів регулятора. Серед таких методів можуть бути генетичні алгоритми, методи градієнтного спуску, алгоритми оптимізації зі зворотним поширенням (backpropagation) та інші[24].

Траленик (Trial-and-Error): Повторне виконання експериментів з різними значеннями параметрів регулятора та аналіз впливу цих параметрів на роботу системи. Цей метод можна використовувати для простих систем.

Системи автоматичного налаштування (Autotuning): Використання спеціальних програмних засобів або апаратних пристроїв для автоматичного налаштування параметрів регулятора на основі аналізу відгуку системи[25].

2.2 Вибір мови програмування та середовища розробки контролера

Вибір мови програмування для розробки регулятора в системі автоматичного регулювання температури водонагрівача залежить від ряду факторів, таких як потреби проекту, ефективність, швидкість розробки та зручність. Однак показано, що використання JavaScript (JS) може бути доцільним у деяких випадках. Ось деякі аргументи, які можуть вказувати на те, що JS є гарним вибором для цього типу проекту:

JS широко використовується у веб-розробці, і велика частина системи автоматичного регулювання може бути впроваджена у вікні веб-застосунку. Це забезпечує безпечний дистанційний доступ та систему керування через

веб-інтерфейс.

Node.js для серверної частини:

Якщо необхідна серверна частина, Node.js, яка використовує JS, може бути ефективним вибором. Вона дозволяє швидко створювати сервери та використовувати JavaScript як мову на обох сторонах, клієнтській та серверній, середовище Node.js зображено на (рис 2.2).

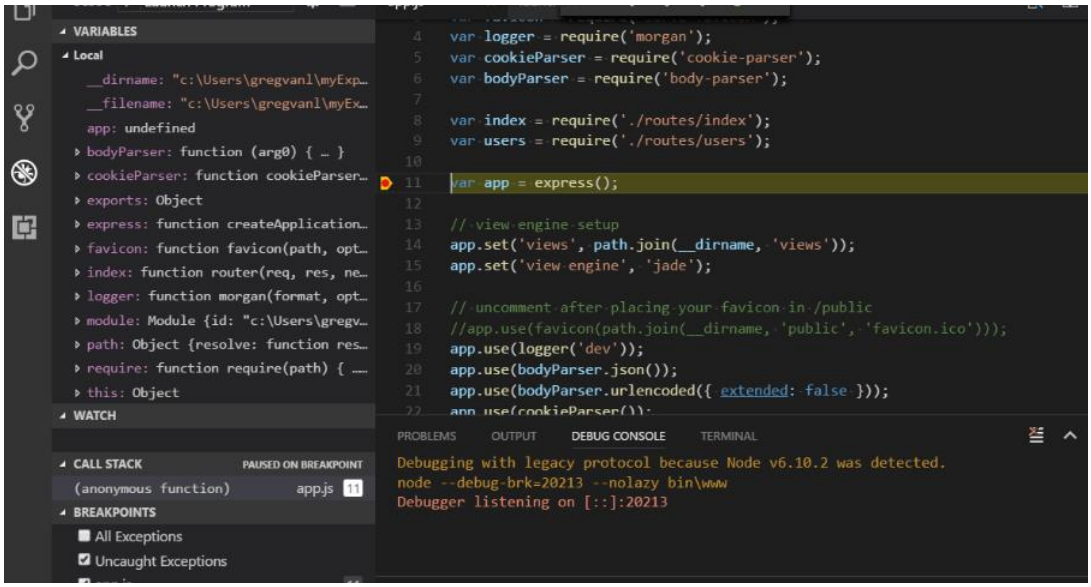

Рисунок 2.2 - Середовище розробки серверної частини Node.js

JS має велику кількість бібліотек та фреймворків, таких як React або Angular, які полегшують взаємодію з веб-інтерфейсом та можуть сприяти швидкому розгортанню та розробці.

JS має велику та активну спільну роботу розробників, а його екосистема надає доступ до численних плагінів, бібліотек та інструментів, які можуть полегшити розробку та оптимізувати код.

Властивість асинхронності JS, зокрема в Node.js, може бути корисною у випадках, коли система потребує реагування на події в реальному часі, наприклад, швидко регулювати температуру відповідно до змін у середовищі.

Середовище розробки Visual Studio Code (VS Code) використовує вашу популярність та гнучкість, і воно може бути відмінним вибором для розробки

системи автоматичного регулювання температури водонагрівача. Ось деякі аспекти, які створюють код VS у вигляді вибору для цієї роботи зображення програмного середовища можна побачити на (рис.2.1):

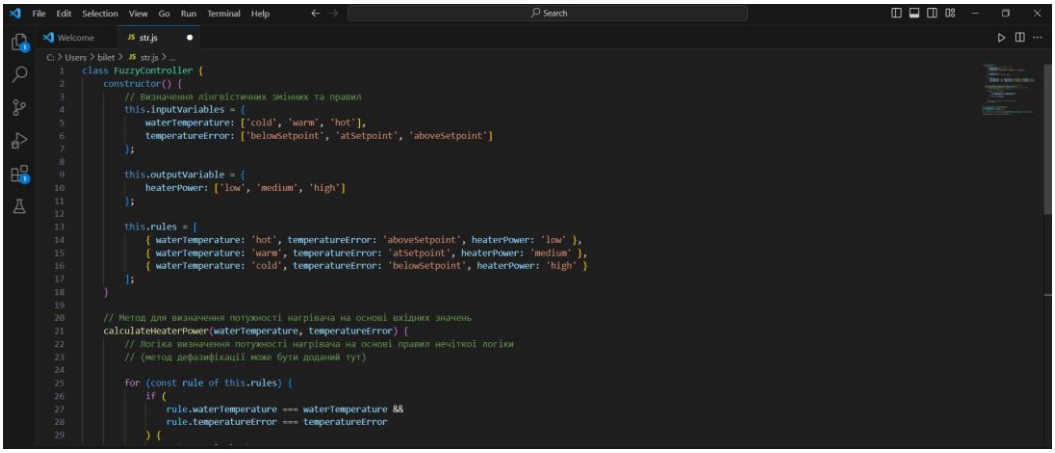

Рисунок 2.1 - Середовище розробки Vs code

VS Code має велику кількість розширень та додатків, що дозволяє розробникам легко адаптувати середовище до своїх потреб. Це особливо важливо при розробці системи автоматичного регулювання, де можна вимагати конкретних інструментів чи розширених можливостей.

VS Code підтримує різні мови програмування, включаючи JavaScript та його розширення, які можна використовувати для розробки веб-застосунків або серверних систем.

Вбудована підтримка Git робить спільну роботу над проектом більш зручною та підтримує контроль версії коду.

VS Code має інтуїтивний і простий інтерфейс, що сприяє легкій навігації та швидкій реакції під час розробки.

Наявність вбудованої консолі спрощує відлагодження та взаємодію з програмою під час розробки та тестування.

VS Code добре підтримує розробку веб-застосунків, що може бути в кількості для автоматичного регулювання веб-інтерфейсу системи.

VS Code використовує велику та активну спільну розробку, що означає наявність багатьох функціональних плагінів, навчальних матеріалів та

регулярне оновлення.

Загалом, VS Code володіє всіма необхідними інструментами для ефективної розробки системи автоматичного регулювання температури водонагрівача, забезпечуючи комфортний та продуктивний процес програмування.

2.2 Створення UML, IDEF діаграм

IDEF0 (Integration Definition for Function Modeling) - це методологія моделювання функцій, яка використовується для визначення, опису та аналізу системних функцій. Моделі IDEF0 розглядають систему з точки зору виконання різних функцій та їх взаємодії.

IDEF0 першого рівня для процесу залучення інвесторів представляє собою визначення ключових функцій та їхніх взаємозв’язків у контексті залучення інвесторів до певної діяльності чи проекту. Це може бути корисно при розробці стратегій та планів для привернення фінансування від інвесторів.

На першому рівні IDEF0 для процесу залучення інвесторів можуть бути визначені такі елементи

Визначення конкретних завдань, які повинні бути виконані в рамках процесу залучення інвесторів, наприклад, проведення маркетингових досліджень або розробка бізнес-плану.

Вказання ресурсів, які використовують для здійснення функцій, наприклад, персонал, фінансові ресурси, маркетингові інструменти тощо.

Визначення взаємодії між безкоштовними функціями та елементами системи в контексті залучення інвесторів.

Вихідні результати:

Вказання очікуваних результатів виконання функцій, наприклад, залучення певної суми інвестицій чи залучення нових інвесторів.

IDEF0 допоможе систематизувати та уточнити процеси та взаємозв'язки в складних системах, що дозволяє розробникам та менеджерам

отримати чіткі уявлення про функціональні аспекти системи та їхню взаємодію, на рисунку 2.2 зображено - IDEF0 першого рівня для процесу залучення інвесторів

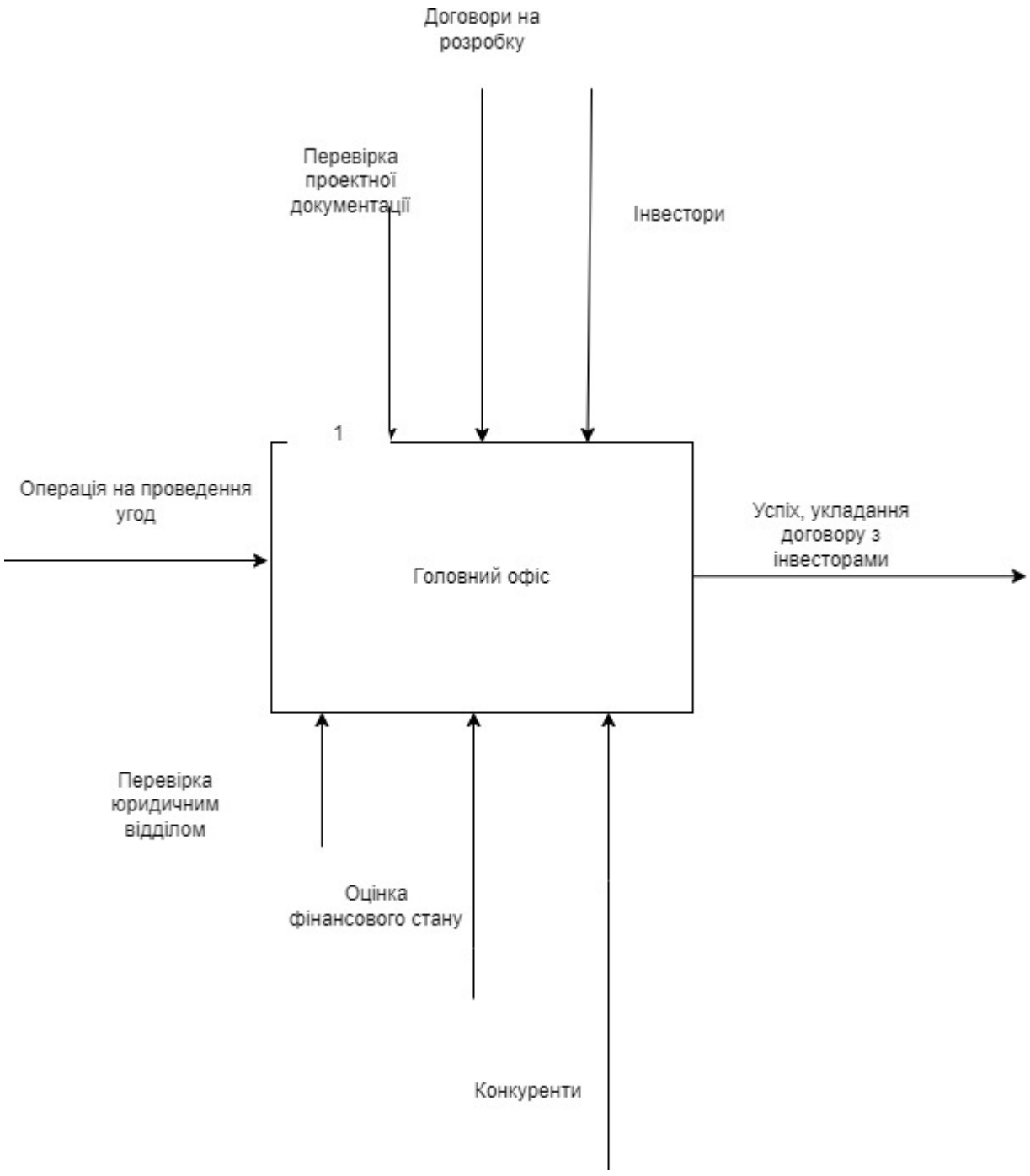

Рисунок 2.1 - IDEF0 першого рівня для процесу залучення інвесторів

Проектування структури бази даних для веб-проекту регулятора в системі автоматичного регулювання температури водонагрівача на Node.js

може бути ключовим етапом у розробці. Враховуючи потреби системи автоматичного регулювання, ось конкретний підхід до структури бази даних, та корпоративної мережі, зображений на рисунку 2.2.

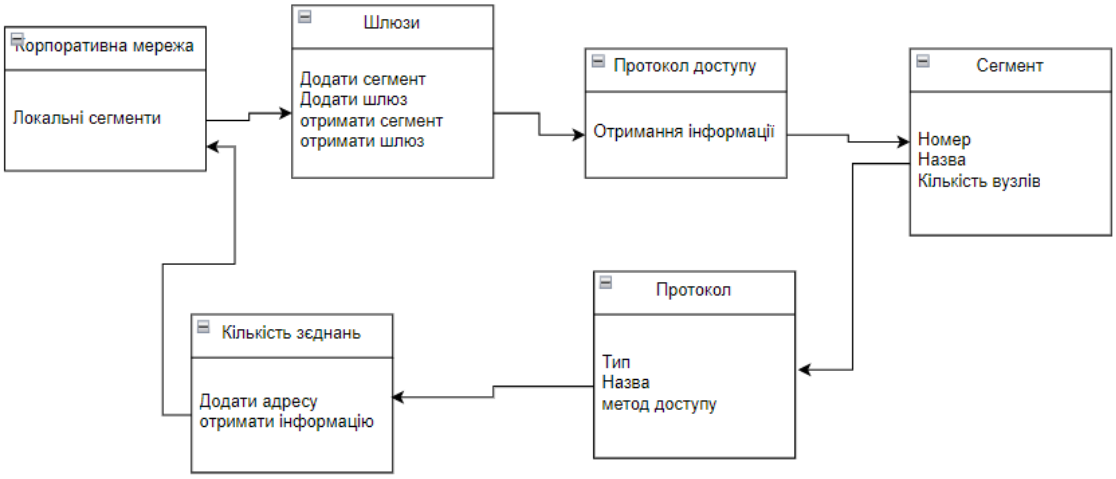

Рисунок 2.2 - UML-object diagram корпоративної мережі за стосунку

UML-діаграма об'єктів (Object Diagram) у контексті корпоративної мережі може служити для візуалізації конкретних об'єктів та їх взаємозв'язків у мережевому середовищі. На такій діаграмі можуть бути представлені конкретні компоненти, пристрої та взаємодія між ними.

Проте слід врахувати, що UML-діаграми об'єктів традиційно використовуються для моделювання статичних структурних систем на рівнях об'єктів у певний момент часу, тому їх застосування в корпоративних мережах може бути обмеженим. Давайте розглянемо, як можна представити таку діаграму.

Кожен сервер може бути представлений об'єктом, включаючи його характеристики, такі як IP-адреса, назва, обладнання та ін.

Об'єкти, які представляють комп'ютери або інші пристрої, що підключені до мережі, із зазначенням своїх параметрів та характеристик.

Маршрутизатори, комутатори та інші мережеві пристрої можуть бути

представлені в якості об'єктів з описом їхніх функцій та з'єднані.

Якщо в мережі присутні бази даних, їх можна включити до діаграм об'єктів, що показують взаємодію з серверами та клієнтами.

2.3 Реалізація серверної частини

Програмна реалізація проекту веб-системи для регулятора в системі автоматичного регулювання температури водонагрівача на Node.js може бути здійснена за допомогою фреймворка Express.js для веб-сервера та MongoDB для зберігання даних. Давайте розглянемо загальний каркас реалізації:

Крок 1: Установка пакетів

Встановлюємо кількість пакетів за допомогою npm (Node Package Manager) підключення підключається на (рис. 2.11):

```
npm init -y
npm install express mongoose body-parser
```

Рисунок 2.11 - Встановлення пакетів

Крок 2: Створення сервера

Створюємо файл app.js програмування сервера відбувається на (рис 2.12):

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');
const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

// Підключення до бази даних MongoDB
mongoose.connect('mongodb://localhost:27017/temperature_control', { useNewUrl

// Структура моделей для користувачів, пристроїв та журналу температур
const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  password: String,
});

const deviceSchema = new mongoose.Schema({
  name: String,
  user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  maxTemperature: Number,
  minTemperature: Number,
});

const temperatureLogSchema = new mongoose.Schema({
  device: { type: mongoose.Schema.Types.ObjectId, ref: 'Device' },
  temperature: Number,
  timestamp: { type: Date, default: Date.now },
```

Рисунок 2.12 - Програмування сервера

Програмування сервера на Node.js включає в себе створення серверного коду за допомогою JavaScript або інших програмних засобів, таких як TypeScript. Вибравши підходящий фреймворк, наприклад, Express, розробник створює логіку сервера для обробки запитів, визначення маршрутів та взаємодії з базою даних, якщо це необхідно. Результативний сервер може виконувати різні завдання, такі як обробка запитів, відправлення даних клієнтам і виконання бізнес-логіки. Програміст також може розглядати використання різних пакетів і бібліотек для покращення функціональності сервера.

3 РОЗДІЛ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ВЕБ СИСТЕМИ РОЗРОБКИ РЕГУЛЯТОРА В СИСТЕМІ АВТОМАТИЧНОГО РЕГУЛЮВАННЯ ТЕМПЕРАТУРИ ВОДОНАГРІВАЧА

1.1 Програмна реалізація проекту

HTML-документ змінює структуру та елементи сторінки для системи управління температурою, і його поведінка (додавання пристроїв, відображення журналу температури) буде реалізовано за допомогою JavaScript, який закривається у файлі "app.js".

Цей фрагмент коду являє собою базовий HTML-документ, який створений для веб-додатка з системою керування температурою. Давайте розглянемо його основні функції та елементи:

<body>:

У цьому розділі розміщується вміст сторінок, які будуть відображатися користувачам.

:

Створює контейнер, який містить увесь вміст сторінок.

<h1>Система контролю температури</h1>:

Заголовок рівня 1, який вказує назву системи керування температурою.

:

Контейнер для відображення списку пристроїв. Точно він пустий, і його вміст буде вставлений динамічно за допомогою JavaScript.

<form id="add-device-form">:

Форма для додавання нового пристрою. Включає поля для введення імені пристрою, максимальної та мінімальної температури, а також кнопку для відправлення форми.

:

Контейнер для відображення записів журналу температур. Початково він пустий, а вміст його також буде вставлений динамічно за допомогою

JavaScript.

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>:

Підключає бібліотеку Axios для виконання HTTP-запитів з JavaScript.

<script src="app.js"></script>:

Підключає скрипт "app.js", який містить JavaScript-код для взаємодії з сервером, динамічного оновлення сторінки та обробки подій, результат роботи зображено на (рисунку 3.1).

Рисунок 3.1 – Діалогове вікно контролю системою

Далі створюємо CSS-код, який створює стилі та виглядає елементи HTML-сторінки, яка реалізує системну температуру керування. Основна його роль полягає у встановленні зовнішнього вигляду сторінки, щоб зробити зручний та привабливий інтерфейс для користувачів. Декілька ключових аспектів стилів варто відзначити:

body:

Встановлює основні стилі для всього документа, такі як шрифт, відступи та фоновий колір.

.container:

Визначає стилі для контейнера, який обгортає основний вміст сторінки. Встановлює максимальну ширину, відступи, фоновий колір, внутрішні відступи, радіус кутів і тінь, щоб виділити вміст на сторінці.

h1, h2:

Встановлює стилі для заголовків рівнів 1 і 2, такі як вирівнювання тексту, колір тексту.

form:

Додає відступ вгорі до форми для покращення вигляду та відокремлення від інших елементів.

label:

Встановлює стилі для міток, зокрема, визначає вигляд блокових елементів з відступами вниз.

input, button:

Встановлює спільні стилі для текстових полів та кнопок, такі як відступи та величину внутрішнього заповнення.

button:

Визначає стилі для кнопок, такі як фоновий колір, колір тексту, появи рамки, закруглені кути та зміну вигляду при наведенні.

Ці стилі допомагають створити однорідний та привабливий дизайн сторінок, полегшуючи читання та взаємодію користувачів з елементами веб-додатка загальний вигляд показано на (рис. 3.2).

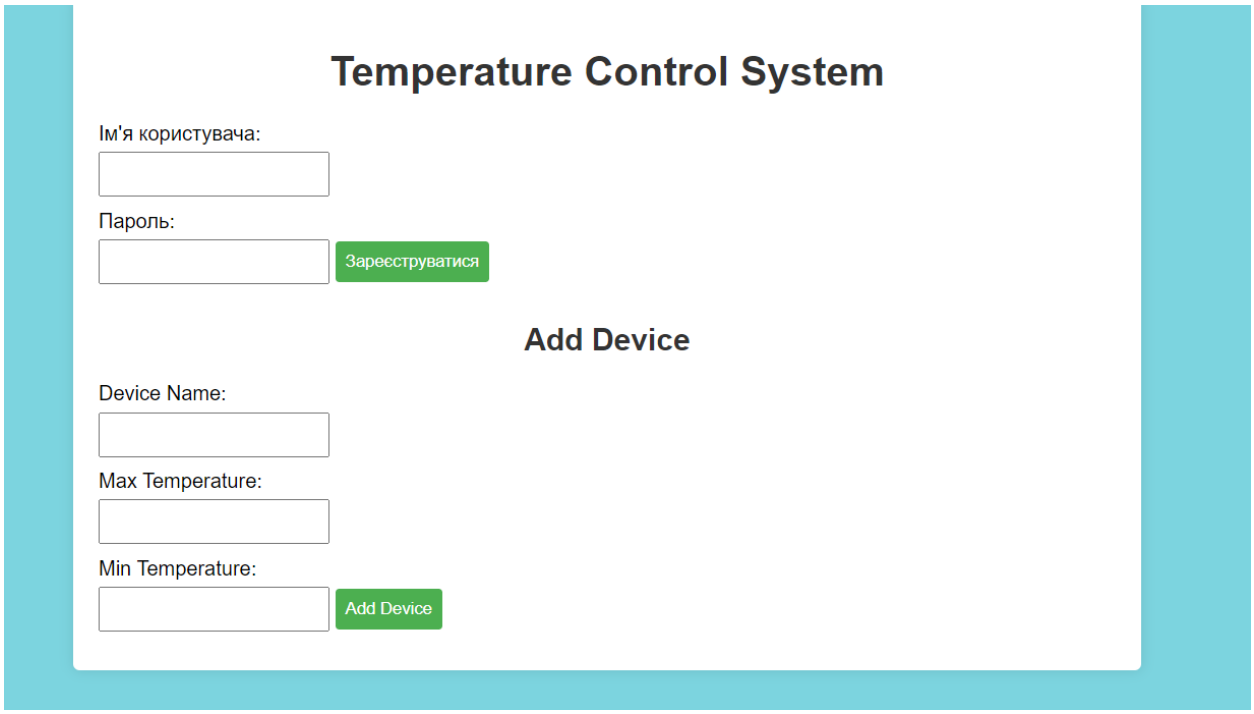

Рисунок 3.2 – Форма реєстрації web-застосунку

Наступний етап написання JavaScript-коду, що виконує кілька функцій для інтерактивності та цієї динамічної оновленості веб-додатоку сторінки.

Давайте розглянемо кожну функцію.

document.addEventListener('DOMContentLoaded', () => { ... });

Ця функція захищена, коли DOM сторінки повністю завантажено. Це гарантує, що всі елементи DOM, до яких звертається код, вже використані та доступні для взаємодії.

const devicesContainer = document.getElementById('devices-container');

Знайти елемент з ідентифікатором 'devices-container' (це, ймовірно, контейнер для відображення списку пристроїв) і зберігає його в змінному devicesContainer.

const addDeviceForm = document.getElementById('add-device-form');

Знайти форму елемента з ідентифікатором 'add-device-form' і зберегти його в зміненому addDeviceForm.

const temperatureLogContainer = document.getElementById('temperature-log-container');

Знайти елемент з ідентифікатором 'temperature-log-container' (це, ймовірно, контейнер для відображення журналу температури) і зберігає його в змінному temperatureLogContainer.

const fetchDevices = async () => { ... }

Асинхронна функція для витягування та відображення списку пристроїв. Використовує бібліотеку Axios для виконання HTTP-запиту до сервера та створення списку пристроїв. Оновлює вміст devicesContainer з використанням даних, отриманих від сервера.

const fetchTemperatureLog = async () => { ... }

Асинхронна функція для витягування та відображення журналу температури. Також використовуйте Axios для виконання HTTP-запиту до сервера та оновлення вмісту temperatureLogContainer на основі отриманих даних.

addDeviceForm.addEventListener('submit', async (event) => { ... });

Додає слухача подій для форми addDeviceForm, який закривається при її відправці (submit). Забороняє стандартну форму поведінки за допомогою

event.preventDefault(). Отримує форми (ім'я, максимальна та мінімальна температура) і відправляє їх на сервер за допомогою HTTP-запиту POST. Після успішної відправки завантажте функцію fetchDevices для оновлення списку пристроїв та форм очищення.

fetchDevices();

Використовує функцію fetchDevices для відновлення початкового списку пристроїв при завантаженні сторінки.

fetchTemperatureLog();

Використовує функцію fetchTemperatureLog для відновлення початкового журналу температури при завантаженні сторінки.

Отже, цей код реалізує витягування та відображення списку пристроїв і журнал температур, а також надає можливість динамічно додавати нові пристрої за допомогою форми відправлення.

1.2 Програмна реалізація бази даних

Налаштування сервера:

Визначається порт, на якому сервер буде слухати запити (або використовувати 3000, якщо не визначено інше).

Підключення до бази даних:

Встановлено з'єднання з базою даних MongoDB за допомогою Mongoose.

Визначення схем для моделей даних:

Визначається схема для моделі "Пристрій", яка включає поля name, maxTemperature та minTemperature.

Визначається схема для моделі "TemperatureLog", яка включає поля пристрою (посилання на модель Device), температуру та мітку часу.

Створення моделей на основі схеми:

Створюється модель "Пристрій" за допомогою визначеної схеми.

Створюється модель "TemperatureLog" за допомогою визначеної схеми.

Обробка HTTP-запитів:

POST /api/devices:

Створює новий пристрій у базі даних на основі отриманих даних у запиті.

Відповідає зі створеним пристроєм у форматі JSON.

GET /api/devices:

Отримано всі пристрої з бази даних.

Відповідає список пристроїв у форматі JSON.

GET /api/temperature-log:

Останні 10 записів журналу температури з бази даних, сортуючи їх за датою.

Відповідає списком записів журналу у форматі JSON, включаючи пов'язану інформацію про пристрій.

Запуск сервера:

Запускає сервер на визначений порт і виводить повідомлення про те, що сервер працює.

Код створює API для додавання пристроїв, створення списку цих пристроїв та створення останніх записів журналу температури, програмна реалізація зображення на (рис. 3.3).

```
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/temperature_control', { useNewUrl

const deviceSchema = new mongoose.Schema({
  name: String,
  maxTemperature: Number,
  minTemperature: Number,
});

const temperatureLogSchema = new mongoose.Schema({
  device: { type: mongoose.Schema.Types.ObjectId, ref: 'Device' },
  temperature: Number,
  timestamp: { type: Date, default: Date.now },
});

const Device = mongoose.model('Device', deviceSchema);
const TemperatureLog = mongoose.model('TemperatureLog', temperatureLogSchema)

app.post('/api/devices', async (req, res) => {
  try {
    const device = await Device.create(req.body);
```

Рисунок 3.3 - Програмна реалізація бази даних

Представлений у вигляді прямокутника з назвою "Сервер". У середовищі можуть бути наведені основні характеристики сервера, такі як операційна система, апаратне забезпечення, IP-адреса тощо, це зображено на рисунку 3.4 - UML-deployment diagram.

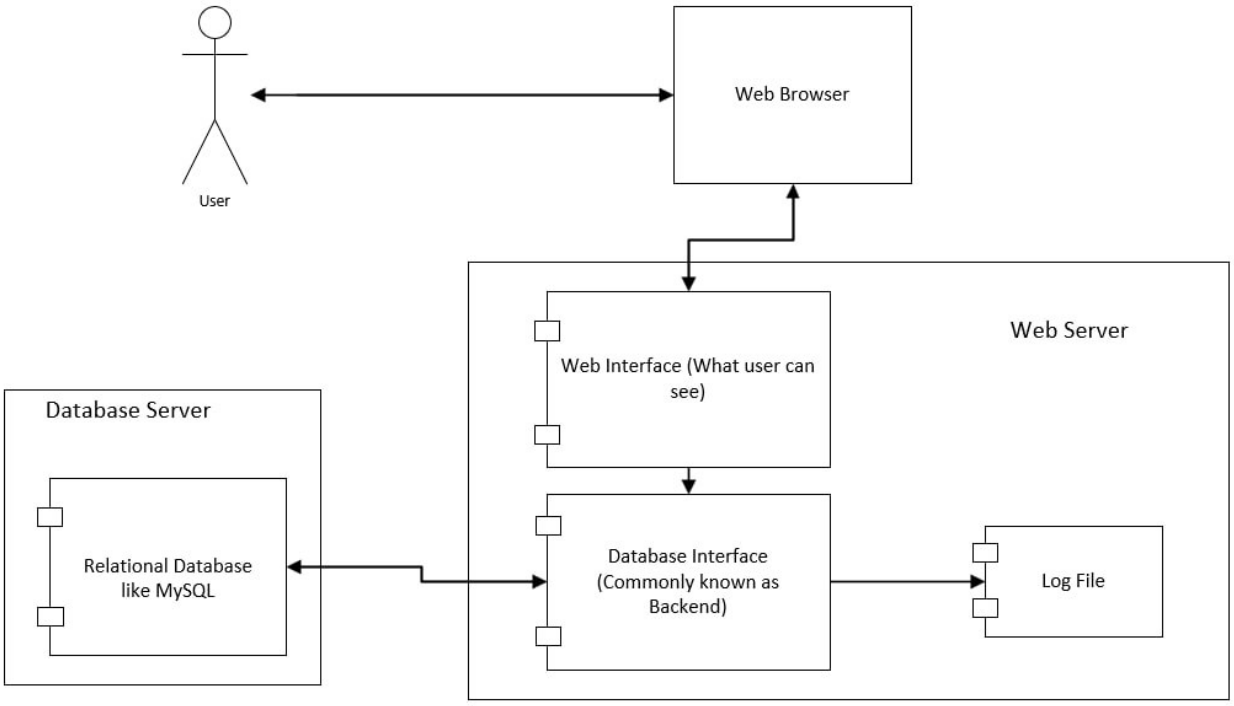

Рисунок 3.4 - UML-deployment diagram

UML-діаграма розгортання (Deployment Diagram) визначає фізичне розташування компонентів системи та їх взаємодію на рівнях апаратного забезпечення. Давайте розглянемо, як можна описати діаграму розгортання для серверної частини системи:

Індикатори додатків, які розгортані на сервері. зазвичай представлені у вигляді прямокутників з назвами додатків.

Якщо є бази даних, вони можуть бути представлені в окремих прямокутниках під сервером або поруч із додатками, які використовують ці бази даних.

Якщо сервер взаємодіє з клієнтами, їх можна відображати окремими символами, наприклад, комп’ютерами або мобільними пристроями.

З'єднання між компонентами можна зображати стрілками, які відображають напрямок взаємодії та за допомогою протоколу зв'язку.

4 ТЕСТУВАННЯ ТА ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ МОБІЛЬНОЇ ПРОГРАМИ ДЛЯ ОЦІНЮВАННЯ ЯКОСТІ НАВЧАННЯ В АКАДЕМІЧНІЙ ГРУПІ

4.1 Тестування

Тестування модуля (або блоку) — це процес тестування окремої підпрограми або процедури програми. Це означає, що перед початком тестування всієї програми ви повинні протестувати окремі невеликі модулі, які складають програму. Цей підхід мотивований трьома причинами.

По-перше, склад тестів можна контролювати, оскільки початкова увага приділяється невеликим модулям програми.

По-друге, це допомагає з налагодженням програми, тобто знайти місця помилок і виправити текст програми.

По-третє, допускається паралельність, що дозволяє тестувати кілька модулів одночасно. Метою тестування модуля є порівняння функціональності, реалізованої модулем, зі специфікацією його функціональності або інтерфейсу. Тестування модулів в основному зосереджено на принципі «білого ящика». В основному це пов’язано з тим, що принцип «білого ящика» складніше реалізувати при переході до тестів більших одиниць, наприклад загальних програм.

Крім того, подальші етапи тестування зосереджені на виявленні різних типів помилок, тобто помилок, які не обов’язково пов’язані з логікою програми, але виникають через те, що програма не відповідає вимогам користувача. Існує кілька можливих способів об’єднання модулів у більші блоки.

Тестуйте крок за кроком.

Реалізація процесу модульного тестування базується на двох ключових положеннях: побудові ефективного набору тестів і способі підбору комбінацій модулів при побудові з них робочої програми. Друга умова

важлива, оскільки вона визначає форму написання тестів модулів, типи інструментів, які використовуються в тестах, порядок кодування та тестування модулів, вартість створення тестів і вартість налагодження. Розглянемо два підходи до об'єднання модулів: поетапне тестування та цілісне тестування.

Виникає питання: «Що краще – виконати окремі тести для кожного модуля, а потім об'єднати їх у робочу програму або підключити кожен модуль до набору раніше перевірених модулів для тестування?».

Перший підхід часто називають цілісним підходом або підходом

«Великий хіт», при тестуванні та компіляції програм;

Другий підхід називається покроковим підходом до тестування або складання.

Підхід до покрокового тестування передбачає, що модулі не тестуються ізольовано один від одного, а послідовно підключаються до набору раніше перевірених модулів для виконання тестів. Покроковий процес триває до тих пір, поки останній модуль не буде підключено до набору тестових модулів.

Ми не будемо надавати детальний аналіз цих двох методів, наведемо лише деякі загальні висновки.

1. Однокристальне тестування вимагає великих витрат праці. При покроковому тестуванні знижуються витрати на оплату праці «знизу вгору».

2. Тест MCU займає менше машинного часу.

3. Використання монолітного підходу дає хорошу можливість для паралельної організації роботи на початковому етапі Тестувати (тестувати всі модулі одночасно). Це положення може бути важливим для реалізації великих проектів з великою кількістю модулів і багатьма виконавцями, оскільки найбільша кількість людей залучена до проекту на початковій фазі.

4. При покроковому тестуванні помилки інтерфейсу між модулями виявляються раніше, оскільки збірка програми починається раніше. Навпаки, при модульному тестуванні модулі «не бачать один одного» до самого кінця

процесу тестування.

5. Налагодження легше за допомогою покрокового тестування. Якщо в міжмодульному інтерфейсі є помилки, що зазвичай буває, то при модульному тестуванні їх можна виявити лише тоді, коли зібрана вся програма. На цьому етапі важко знайти помилку, оскільки вона може бути де завгодно програми. Навпаки, при покроковому тестуванні такі помилки в основному пов’язані з останнім підключеним модулем.

6. Результати покрокового тестування більш досконалі. На закінчення зазначимо, що пункти 1, 4, 5, 6 демонструють переваги поетапного тестування, а пункти 2 і 3 - його недоліки. Оскільки сучасний розвиток комп’ютерних технологій має тенденцію до зниження витрат на обладнання та збільшення витрат на оплату праці, то наслідки помилок у математичних програмних засобах є досить серйозними, а витрати на усунення помилок менші за встановлені раніше; зазначені в пунктах 1, 4, 5 , і 6 Переваги очевидні. У той же час недоліки (пункти 2 і 3) завдають незначної шкоди. Усе це приводить нас до висновку, що покрокове тестування є найбільш прийнятним.

Вірячи в переваги покрокового тестування перед модульним тестуванням, ми розглянули дві можливі стратегії тестування: спадну та висхідну. Спочатку з’ясуємо термінологію.

1. Терміни «спадний тест», «спадний розвиток»,

«Спадний дизайн» часто використовується як синонім. Насправді терміни «спадаючий тест» і «розробка за спаданням» є синонімами (у тому сенсі, що вони мають на увазі певну стратегію при тестуванні та створенні тексту модуля), але низхідне проектування — це зовсім інший і окремий процес. Програми, розроблені в порядку спадання, можна тестувати в порядку спадання і зростання.

По-друге, висхідна розробка або тестування часто поєднується з модульним тестуванням. Це непорозуміння виникає через те, що під час тестування нижнього або кінцевого модуля тест підйому починається так

само, як і монолітний тест. Але вище ми показали, що тест підйому насправді є покроковою стратегією.

Тестування за зростанням У методі за зростанням програми збираються та тестуються «знизу вгору». Тільки модулі найнижчого рівня (

«Термінальні» модулі, модулі, які не викликають інші модулі) незалежно та автономно тестуються. Після тестування цих модулів їх виклик має бути таким же надійним, як виклик вбудованих мовних функцій або операторів присвоєння. Потім протестуйте модуль і викличте вже перевірений напряму. Ці модулі високого рівня тестуються не автономно, а разом з уже перевіреними низькорівневими модулями. Повторюйте процес, поки не досягнете вершини. На цьому перевірка модуля та перевірка підключення програми завершена. У висхідних тестах для кожного модуля потрібен драйвер: тести повинні бути подані проти комбінації модулів, що тестуються. Можливе рішення – написати невеликий завантажувач для кожного модуля. Тестові дані «вбудовуються» безпосередньо в цю програмну змінну та структуру даних, яка викликає тестовий модуль неодноразово, передаючи щоразу нові тестові дані. Є краще рішення: скористайтеся тестером модулів — інструментом тестування, який дозволяє описувати свої тести спеціальною мовою і не вимагає написання драйверів. Немає жодних проблем, пов’язаних з можливістю або складністю створення тестових ситуацій для всіх типових тестів зверху вниз. Драйвер безпосередньо застосовується до тестованого модуля як інструмент тестування, без урахування проміжних модулів. Аналіз інших проблем, які виникають під час низхідного тестування, показує, що висхідне тестування не може прийняти нерозумне рішення поєднати тестування з розробкою програми, оскільки тестування не може розпочатися, поки не будуть розроблені модулі низького рівня. При переході до тестування не виникає труднощів у неповному тестуванні модуля Інше, тому що при використанні кількох версій заглушок для тестування оновлення немає ніяких труднощів у відтворенні тестових даних.

Тест за спаданням. Тест на спуск не є точно протилежним випробуванню підйому, але в першому наближенні його можна вважати таким. При низхідному підході програми збираються і тестуються «зверху вниз». Тільки основний модуль був протестований окремо. Після завершення тестування модуля по черзі підключіть безпосередньо викликані модулі (наприклад, редактор комунікацій) і перевірте отриману комбінацію. Повторюйте процес, поки всі модулі не будуть зібрані та перевірені. Цей підхід викликає два питання:

1. Що робити, якщо тестовий модуль викликає низькорівневий модуль (який на даний момент не існує)?

2. Як подаються дані тесту?

Відповідь на перше питання полягає в тому, що для імітації функціональності відсутнього модуля програмується модуль – заглушка, яка моделює функціональність відсутнього модуля. Цікавим є й друге питання: у якому вигляді готуються тестові дані і як вони передаються в програму? Якщо основний модуль містить усі необхідні операції введення та виведення, відповідь проста: тест записується у вигляді зовнішніх даних, загальних для користувача, і передається програмі через її виділений пристрій введення. Так, але це рідко. У добре розробленій програмі фізичні операції введення-виводу виконуються на нижчих рівнях структури, оскільки фізичний ввод-вивод є досить низькорівневою абстракцією. Тому для економічного вирішення проблеми модулі додаються не в строгому порядку спадання (всі модулі в одному горизонтальному шарі, потім модулі в наступному), а таким чином, що гарантує фізичну функціональність I. Операція /O виконується максимально швидко. При досягненні цієї мети низхідні тести мають значну перевагу: всі наступні тести готуються в тій самій формі, яка розроблена користувачем. Порівняно з висхідним метод низхідний має як переваги, так і недоліки. Найбільш істотна перевага полягає в тому, що цей підхід поєднує тестування модулів, тестування з’єднання та часткове тестування зовнішніх функцій. Ще одна перевага така ж: коли модулі вводу/виводу вже

підключені, тест можна легко підготувати. Знижуючий підхід також корисний, коли є сумніви щодо життєздатності загального плану або коли проект плану може мати серйозні недоліки. Перевагою підходу до зниження зазвичай вважається те, що драйвер не потрібен; замість драйвера вам потрібно лише написати

«вилка». На жаль, метод падіння тесту має деякі недоліки. Основна проблема полягає в тому, що модулі рідко ретельно перевіряються відразу після підключення. Насправді, дуже детальне тестування деяких модулів може вимагати надзвичайно складних плагінів. Програмісти часто вирішують не витрачати багато часу на їх програмування, а написати прості заглушки і перевірити лише деякі умови в модулі. Звичайно, він повернеться і закінчить тестування відповідних модулів Пізніше, коли вийде пробку. Такий план тестування, безумовно, не найкраще рішення, оскільки про умови затримки часто забувають. Другим тонким недоліком спадного підходу є те, що він може переконати в можливості розпочати програмування та тестування програм верхнього рівня до того, як вся програма буде повністю розроблена. Ідея на перший погляд може здатися економічною, але часто виявляється навпаки. Більшість опитаних дизайнерів визнали, що процес проектування є інтерактивним процесом. Перший проект рідко буває ідеальним. Звичайний стиль розробки структури програми полягає в тому, щоб повернутися назад і виправити верхній шар після того, як розроблено нижній шар, внести деякі покращення або виправити помилки, а іноді навіть відкинути проект і почати знову, тому що розробник раптом побачить кращий шлях. Якщо основна частина програми вже запрограмована та перевірена, будь-якому покращенню її структури буде серйозний опір. Зрештою, подібні вдосконалення часто можуть заощадити вам більше часу, ніж дні чи тижні, які розраховує отримати дизайнер, розпочавши програмування занадто рано.

4.2 Програмна реалізація тестування

Створюємо приклад тестової програми на мові JavaScript, яка буде використана для тестування функціонального додавання пристроїв, що зображена на (рисунку 4.1):

```
// Припустимо, що у вас є функція для відправки запитів на сервер/
// Приклад функції відправки POST-запиту на сервер
async function sendPostRequest(url, data) {
  try {
    const response = await axios.post(url, data);
    return response.data;
  } catch (error) {
    console.error('Error:', error);
    throw error;
  }
}

// Тестова програма для додавання пристрою
async function testAddDevice() {
  const testDevice = {
    name: 'Test Device',
    maxTemperature: 30,
    minTemperature: 10,
  };

  try {
    // Відправка POST-запиту для додавання пристрою
    await sendPostRequest('/api/devices', testDevice);

    // Після успішного додавання пристрою, вивести повідомлення
    console.log('Test device added successfully.');
  } catch (error) {
    console.error('Error adding test device:', error);
  }
```

Рисунок 4.1 - Тестування функції відправки POST-на сервер

Визначається функція sendPostRequest, яка використовується для відправки POST-запитів на сервер.

Функція testAddDevice тестує пристрій та створює функцію sendPostRequest для додавання цього пристрою на сервер.

Тестова програма запускається за допомогою виклику testAddDevice().

Успішне тестування може бути визначено, якщо тестові сценарії виконані без помилок, а система поводиться так, як очікується. У цьому конкретному тесті додавання пристрою (testAddDevice), успішне виконання

може бути підтверджено, якщо

Програма виводить повідомлення "Тестовий пристрій додано успішно" без повідомлення про помилки в консолі.

Новий пристрій успішно додається на сервер (ви можете перевірити це, отримавши відповідь від сервера або стан системи через інші інтерфейси).

Виконуються додаткові очікування, такі як правильне відображення списку пристроїв, коректність збереження параметрів пристрою та інші.

Невиявлення помилок може бути додатково підтверджено високою кришкою коду тестами, включаючи обробку граничних випадків та різних сценаріїв взаємодії.

Якщо під час тестування виникають непередбачувані помилки, це може бути ознакою неправильної реалізації або нестабільності системи.

Поведінка не відповідає очікуванню:

Якщо результати тестування не відповідають очікуванням (наприклад, новий пристрій не з’являється у списку пристроїв), це можна вказати на проблеми в логічній програмі.

Поганий дизайн інтерфейсу:

Якщо інтерфейс системи користувача не є зрозумілим чи незручним для використання, це також можна розглядати як недолік.

При проведенні тестування помилок не було система користувача та відправляє запити відповідно до умов коректної роботи.

5 ЕКОНОМІЧНИЙ РОЗДІЛ

5.1 Технологічний аудит результатів проведених досліджень нечітких регуляторів в системах автоматичного керування

Як було зазначено у попередніх розділах виконаної нами роботи, дослідження нечітких регуляторів в системах автоматичного керування є актуальною задачею, оскільки вони використовуються як інструмент для вирішення проблеми управління системами, що мають нечіткі, невизначені або змінні параметри, які не можуть бути точно описані класичними математичними моделями.

Тому метою наших досліджень було удосконалення нечітких регуляторів у системах автоматичного керування, що мало забезпечити підвищення ефективності та точності керування у різноманітних додатках, збільшення стійкості системи до змін параметрів, завад та зовнішніх впливів.

Для досягнення цієї мети нами було розв’язано низку задач, а саме: розроблено нечіткі моделі системи; розроблено алгоритми нечіткого керування, які враховують нечіткість вхідних даних; зроблено налаштування параметрів регуляторів, які забезпечують бажану динаміку системи; розроблено механізми адаптації нечітких регуляторів до змінних умов роботи системи; проведено експерименти для перевірки ефективності розроблених нечітких регуляторів; зроблено порівняння результатів роботи розроблених нечітких регуляторів з іншими методами керування: класичними ПІД-регуляторами і іншими адаптивними алгоритмами тощо.

В результаті було удосконалено нечіткі регулятори, які можуть використовуватися в системах автоматичного керування, і забезпечать підвищення стійкості роботи всієї системи, її надійність та ефективність.

Для встановлення комерційного потенціалу нашої розробки було проведено її технологічний аудит, для чого було запрошено 3-х експертів – відомих фахівців у цій галузі знань: кандидата технічних наук, доцента

Кривогубченка С.Г., кандидата технічних наук, професора Папінова В.М. та провідного фахівця Присяжнюка В.В.

Визначення потенційних можливостей комерційного використання результатів нашої розробки було здійснено за критеріями, наведеними в табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу будь-якої наукової розробки та бальна оцінка цих критеріїв

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти провели об’єктивне оцінювання результатів проведених нами досліджень нечітких регуляторів в системах автоматичного керування і звели свої експертні висновки в таблицю 5.2.

Таблиця 5.2 – Експертні висновки (результати) технологічного аудиту нашої розробки (за шкалою оцінювання 0-1-2-3-4)

Критерії	Прізвище, ініціали експертів		
	Кривогубченко С.Г.	Папінов В.М.	Присяжнюк В.В.
	Бали, що їх виставили експерти:		
1	3	3	3
2	4	4	4
3	4	3	4
4	3	4	3
5	3	3	4
6	4	4	3
7	3	3	4
8	3	3	4
9	3	4	3
10	4	3	4
11	3	4	4
12	4	4	3
Сума балів	$СБ_1 = 41$	$СБ_2 = 42$	$СБ_3 = 43$

Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{41+42+43}{3} = \frac{126}{3} = 42$

Встановлення комерційного потенціалу нашої розробки будемо здійснювати на основі рекомендацій, наведених в таблиці 5.3 [31].

Таблиця 5.3 – Рівні комерційного потенціалу будь-якої наукової розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу наукової розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Оскільки середньоарифметична сума балів, що їх виставили експерти, складає 42 бал, то це свідчить, що удосконалені нами нечіткі регулятори, які можуть використовуватися в системах автоматичного керування, мають рівень комерційного потенціалу, який вважається «вище середнього».

Це пояснюється тим, що удосконалені нами нечіткі регулятори дозволяють ефективніше вирішувати низку важливих завдань та розв’язувати проблеми, які важко або навіть неможливо розв’язати за допомогою класичних лінійних методів керування.

5.2 Розрахунок витрат на проведення досліджень нечітких регуляторів в системах автоматичного керування

При проведенні наших досліджень було зроблено такі витрати.

А). Основна заробітна плата $З_о$ розробників, яка визначається за формулою:

$$З_о = \frac{M}{T_р} \cdot t \text{ [грн]}, \qquad (5.1)$$

де М – місячний посадовий оклад розробника, грн; приймемо, що

М = (6700…22000) грн/місяць;

$Т_р$ – число робочих днів в місяці; приймемо $Т_р$ = 20 днів;

t – число днів роботи розробників.

Зроблені розрахунки зведемо до таблиці 5.4:

Таблиця 5.4 – Основна заробітна плата виконавців роботи

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи	20000	1000	20 годин	≈ 3333 (при 6-годинноиу робочому дні)
2. Магістрант-студент-виконавець	2000 (беремо 6700)	335	77	25795
3. Консультант з економічної частини	18000	900	1,5 години	225 (при 6-год. робочому дні)
Загалом				$З_о$ = 29353 грн

Б). Додаткова заробітна плата $З_д$ розробників розраховується як (10…12)% від величини їх основної заробітної плати, тобто:

$$З_д = \alpha \cdot З_о = (0{,}1...0{,}12) \cdot З_о . \qquad (5.2)$$

Приймемо, що α = 0,12. Тоді для нашого випадку отримаємо:

$З_д$ = 0,12 × 29353 = 3522,36 ≈ 3523 грн.

В). Нарахування на заробітну плату $НЗП_{зп}$ розробників розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \cdot \frac{\beta}{100}, \qquad (5.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. В 2023 (і мабуть у наступних роках) році β = 22%. Тоді:

$НЗН_{зп}$ = (29353 + 3523) × 0,22 = 7232,72 ≈ 7233 грн.

Г). Амортизація основних засобів А, які використовувались під час виконання даної роботи:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ [грн]}, \qquad (5.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань. Для нашого випадку можна прийняти, що $Н_а$ = (5...25)%;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн.	Норма амортизації, %	Термін використання, місяці	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання, принтер тощо	58000	20	4 (при 50% використанні)	≈1933
2. Приміщення факультету, університету, кафедри	36000	5	4 при 20% використанні	120
Всього				*А = 2053 грн*

Д). Витрати на матеріали М розраховуються за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ [грн].}, \quad (5.5)$$

де $Н_i$ – витрати матеріалу *і*-го найменування, кг; $Ц_i$ – вартість матеріалу *і*-го най-менування; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); $В_i$ – маса відходів матеріалу *і*-го найменування; $Ц_в$ – ціна відходів матеріалу *і*-го найменування; n – кількість видів матеріалів.

Е). Витрати на комплектуючі К розраховуються за формулою:

$$К = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i \text{ [грн]}, \quad (5.6)$$

де $Н_i$ – кількість комплектуючих *і*-го виду, шт.; $Ц_i$ – ціна комплектуючих *і*-го виду; $К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15); n – кількість видів комплектуючих.

Під час виконання роботи загальні витрати на матеріали та комплектуючі (за аналогією з попередніми розробками) склали приблизно 3000 грн.

Ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д}, \quad (5.7)$$

де В – вартість 1 кВт-год. електроенергії, в 2023 р. В ≈ 4,5 грн/кВт;

П – установлена потужність обладнання, кВт; П = 0,95 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 350 годин;

$К_п$ – коефіцієнт використання потужності; $К_п < 1 = 0,74$.

$К_д$ – коефіцієнт корисної дії, $К_д = 0,61$.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \cdot П \cdot Ф \cdot К_п}{К_д} = \frac{4,5 \cdot 0,95 \cdot 350 \cdot 0,74}{0,61} = 1815,13 \approx 1816 \text{ [грн].}$$

И). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати виконавців, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (5.8)$$

Для нашого випадку отримаємо:

$$В_{інш} = 1,5 \times 29353 = 44029,5 \approx 44030 \text{ [грн].}$$

К). Сума всіх попередніх статей витрат складає витрати на виконання нашої роботи (безпосередньо розробником-магістрантом) – В.

$$В = 29353 + 3529 + 7233 + 2053 + 3000 + 1816 + 44030 = 91014 \text{ грн.}$$

Л). Загальні витрати на проведення досліджень нечітких регуляторів в системах автоматичного керування $В_{заг}$ розраховуються за формулою:

$$В_{заг} = \frac{В}{\beta}, \quad (5.9)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи. Можна прийняти, що, $\beta \approx 0,7$, оскільки робота потребує ще незначного доопрацювання.

Тоді: $В_{заг} = \frac{91014}{0,7} = 130020$ грн або приблизно 130 тисяч [грн].

Тобто прогнозовані можливі загальні витрати на проведення досліджень нечітких регуляторів в системах автоматичного керування становлять приблизно 130 тисяч грн.

5.3 Розрахунок економічного ефекту від можливої комерціалізації нашої розробки

Як було зазначено в підрозділі 5.1, удосконалені нами нечіткі регулятори в системах автоматичного керування дозволяють ефективніше вирішувати низку важливих завдань та розв’язувати проблеми, які важко або навіть неможливо розв’язати за допомогою класичних лінійних методів керування.

Тому наша розробка, на наше переконання, буде користуватися значним попитом на ринку аналогічного продукту.

Оскільки станом на 01.01.2023 року подібні, але значно гірші за своїми функціональними можливостями системи коштують приблизно 15 тисяч грн, то нашу розробку (наші нечіткі регулятори) можна буде реалізовувати на ринку значно дорожче, принаймні аж за 20 тисяч грн, або на (20-15) = 5 тисяч грн дорожче.

Аналіз місткості ринку аналогічного продукту показує, що наразі цей попит становить приблизно 100 шт. на рік і цей попит буде стабільно зростати принаймні протягом 3-х років після виведення нашої розробки на ринок. Тобто, якщо удосконалені нами нечіткі регулятори будуть впроваджені з 1 січня 2025 року, то її результати будуть виявлятися протягом 2025-го, 2026-го та 2027-го років.

Прогноз зростання попиту на нашу розробку складає по роках:

а) 2024 р. – доробка системи;

б) 2025 р. – +100 шт. до базового року;

в) 2026 р. – +200 шт. до базового року.

г) 2027 р. – + 300 шт. до базового року.

Тоді можливе збільшення чистого прибутку $\Delta П_i$, що його може отримати потенційний інвестор від комерціалізації нашої розробки становитиме:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (5.10)$$

де $\Delta Ц_о$ – покращення основного якісного показника від впровадження результатів розробки у цьому році. Для нашого випадку це є просто нова ціна реалізації нашої розробки $\Delta Ц_о$ = 5 тисяч грн;

N – основний кількісний показник, який визначає обсяг діяльності у році до впровадження результатів розробки; N = 100 шт.;

ΔN – покращення основного кількісного показника від впровадження результатів розробки. Таке покращення становитиме по роках, відповідно: + 100, +200 та +300 шт. (до базового 2023 року);

$Ц_о$ – основний якісний показник, який визначає обсяг діяльності у році після впровадження результатів розробки, грн; $Ц_о$= 20 тисяч грн;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки; для нашого випадку n = 4;

λ – коефіцієнт, який враховує сплату податку на додану вартість; $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = (0,2...0,5), приймемо, що ρ = 0,5;

υ – ставка податку на прибуток. У 2023 році υ = 18%.

Тоді можливе зростання чистого прибутку $\Delta П_1$ для потенційного інвестора протягом першого року від можливого впровадження нашої розробки (2025 р.) становитиме:

$$\Delta П_1 = [5 \cdot 100 + 20 \cdot 100] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 854{,}13 \approx 855 \approx \text{тис. [грн].}$$

Можливе зростання чистого прибутку $\Delta П_2$ для потенційного інвестора від можливого впровадження нашої розробки протягом другого (2026) року складе:

$$\Delta П_2 = [5 \cdot 100 + 20 \cdot 200] \cdot 0{,}8333 \cdot 0{,}5 \cdot (1 - \frac{18}{100}) = 1537{,}43 \approx 1538 \text{ тисяч [грн].}$$

Можливе зростання чистого прибутку $\Delta П_3$ для потенційного інвестора від можливого впровадження нашої розробки протягом третього (2027) року складе:

$$\Delta П_З = [5 \cdot 100 + 20 \cdot 300] \cdot 0,8333 \cdot 0,5 \cdot (1 - \frac{18}{100}) = 2220,74 \approx 2221$$ тисячу [грн].

Тоді приведена вартість зростання всіх чистих прибутків від можливого впровадження нашої розробки (за всі роки) становитиме:

$$ПП = \sum_{1}^{т} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої роботи, грн;

т – період часу, протягом якого виявляються результати впровадженої роботи, роки. Для нашого випадку т = 4 роки;

τ – ставка дисконтування;

t – період часу від моменту початку проведення досліджень нечітких регуляторів в системах автоматичного керування (2023 рік) до отримання зростання можливих чистих прибутків від їх впровадження.

Розрахунки приведеної вартості зростання можливих отриманих чистих прибутків та економічної ефективності від впровадження удосконалених нечітких регуляторів в системах автоматичного керування проведемо для різних значень рівня інфляції в Україні.

Так, для «традиційного» рівня інфляції в країні τ = 10% (або τ = 0,1) приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, становитиме:

$$ПП = \frac{855}{(1+0,1)^2} + \frac{1538}{(1+0,1)^3} + \frac{2221}{(1+0,1)^4} \approx 707+1156+1517=3380$$ тисяч [грн].

Теперішня вартість інвестицій PV, що повинні бути вкладені для реалізації нашої розробки: $PV = (1,0…5) \times В_{заг}$.

Для нашого випадку $PV = 5 \times 130 = 650$ тисяч грн.

Абсолютний економічний ефект $Е_{абс}$ від можливих вкладених інвестицій може становити:

$$Е_{абс} = ПП – PV, \qquad (5.12)$$

де ПП – приведена вартість збільшення всіх чистих прибутків для інвестора від можливого впровадження нашої розробки, ПП = 3380 тисячі грн;

PV – теперішня вартість інвестицій PV = 650 тисяч грн.

Абсолютний ефект від можливого впровадження нашої розробки (з врахуванням витрат на її впровадження) за всі роки становитиме:

$Е_{абс}$ = 3380 – 650 = 2730 тисяч грн.

Далі розрахуємо внутрішню дохідність $Е_в$ вкладених інвестицій:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (5.13)$$

де $Е_{абс}$ – абсолютний ефект вкладених інвестицій; $Е_{абс}$ = 2730 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 650 тис. грн;

$Т_ж$ – життєвий цикл розробки, роки.

$Т_ж$ = 4 роки (2024-й, 2025-й, 2026-й, 2027-й роки)

Для нашого випадку отримаємо:

$$Е_в = \sqrt[4]{1 + \frac{2730}{650}} - 1 = \sqrt[4]{1 + 4,2} - 1 = \sqrt[4]{5,2} - 1 = 1,51 - 1 = 0,51 = 51,0\%.$$

Далі визначимо ту мінімальну дохідність, нижче за яку потенційному інвестору не вигідно буде займатися комерціалізацією нашої розробки.

Мінімальна дохідність або мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f, \quad (5.14)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,10...0,12);

f – показник, що характеризує ризикованість вкладень; f = (0,05...0,30).

Для нашого випадку отримаємо:

$$\tau_{мін} = 0,10 + 0,30 = 0,40 \text{ або } \tau_{мін} = 40\%.$$

Оскільки величина $Е_в$ = 51,0% > $\tau_{мін}$ = 40%, то потенційний інвестор у принципі може бути зацікавлений у фінансуванні та комерціалізації нашої розробки.

Для рівня інфляції в Україні у τ = 20% (або τ = 0,2) приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, становитиме:

$$ПП = \frac{855}{(1+0,2)^2} + \frac{1538}{(1+0,2)^3} + \frac{2221}{(1+0,2)^4} \approx 594+890+1071=2555 \text{ тисяч грн.}$$

Абсолютний ефект від можливого впровадження нашої розробки (з врахуванням витрат на її впровадження) за чотири роки складе:

$E_{абс}$ = 2555 – 650 = 1905 тисяч грн.

Далі розрахуємо внутрішню дохідність $E_в$ вкладених інвестицій:

$$E_в = \sqrt[T_ж]{1+\frac{E_{абс}}{PV}} - 1, \qquad (5.13)$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 1905 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 650 тис. грн;

$T_ж$ – життєвий цикл розробки, роки.

$T_ж$ = 4 роки (2024-й, 2025-й, 2026-й, 2027-й роки)

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1+\frac{1905}{650}} - 1 = \sqrt[4]{1+2,9307} - 1 = \sqrt[4]{3,9307} - 1 = 1,408 - 1 = 0,408 = 40,8\%.$$

Оскільки величина $E_в$ = 40,8% > $\tau_{мін}$ = 40%, то потенційний інвестор у принципі також може бути зацікавлений у комерціалізації нашої розробки.

Для рівня інфляції в Україні у τ = 30% (або τ = 0,3) приведена вартість зростання всіх можливих чистих прибутків ПП, що їх може отримати потенційний інвестор від комерціалізації нашої розробки, становитиме:

$$ПП = \frac{855}{(1+0,3)^2} + \frac{1538}{(1+0,3)^3} + \frac{2221}{(1+0,3)^4} \approx 506+700+778=1284 \text{ тисяч грн.}$$

Абсолютний ефект від можливого впровадження нашої розробки (з врахуванням витрат на її впровадження) за чотири роки складе:

$E_{абс}$ = 1284 – 650 = 634 тисяч грн.

Далі розрахуємо внутрішню дохідність $E_в$ вкладених інвестицій:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.13)$$

де $E_{абс}$ – абсолютний ефект вкладених інвестицій; $E_{абс}$ = 634 тис. грн;

PV –теперішня вартість початкових інвестицій PV = 650 тис. грн;

$T_ж$ – життєвий цикл розробки, роки.

$T_ж$ = 4 роки (2024-й, 2025-й, 2026-й, 2027-й роки)

Для нашого випадку отримаємо:

$$E_в = \sqrt[4]{1 + \frac{634}{650}} - 1 = \sqrt[4]{1 + 0,9754} - 1 = \sqrt[4]{1,9754} - 1 = 1,185 - 1 = 0,185 = 18,5\%.$$

Оскільки величина $E_в$ = 18,5% < $\tau_{мін}$ = 40%, то потенційний інвестор з великою вірогідністю (!) не буде зацікавлений у комерціалізації нашої розробки.

Залежність величини внутрішньої дохідності вкладених потенційних інвестицій від рівня інфляції в Україні наведено нами на рис. 5.1

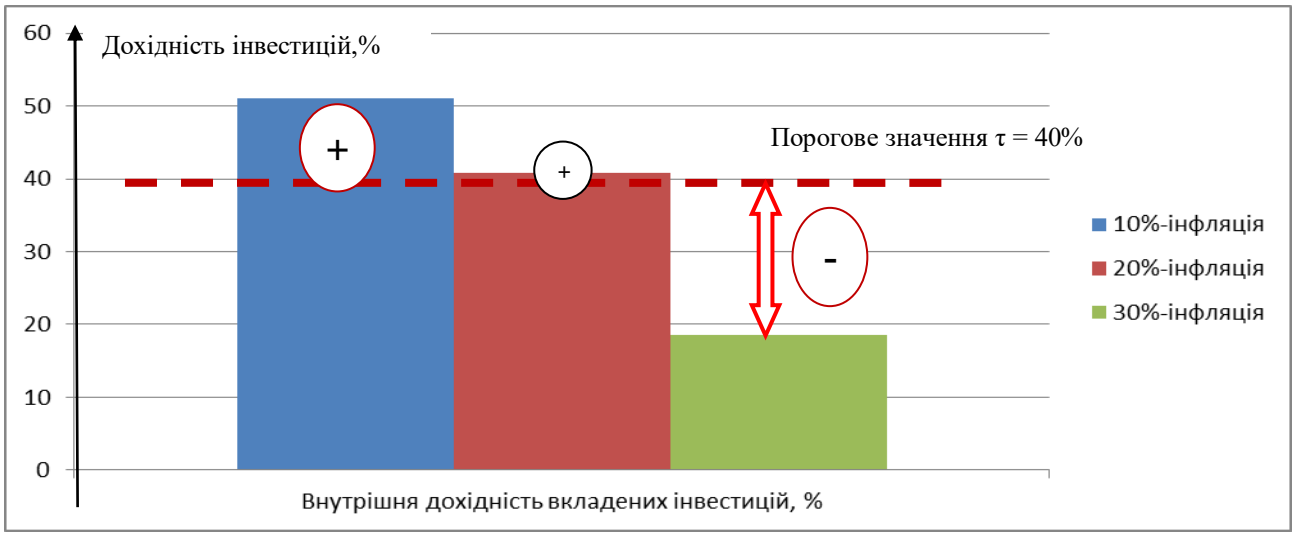

Рисунок 5.1 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій від рівня інфляції в країні (для рівня інфляції 10%, 20% та 30%)

Аналіз діаграм на графіку 5.1 показує, що комерціалізація удосконалених нами нечітких регуляторів в системах автоматичного керування може забезпечити дохідність вкладених інвестицій у 51% (при

рівні інфляції у 10%) та 40,8% (при рівні інфляції у 20%), тобто при таких рівнях інфляції потенційний інвестор може бути зацікавлений у комерціалізації нашої розробки. При рівні інфляції у 30% потенційний інвестор отримає дохідність вкладених коштів у 18,5%, що є значно нижче за порогову норму дохідності, яка становить 40%. Тому комерціалізація нашої розробки буде для інвестора проблематичною. Для прийняття остаточного рішення потрібно провести додаткові обгрунтування і розрахунки (наприклад, підняти ціну реалізації нашої розробки тощо).

Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Показники	Задані у ТЗ	Досягнуті у магістерській кваліфікаційній роботі	Висновок
1. Витрати на розробку	Не більше 150 тисяч грн	130 тисяч грн.	Досягнуто
2. Абсолютний ефект від впровадження розробки, тисяч грн	Не менше 2500 тисяч грн за 4 роки	2730 тисяч грн при рівні інфляції у 10%	Виконано
3. Внутрішня дохідність інвестицій, %	Не менше 40%	51,0% при 10% інфляції; 40,8% – при 20% інфляції	Досягнуто

Таким чином, основні техніко-економічні показники удосконалених нами нечітких регуляторів в системах автоматичного керування, визначені у технічному завданні, виконані .

ВИСНОВКИ

У ході вивчення теми "Синтез нечітких регуляторів в системах автоматичного керування" та розгляду питань, пов'язаних із синтезом нечітких регуляторів, автоматичним керуванням температурою та іншими аспектами, було показано і досліджено різні аспекти цієї теми.

Сучасні технології вимагають ефективних систем керування для оптимізації процесів та покращення функціональності. Використання нечітких регуляторів у системах автоматичного керування може бути актуальним рішенням для досягнення цих цілей, більше 10% користувачів здійснюють взаємодію зі сферою нечітких регуляторів.

Переваги та недоліки нечітких регуляторів у системах автоматичного керування температурою водонагрівача:

Переваги включають адаптивність до змінних умов, здатність працювати без точних математичних моделей та ефективність в умовах нечіткості. Недоліки можуть включати складність підбору параметрів та обчислювальну складність.

Розглянуто застосування нечітких регуляторів у системі автоматичного регулювання температури водонагрівача. Це може бути тільки для оптимізації енергоспоживання та забезпечення комфортних умов для користувачів.

Розглянуто приклади конкурентів, таких як Siemens, Honeywell та Schneider Electric, з аналізом їхніх переваг та недоліків.

Проведено аналіз Siemens, Honeywell та Schneider Electric в контексті розробки нечітких регуляторів та системи автоматичного регулювання температури. Визначено їхні переваги та недоліки.

Виявлено, що синтез нечітких регуляторів у системах автоматичного керування є перспективним напрямком, особливо в умовах нечіткості та змінних умов.

Розглянуті аспекти програмування та веб-розробки, такі як вибір мови

програмування (JavaScript), використання середовища розробки (VS Code), проектування структури бази даних та створення фронтенду для веб-системи.

Представлено приклад тестової програми для додавання пристроїв у систему керування температурою.

Робота вивчає та аналізує ключові аспекти синтезу нечітких регуляторів, виявляє їхні переваги та недоліки, розглядає приклад використання в реальних системах та розглядає конкурентний ринок. Також надається практичний приклад програмування та тестування системи автоматичного керування температурою.

Ця робота дозволяє отримати загальність про важливі аспекти в галузі автоматичного керування та нечітких регуляторів, які можуть бути використані для підвищення ефективності та функціональності системи.

Визначена економічна доцільність розробленого пристрою, враховуючи покращення продуктивності та зниження витрат на процес створення пристрою. Економічні розрахунки показують що розроблений пристрій є конкурентно здатний на ринку аналогів (орієнтовна собівартість пристрою для масового виробництва 15 тисячі гривень).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Нейромережеві технології в системах управління: Підручник для візів./ Б. І. Кузнецов, та інші; Укр. інж.-пед. акад.. - Харків: УІПА, 2014. - 232 с.

2) Кирик В. В. Математичний апарат штучного інтелекту в електроенергетичних системах:підручник..– Київ: КПІ ім. Ігоря Сікорського, Видво «Політехніка», 2019. –224 с.

3) Цикл практичних робіт із дисципліни «Теорія нечітких множин».

URL : http://lib.chdu.edu.ua/pdf/dovidnuku/15/4.pdf

4) Субботін С. О., Олійник А.О., Субботін С.О. Нейронні мережі: навч. посібн. – Запоріжжя ЗНТУ, 2014. – 132 с.

5) Тимощук П.В. Штучні нейронні мережі; Навч. посібн. - Львів: Львівська політехніка, 2011. - 444 с.

6) Лозинський А. О., Демків Л. І. Аналіз стійкості систем з регулятором Такагі Сугено // Штучний Інтелект. – 2008. – № 4. – 545–550 с.

7) Лозинський А. О., Демків Л. І. Аналіз стійкості систем з регулятором Такагі Сугено-Канга // Вісн. НТУ «ХПІ», серія «Електротехніка, електроніка і електродвигун». – 2008. – Вип. 30. – 89–90 с.

8) Лайза К., Джанет Г. Гнучке тестування: практичний посібник для тестувальників ПЗ та гнучких команд / Лайза Кріспін, Джанет Грегорі. - "Вільямс", 2010. - 464 с.

9) Демків Л. І. Аналіз стійкості системи з фаззі регуляторами частотними методами // Тематичний випуск «Проблеми автоматизованого електроприводу. Теорія і практика» науково-технічного журналу «ЕЛЕКТРОІНФОРМ» – Львів: ЕКОінформ. – 2009. – 435–436 с.

10) Мета і функції тестування і тестів URL: https://studfiles.net/preview/2227615/page:4/.

11) Лозинський О. О., Демків Л. І. Умовно стійкі системи з фазі-регулятором // Штучний інтелект. - 2010. - № 4. - 415-420 с.

12) Лозинський А. О., Демків Л. І. Дослідження стійкості систем з нестійкою підсистемою // Електротехніка та електроенергетика. – 2010. – № 1. – 19–29 с.

13) . Lozynskyi A., Demkiv L. Forming of the controlled influence in the system with fuzzy regulator // Comp.Probl. Of Elec.Eng. – Lviv:LPNU, 2011. – Vol. 1. – 27–34 p.

14) Лозинський А. О., Демків Л. І. Дослідження впливу виду функції належності на динамічні показники системи при багатокритеріальні оптимізації // Електротехнічні та комп'ютерні системи. – 2012. – № 5. – 137–144 с.

15) Коли тестування повинно бути автоматизованим? URL: https://www.stickyminds.com/article/when-should-test-be-automated

16) Практичний досвід автоматизованого тестування URL: http://www.methodsandtools.com/archive/archive.php?id=33.

17) Демків Л. І. Дослідження впливу параметрів функції належності на якісні показники функціонування системи з двома коренями в правій півплощині // Вісник Нац. ун-ту «Львівська політехніка». – 2012. – № 736. – 36–43 с.

18) Демків Л. І. Дослідження двомасової системи, що складається з двох підсистем, при дії зовнішніх збурень // Електромеханічні і енергозберігаючі системи. – 2012. – № 3. – 505–506 с.

19) Lozynskyy A., Demkiv L. Investigation of multicriteria optimal control with time-variable weight coefficients // Electrical Review. – 2013. –№2a. – 195–198 с.

20) Демків Л. І. Дослідження впливу методу дефазифікації на характеристики системи з нечітким регулятором Такагі-Сугено // Вісник Нац. ун-ту «Львівська політехніка», серія «Електроенергетичні та електромеханічні системи». – 2013. – № 763. – 34–39 с.

21) Л.І. Демків Вплив вибору стандартної лінійної форми на характеристики динамічної системи з нечітким модальним регулятором //

Вісник Нац. ун-ту «Львівська політехніка», серія «Електроенергетичні та електромеханічні системи». – 2014.–№ 785. – 20–33 с.

22) Demkiv L. I. Research of dynamic system with unstable subsystem that has one root in the right half-plane // Mathematical modeling and computing, Vol. 1, No. 2, (2014), P. 156–162 .

23) Лозинський А. О., Демків Л. І. Застосування нечіткої моделі системи при синтезі системи автоматичного керування нелінійними об’єктами // Електромеханічні і енергозберігаючі системи. – 2015. – № 2(30). – 24–30 с.

24) Lozynskyy A., Demkiv L. Application of dynamic systems family for synthesis of fuzzy control with account of non-linearities // Advances in electrical and electronic engineering, volume 14, number: 5, 2016, P. 543–550 .

25) Lozynskyy A., Demkiv L. Synthesis of fuzzy logic controller of nonlinear dynamic system with variable parameters // Computational problems of electrical engineering Vol. 6, No. 2, 2016, P. 91–98.

26) Demkiv L., Lozynskyy A., Lozynskyy O. and Demkiv I. A new approach to dynamical system’s fuzzy controller synthesis: Application of the unstable subsystem, 2017 International Conference on Modern Electrical and Energy Systems (MEES), Kremenchuk, 2017, P. 84–87.

27) Vantsevich V. V., Lozynskyy A., Demkiv L., Holovach I. Fuzzy Logic Control of Agile Dynamics of a Wheel Locomotion Module, 25th International Symposium on Dynamics of Vehicles on Roads and Tracks, taking place at Central Queensland University Rockhampton, Queensland, from 14–18 August 2017.

28) Vantsevich V. V., Lozynskyy A., Demkiv L. A wheel rotational velocity control strategy for an open-link locomotion module, 19th International & 14th EuropeanAfrican Regional Conference of the ISTVS, Budapest, Hungary, 25–27 September 2017.

29) Демків Л. І. Дослідження стійкості систем з нечіткими регуляторами // Автоматика – 2010: Матеріали міжнародної конференції з автоматичного управління. – Харків: ХНУРЕ, 2010 – 203–205 с.

30) Lozynskyy A. Demkiv L. Stability analysis of dynamical system with variable coefficients and fuzzy controller//CPEE ‘2015: Proc.Int.Conf. – Lviv, Ukraine, September 2-5, 2015. – P. 99–102.

31) Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А

(обов'язковий)

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖЕНО
Зав. кафедри АІІТ,
д.т.н., проф.
_____Олег БІСІКАЛО

" __ " _____ 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерської кваліфікаційну роботу

Синтез нечітких регуляторів в системах автоматичного керування.

08-31.МКР.003.02.000 ТЗ

Керівник к.т.н. доцент кафедри АІІТ

_____ Володимир ГАРМАШ

« _____» _____ 2023 р.

Розробив студент гр. 1АКІТ-22м

_____ Іван ГНАТЮК

«_____» _____ 2023 р.

Вінниця ВНТУ 2023

1. Назва та галузь застосування

1.1. Назва – синтез нечітких регуляторів у системах автоматичного керування.

1.2. Галузь застосування – Сфера загального користування.

2. Підстава проведення для розробки.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2023р., та індивідуальне завдання на МКР, затверджене протоколом №__ засідання кафедри АІІТ від «__» _____ 2023р.

Термін виконання робіт:

3. Мета та призначення розробки.

Методом розробки є створення системи для нечіткого регулятора в системах автоматичного керування.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1.Кирик В. В. Математичний апарат штучного інтелекту в електроенергетичних системах:підручник..– Київ: КПІ ім. Ігоря Сікорського, Видво «Політехніка», 2019. –224 с.

2. Цикл практичних робіт із дисципліни «Теорія нечітких множин». Електронний ресурс // Режим доступу: http://lib.chdu.edu.ua/pdf/dovidnuku/15/4.pdf

3. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт. / Укладачі В.О. Козловський, О.Й. Лесько, В.В.Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

5 Технічні дані.

5.1. Програма повинна відповідати наступним критеріям:

має форму реєстрації та вибір девайсів;

має зручну форму для введення необхідних даних;

якісна система захисту даних та синхронізація з усіма пристроями.

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- Операційна система Windows
- Центральний процесор core i5
- Оперативна пам`ять 8 гб
- Відеокарта 1 гб
- Жорсткий диск 200гб

5.2.2. Умови експлуатації системи:

- робота на веб ресурсах;
- можливість цілодобового функціонування системи;
- дані оновлюються і керуються з будь-якої відстані.
-

6 Економічні показники.

До економічних показників входять:

-витрати на розробку 130 тисяч гривень

-мінімальна дохідність 42,5% при 10% інфляції

-термін окупностіне більше 3-х років

7 Стадії розробки.

а) Аналіз та специфікація вимог	06.09 – 13.09
б)Проектування системи	13.09 – 22.10
в)Програмна реалізація	23.10 – 30.10
г) Тестування да дослідна експлуатація	31.10 – 18.11
д)Оформлення пояснювальної записки	19.11 – 24.11

е)Підтвердженння економічної доцільності 25.11 – 04.12

є)Остаточний захист роботи 05.12 – 14.12

8 Порядок контролю та приймання.

Рубіжний контроль провести до «___» _____ 2023 р.

Попередній захист МКР провести «___» _____ 2023 р.

Захист МКР провести до «___» _____ 2023 р.

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

Синтез нечітких регуляторів в системах автоматичного керування

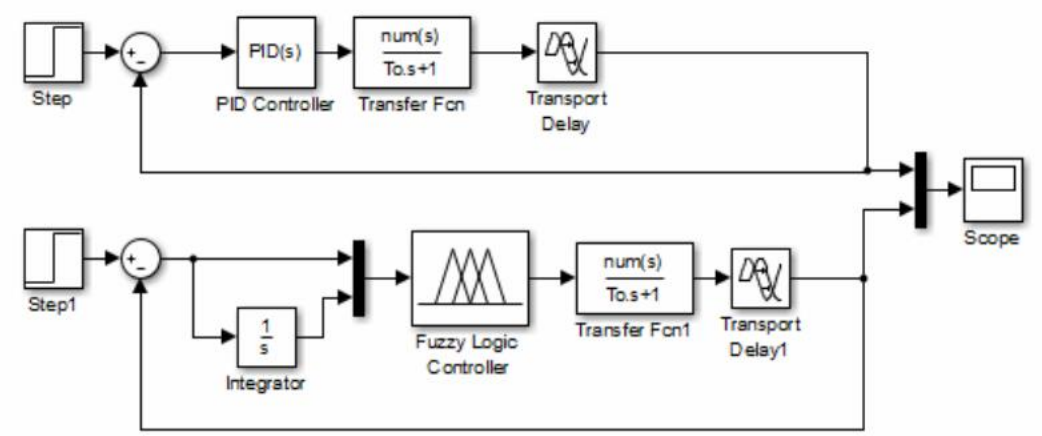

Рисунок Б.1 - Simulink-модель для дослідження нечітких регуляторів

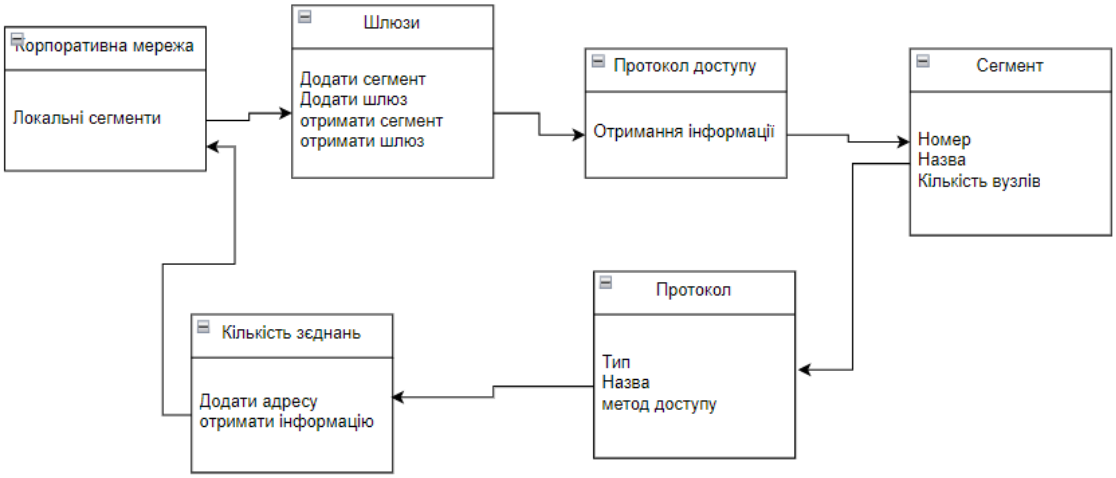

Рисунок Б.2 - UML-object diagram корпоративної мережі застосунку

Рисунок Б.3 - Функціональна схема САР.

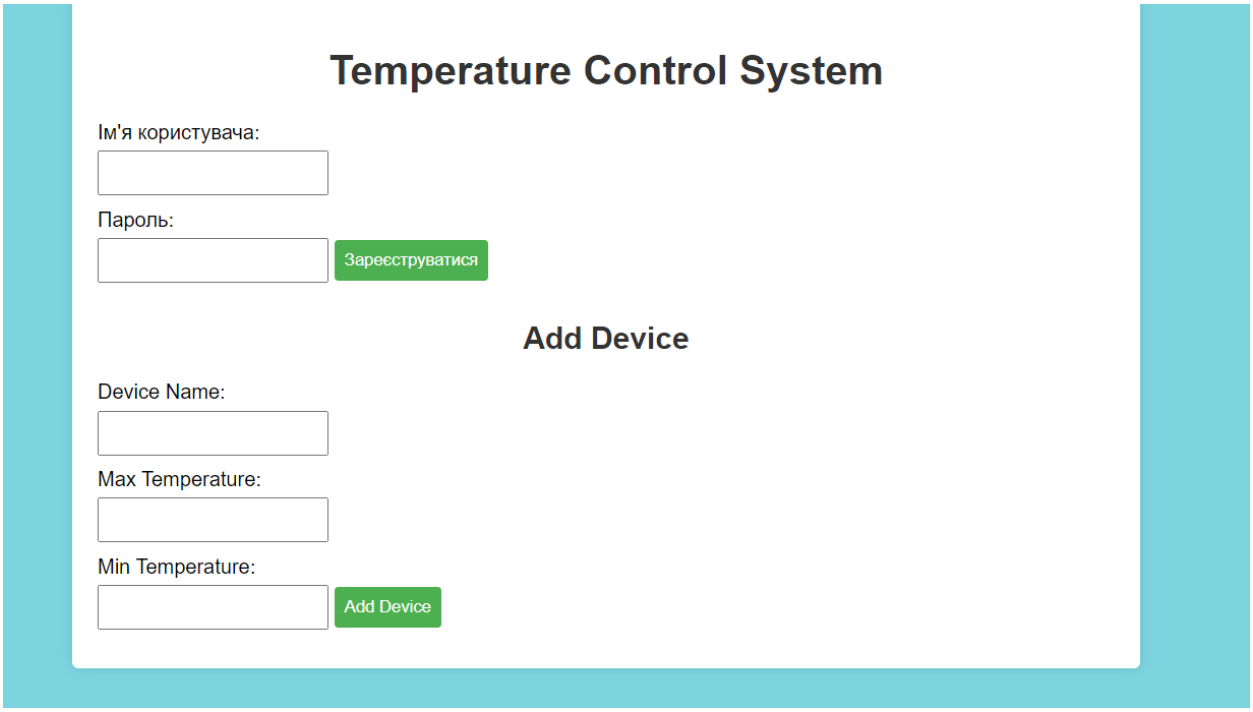

Рисунок Б.4 – Форма реєстрації web-застосунку

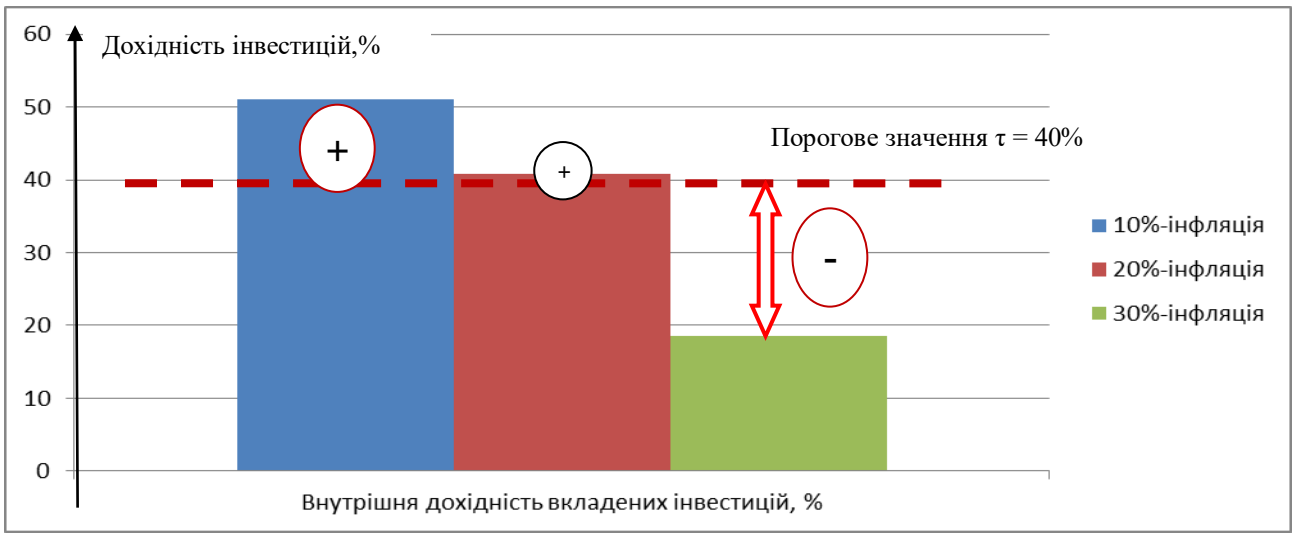

Рисунок Б.5 – Моделювання залежності величини внутрішньої дохідності потенційних інвестицій від рівня інфляції в країні (для рівня інфляції 10%, 20% та 30%)

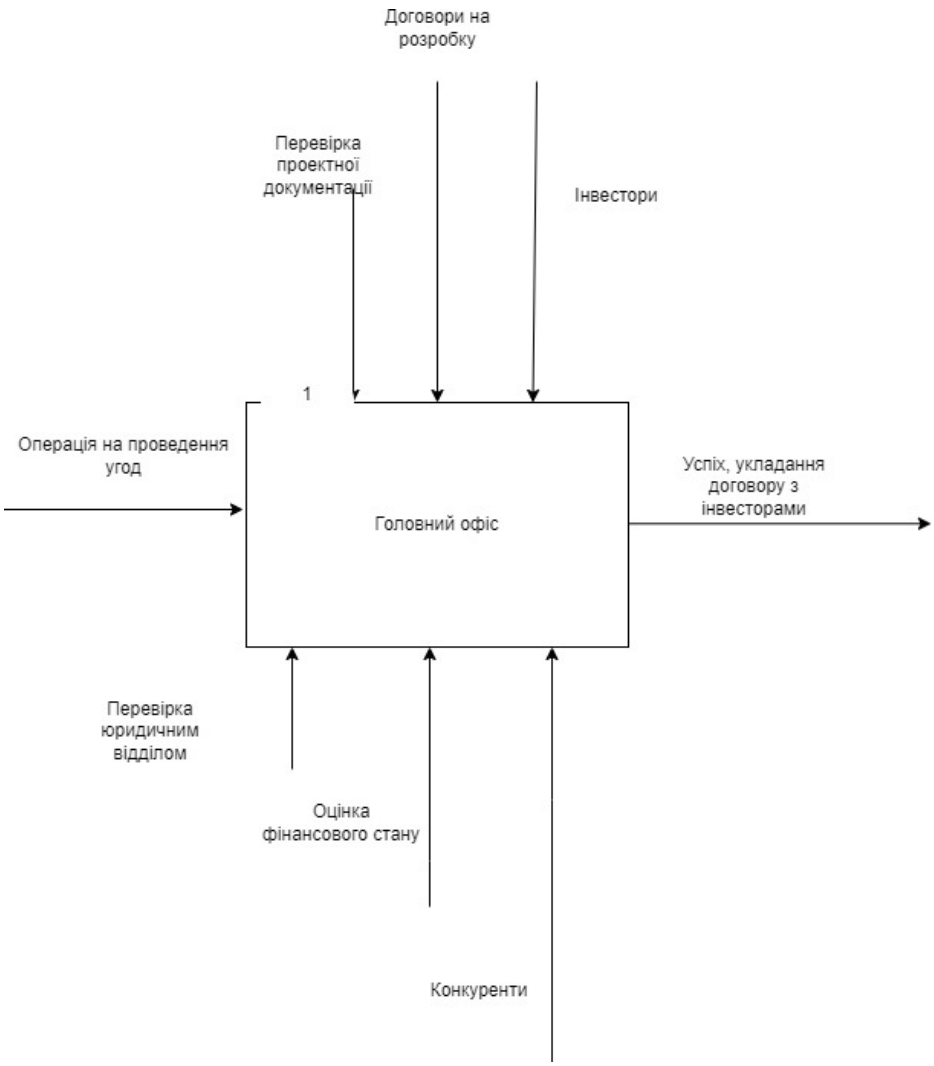

Рисунок Б.6 - IDEF0 першого рівня для процесу залучення інвесторів

Додаток В

(обов'язковий)

Лістинг програми

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Temperature Control System</title>
  <link rel="stylesheet" href="index.css">
</head>
<body>
  <div class="container">
    <h1>Temperature Control System</h1>

    <div id="devices-container">
      <!-- Device list will be displayed here -->
    </div>

    <form id="add-device-form">
      <h2>Add Device</h2>
      <label for="device-name">Device Name:</label>
      <input type="text" id="device-name" required>
      <label for="max-temperature">Max Temperature:</label>
      <input type="number" id="max-temperature" required>
      <label for="min-temperature">Min Temperature:</label>
      <input type="number" id="min-temperature" required>
      <button type="submit">Add Device</button>
    </form>

    <div id="temperature-log-container">
      <!-- Temperature log will be displayed here -->
    </div>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
  <script src="app.js"></script>
</body>
</html>
```

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #7cd4df;
  }

  .container {
    max-width: 800px;
    margin: 50px auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  }

  h1, h2 {
    text-align: center;
    color: #333;
  }

  form {
    margin-top: 20px;
  }

  label {
    display: block;
    margin-bottom: 5px;
  }

  input, button {
    margin-bottom: 10px;
    padding: 8px;
  }

  button {
    background-color: #4caf50;
    color: #fff;
    cursor: pointer;
    border: none;
    border-radius: 3px;
  }

  button:hover {
    background-color: #45a049;
  }



// Suppose you have a function to send requests to the server/
// An example of the function of sending a POST request to the server
```

```
async function sendPostRequest(url, data) {
  try {
    const response = await axios.post(url, data);
    return response.data;
  } catch (error) {
    console.error('Error:', error);
    throw error;
  }
}

// Test program for adding a device
async function testAddDevice() {
  const testDevice = {
    name: 'Test Device',
    maxTemperature: 30,
    minTemperature: 10,
  };

  try {
    // Send a POST request to add the device
    await sendPostRequest('/api/devices', testDevice);

    // After successfully adding the device, display a message
    console.log('Test device added successfully.');
  } catch (error) {
    console.error('Error adding test device:', error);
  }
}

// Start the test program
testAddDevice();
const express = require('express');
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

const app = express();
const PORT = process.env.PORT || 3000;

app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/temperature_control', { useNewUrlParser: true,
useUnifiedTopology: true });

const deviceSchema = new mongoose.Schema({
  name: String,
  maxTemperature: Number,
  minTemperature: Number,
});

const temperatureLogSchema = new mongoose.Schema({
  device: { type: mongoose.Schema.Types.ObjectId, ref: 'Device' },
  temperature: Number,
```

```
  timestamp: { type: Date, default: Date.now },
});

const Device = mongoose.model('Device', deviceSchema);
const TemperatureLog = mongoose.model('TemperatureLog', temperatureLogSchema);

app.post('/api/devices', async (req, res) => {
  try {
    const device = await Device.create(req.body);
    res.json(device);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.get('/api/devices', async (req, res) => {
  try {
    const devices = await Device.find();
    res.json(devices);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.get('/api/temperature-log', async (req, res) => {
  try {
    const temperatureLog = await TemperatureLog.find().sort('-
timestamp').limit(10).populate('device');
    res.json(temperatureLog);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Додаток Г

(обов'язковий)

Протоколи перевірки магістерської кваліфікаційної роботи за наявності текстових запозичень

Назва роботи: «Інформаційна технологія прогнозування курсу біткоїна»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра АІІТ

Показники звіту подібності Unicheck

Оригінальність 97,8% Схожість 2,2%

Аналіз звіту подібності (відмітити потрібне)

- **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**
- Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
- Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Роман МАСЛІЙ
(підпис) (ім'я, прізвище)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи _____ Іван ГНАТЮК
(підпис) (ім'я, прізвище)

Керівник роботи _____ Володимир ГАРМАШ
(підпис) (ім'я, прізвище)