МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Інтелектуальна система управління теплицями на основі генетичного алгоритму»**

Виконав: студент 4 курсу групи Іт-41сп

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Олійник Віталій Володимирович
(Прізвище та ініціали)

Керівник: д.т.н., професор Тригуба А.М.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Кригуль Р.Є.
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А. М. Тригуба
«____» _____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Олійнику Віталію Володимировичу

1. Тема роботи: «Інтелектуальна система управління теплицями на основі генетичного алгоритму»

Керівник роботи Тригуба Анатолій Миколайович, професор
затверджені наказом по університету від 27.11.2023 року № 641/к-с.

2. Строк подання студентом роботи 10.06.2024 р.
3. Вихідні дані до роботи: *вимоги до процесів управління теплицями; засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, методика використання генетичного алгоритму.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз предметної області.
2. Математичний опис задачі проекту та вибір засобів для реалізації проекту.
3. Проектування інтелектуальної системи управління теплицями із використанням поновлюваних джерел енергії.
4. Реалізація інтелектуальної системи управління теплицями із використанням генетичного алгоритму.
5. Охорона праці.
Висновки та пропозиції.
Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень): *особливості управління теплицями; огляд інтелектуальних систем аналогів; результати вибору засобів реалізації проекту управління теплицями; принципова схема системи керування теплицями; архітектура системи керування теплицями; модуль для створення діалового вікна користувача інтелектуальної системи; модуль контролю для реалізації PID-алгоритмів і генетичного алгоритму; модуль для обробки даних, інтеграції алгоритмів і роботи з базою даних SQLite; особливості практичного використання інтелектуальної системи.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Тригуба А.М., зав. кафедри ІТ*		
5	*Тимочко В.О., доцент кафедри фізики, інженерної механіки та безпеки виробництва*		

7. Дата видачі завдання 27 листопада 2023 р.

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*27.11-31.12.23*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*01.02-05.03.24*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*06.03-14.04.24*	
4.	*Виконання четвертого розділу та аркушів ілюстраційного матеріалу до нього*	*15.04-10.05.24*	
5.	*Написання розділу «Охорона праці»*	*11.05-23.05.24*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*24-31.05.24*	
7.	*Завершення роботи вцілому*	*01-10.06.24*	

Студент _____ Олійник В.В.
(підпис)

Керівник роботи _____ Тригуба А.М.
(підпис)

УДК 004.9 : 631.1

Інтелектуальна система управління теплицями на основі генетичного алгоритму.

Олійник В.В. Кафедра ІТ – Дубляни, Львівський НУП, 2024.

Кваліфікаційна робота: 77с. текст. част., 15 рис., 3 табл., 14 арк. ілюстраційного матеріалу, 36 джерел.

Подано особливості інформаційних систем управління теплицями. Наведені основні характеристики інтелектуальних систем для теплиць. Проведено огляд існуючих інтелектуальних систем управління сільськогосподарськими теплицями. Виконано аналіз розробок інтелектуальних систем управління із використанням генетичних алгоритмів.

Подано особливості використання генетичних алгоритмів для вирішення практичних задач. Здійснено математичний опис генетичного алгоритму з адаптивним генетичним оператором. Запропоновано використання у теплицях додаткової системи сонячного та вітрового електропостачання та нечіткого підходу до управління теплицями. Здійснено вибір засобів для виконання проекту.

Розроблено принципову схему системи керування теплицями. Запропоновано архітектуру системи керування теплицями. Розроблено блок-схему удосконаленого адаптивного генетичного алгоритму.

Здійснена розробка модуля для створення діалового вікна користувача інтелектуальної системи. Розроблено модуль контролю для реалізації PID-алгоритмів і генетичного алгоритму. Створено модуль для обробки даних, інтеграції алгоритмів і роботи з базою даних SQLite. Описано використання інтелектуальної системи управління теплицями на основі генетичного алгоритму. Розроблено інструкцію з охорони праці під час використання інтелектуальної системи управління теплицями.

ЗМІСТ

ВСТУП

Сьогодні сільське господарство стикається з низкою викликів, таких як зростання населення, зміна клімату та обмежені ресурси. Для подолання цих викликів важливо впроваджувати нові технології, які допоможуть підвищити продуктивність, ефективність та стійкість сільського господарства. Водночас спостерігається, що дослідження у сфері сільського господарства постійно розвиваються, особливо завдяки використанню новітніх технологій. Інтеграція інтелектуальних систем управління стає ключовим фактором у забезпеченні стабільного та ефективного вирощування сільськогосподарської продукції [6]. У цьому напрямі виникає великий інтерес до розробки і застосування інтелектуальних систем управління теплицями.

Інтелектуальні системи управління (ІСУ) є одним із перспективних напрямків розвитку сільського господарства. ІСУ можуть збирати та аналізувати дані про навколишнє середовище, стан рослин та інші фактори, що впливають на ріст і розвиток рослин. Ці дані можуть використовуватися для прийняття оптимальних рішень щодо поливу, освітлення, опалення, вентиляції та інших параметрів середовища в теплицях.

Наша кваліфікаційна робота ставить за мету виконати обґрунтування складових та розробка інтелектуальної системи управління теплицями на основі генетичного алгоритму. Використання генетичних алгоритмів у сільському господарстві відкриває широкі перспективи для оптимізації процесів, підвищення врожайності та зниження витрат.

Генетичні алгоритми (ГА) є одним із методів штучного інтелекту, які можуть використовуватися для розробки ІСУ [4]. ГА – це евристичні методи пошуку, які імітують природний процес еволюції. ГА можуть генерувати та оцінювати безліч можливих рішень проблеми, а потім вибирати найкращі з них.

Під час виконання роботи розглянуто теоретичні аспекти інтелектуальних систем управління, а також принципи функціонування генетичних алгоритмів. Особлива увага буде приділена адаптації генетичних алгоритмів до конкретних

умов управління теплицями та їх ефективному застосуванню для оптимізації процесів вирощування рослин.

В результаті виконання роботи розроблено прототип інтелектуальної системи управління теплицями, що базується на генетичному алгоритмі, а також перевірено його ефективність та порівняння з існуючими методами управління.

Виконана робота має важливе значення для розвитку сучасного сільського господарства, сприяючи підвищенню продуктивності та стабільності виробництва, а також зменшенню негативного впливу на довкілля.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Особливості інформаційних систем управління теплицями

У більшості тепличних комплексів працівники вручну збирають дані щодо функціонування теплиць і формують їх у файлах Excel для подальшого аналізу та рекомендацій. Цей тип збору даних вимагає від виробників багато ручної роботи, що призводить до затримки із бажаними рекомендаціями стосовно діяльності. Це є проблемою для власників теплиць, оскільки будь-які задачі з кліматичними умовами в приміщенні вимагають швидкого реагування, щоб уникнути шкоди посівам у теплицях. Окремі виробники вимушені шукати зручний спосіб для швидшого збирання даних з теплиць за допомогою створення додатків для розумних теплиць.

Щоб розробити автоматизовану тепличну систему для моніторингу теплиць і подальшого надання точних і швидких рекомендацій агропідприємствам потрібно проводити додаткові науково-дослідні роботи.

Рисунок 1.1 – Використання автоматизованих тепличних систем для моніторингу стану рослин у теплиць [12]

Окремі автори [12] розробили веб-додаток для зручного моніторингу та керування в приміщенні теплиці, збираючи вимоги для вирішення конкретних потреб виробників. Під час дослідження користувачів визнали необхідність автоматично вставляти зібрані дані з нового програмного забезпечення для моніторингу теплиць прямо в уніфіковану платформу управління теплицями, а також обмінюватися даними між виробниками, щоб прискорити реагування на зміни клімату, покращити розвиток рослин та знайти оптимальні умови для зростання врожаю.

Сучасні інтелектуальні інформаційні системи теплиць використовуються для моніторингу, збору даних про мікрокліматичні умови та терміни, які впливають на ріст культур (рис. 1.2).

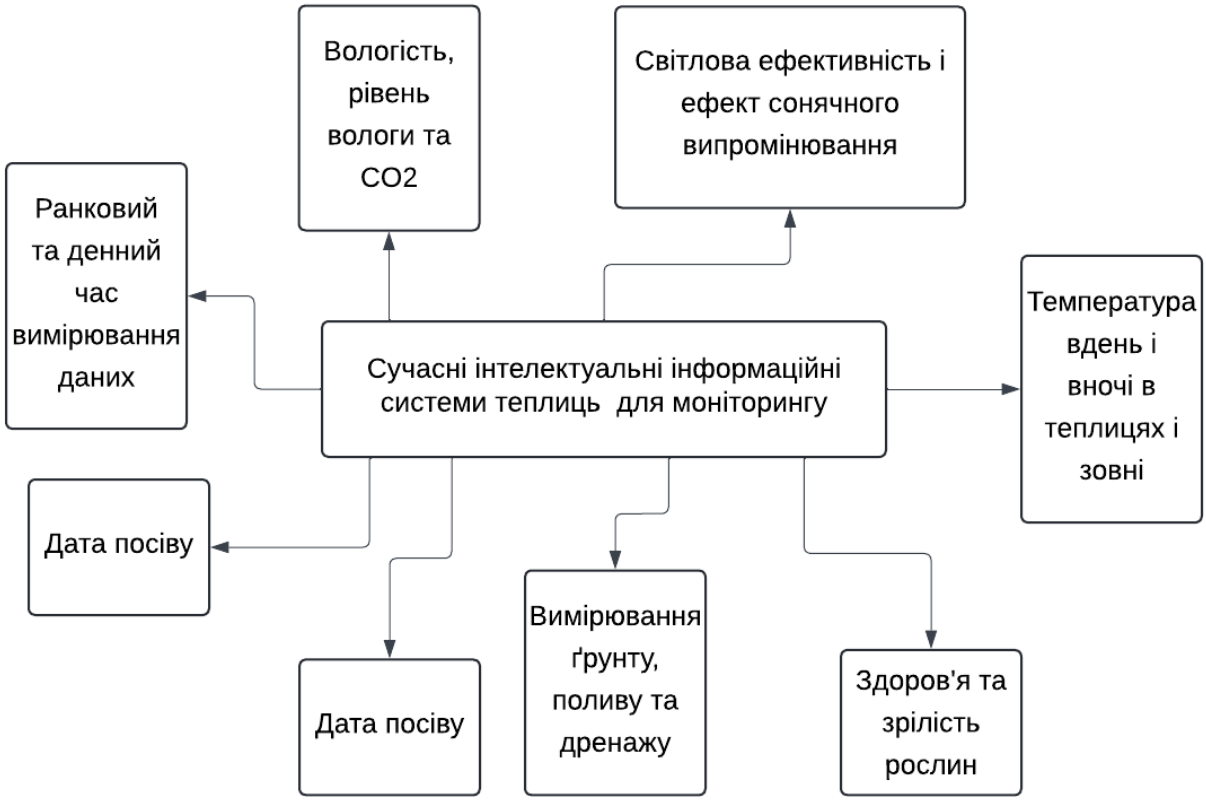

Рисунок 1.2 – Функціонал сучасних інтелектуальних інформаційних систем для моніторингу теплиць

Процес моніторингу та збору даних забезпечується комплексним набором інструментів і підключенням до Інтернету речей. Додатки збирають дані з

підключених датчиків і передають їх аналітикам, щоб вони могли застосовувати потужні інструменти аналітики та звітності для ефективного керування теплицями та надавати точні рекомендації щодо росту врожаю.

Рішення для автоматизованих теплиць включає календар вирощування, де всі виробники можуть отримати доступ до взаємно доступних даних для моніторингу інтелектуальних тепличних вимірювань і спостереження за тим, як певні умови впливають на ріст конкретної культури. Цей календар було перетворено на шість обов'язкових гістограм на основі критичних вимірювань, які кожен виробник може контролювати щотижня. Крім того, рішення містить конструктор для власних гістограм для моніторингу конкретних критеріїв зростання.

Основні характеристики автоматизованих тепличних систем представлено на рис. 1.3.

Рисунок 1.3 – Основні характеристики автоматизованих тепличних систем

1.2. Основні характеристики інтелектуальних систем для теплиць

Інтелектуальні системи для теплиць (ІСТ) – це новий вид автоматизованих тепличних систем (АТС), які використовують штучний інтелект (ШІ) для оптимізації параметрів середовища в теплицях. ІСТ збирають більше даних, ніж традиційні АТС, і використовують їх для більш точного та динамічного керування теплицею.

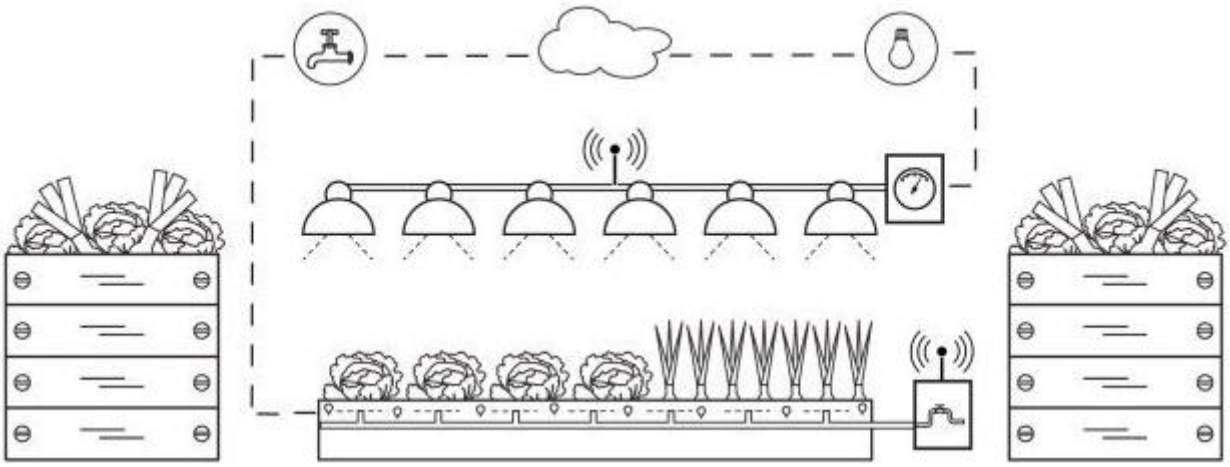

Рисунок 1.4 – Схема інтелектуальної системи для теплиць [8]

Сучасні ІСТ використовують машинне навчання для аналізу даних про рослини, середовище та інші фактори, що впливають на ріст і розвиток рослин. Це дозволяє ІСТ прогнозувати потреби рослин та автоматично регулювати параметри середовища відповідно до цих потреб.

Відомі ІСТ можуть оптимізувати не лише такі параметри, як температура, вологість та освітлення, але й інші фактори, такі як концентрація CO_2, pH ґрунту, полив та підживлення.

Теперішні ІСТ можуть враховувати особливості різних сортів рослин та стадій їх розвитку, що дозволяє забезпечити більш персоналізований та ефективний підхід до вирощування.

Відомі ІСТ можуть прогнозувати ризики виникнення проблем, таких як хвороби, шкідники або несприятливі погодні умови, та вживати заходів для їх

попередження. ІСТ можуть надавати фермерам інформацію та рекомендації щодо того, як покращити вирощування рослин та підвищити врожайність.

Основними перевагами використання ІСТ є:

- ✓ Підвищення врожайності та якості продукції. ІСТ можуть значно підвищити врожайність та якість продукції завдяки більш точному та динамічному керуванню теплицею.
- ✓ Економія ресурсів. ІСТ можуть економити воду, енергію, добрива та інші ресурси завдяки більш ефективному їх використанню.
- ✓ Зниження ризиків. ІСТ можуть допомогти запобігти загибелі рослин через несприятливі умови середовища, хвороби та шкідників.
- ✓ Полегшення роботи персоналу. ІСТ автоматизують багато рутинних завдань, що звільняє час персоналу для більш важливої роботи.
- ✓ Збільшення прибутку. ІСТ можуть допомогти фермерам збільшити свій прибуток за рахунок підвищення врожайності, економії ресурсів та зниження ризиків.

Також використання ІСТ має недоліки. Вони переважно є дорогими у впровадженні через складність програмного забезпечення та обладнання. Потребують кваліфікованого персоналу для їх обслуговування та оновлення. Потребують великої кількості даних для навчання та роботи.

Інтелектуальні системи для теплиць – це перспективний напрямок розвитку тепличного господарства, який має потенціал значно підвищити продуктивність та рентабельність тепличного виробництва.

1.3. Огляд існуючих інтелектуальних систем управління сільськогосподарськими теплицями

Відома система моніторингу теплиць на основі Інтернету речей використовує різні датчики для збору даних про умови навколишнього

середовища, ріст рослин та інші змінні [28]. Ці датчики можна встановити по всій теплиці, щоб контролювати її зони в реальному часі.

Датчики є основними пристроями в розумній теплиці, яка використовує IoT. Доступні різні типи, включаючи бездротові та дротові. Деякі вузькоспеціалізовані гаджети коштують дорого. Проте багато інших є доступнішими та контролюють декілька зон одночасно.

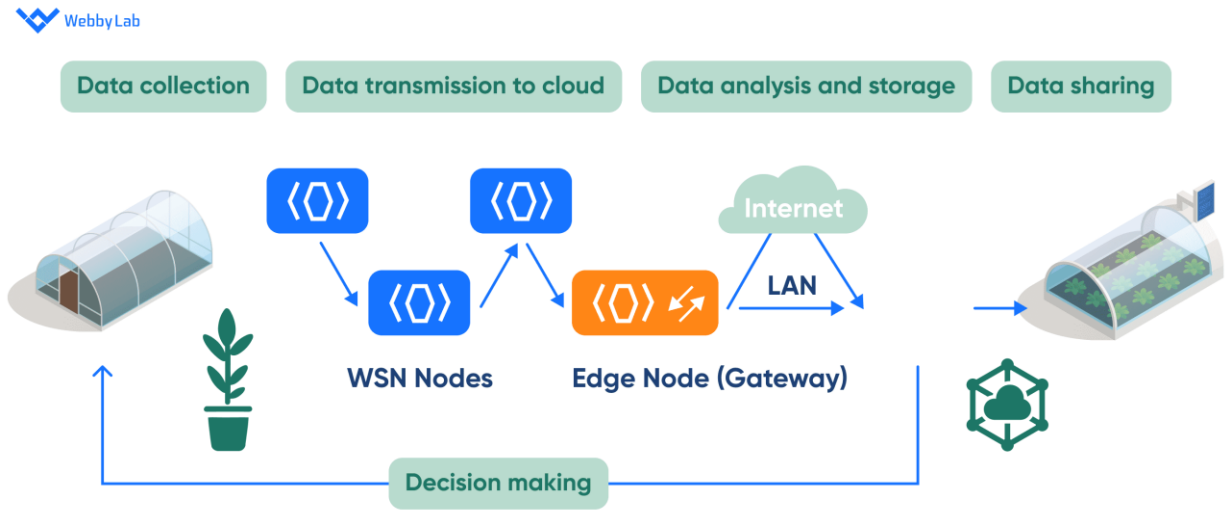

Рисунок 1.5 – Процес роботи розумної теплиці: збір даних, передача, аналіз, зберігання та обмін [28]

Базовий блок – це пристрій, який збирає дані з датчиків і надсилає їх у хмару або на локальний сервер. Він може підключатися до датчиків через Інтернет, радіопередавачі або дротове підключення.

Дані, зібрані датчиками, зазвичай надсилаються в хмару або локальний сервер для обробки, зберігання та аналізу. Оператори теплиць Інтернету речей можуть отримати доступ до цієї інформації, отримати з неї висновки та, зрештою, покращити екологічні умови.

Можна розмістити датчики IoT для моніторингу навколишнього середовища в різних зонах теплиці, включаючи ґрунт, стіни та обладнання. Дротові датчики зазвичай підключаються безпосередньо до базового блоку, а бездротові мають більшу мобільність.

Комерційні виробники створюють інтелектуальні системи автоматизації теплиць (рис. 1.6), щоб покращити якість і виробництво.

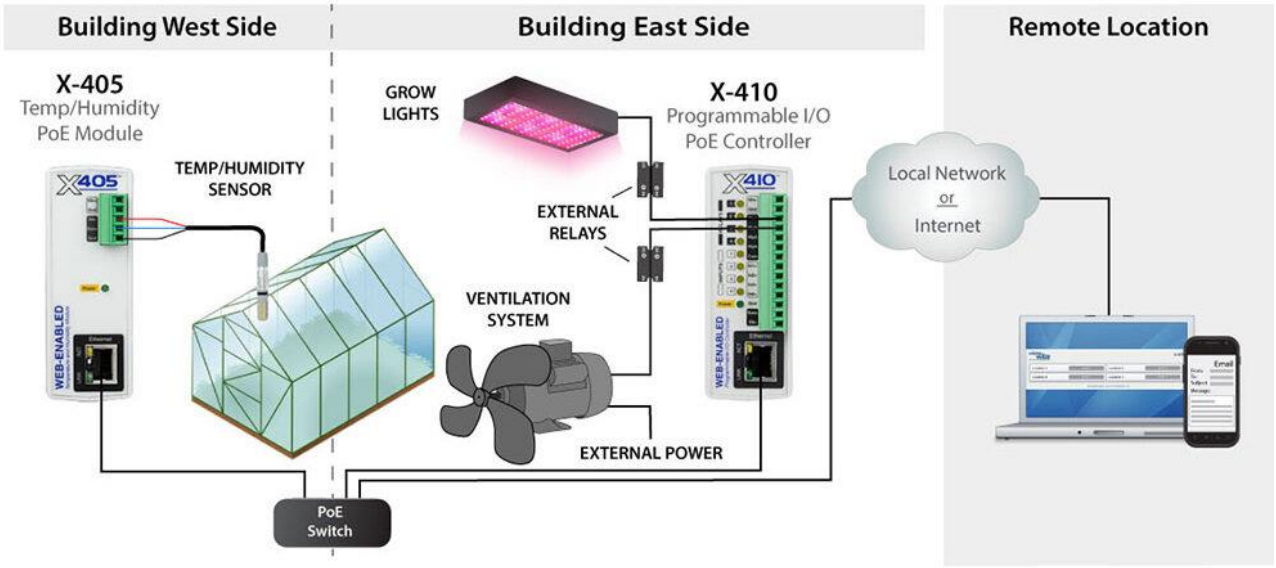

Рисунок 1.6 – Архітектура системи автоматизації теплиці [27]

Щоб ефективно автоматизувати теплицю, потрібен надійний моніторинг і контроль, який масштабується від невеликих теплиць до великих постійних споруд площею багато гектарів.

Особливістю системи автоматизації теплиці [27] є те, що вона забезпечує контроль температури/вологості і має датчики вологості ґрунту. Дає можливість контролювати вентилятори, вентиляційні отвори, обігрівачі, вологість ґрунту тощо. Передбачає локальний і віддалений моніторинг у реальному часі. Має журнал даних датчика. Забезпечує сповіщення електронною поштою/текстовими повідомленнями.

Також відомі розумні системи віддаленого моніторингу теплиць (рис. 1.7).

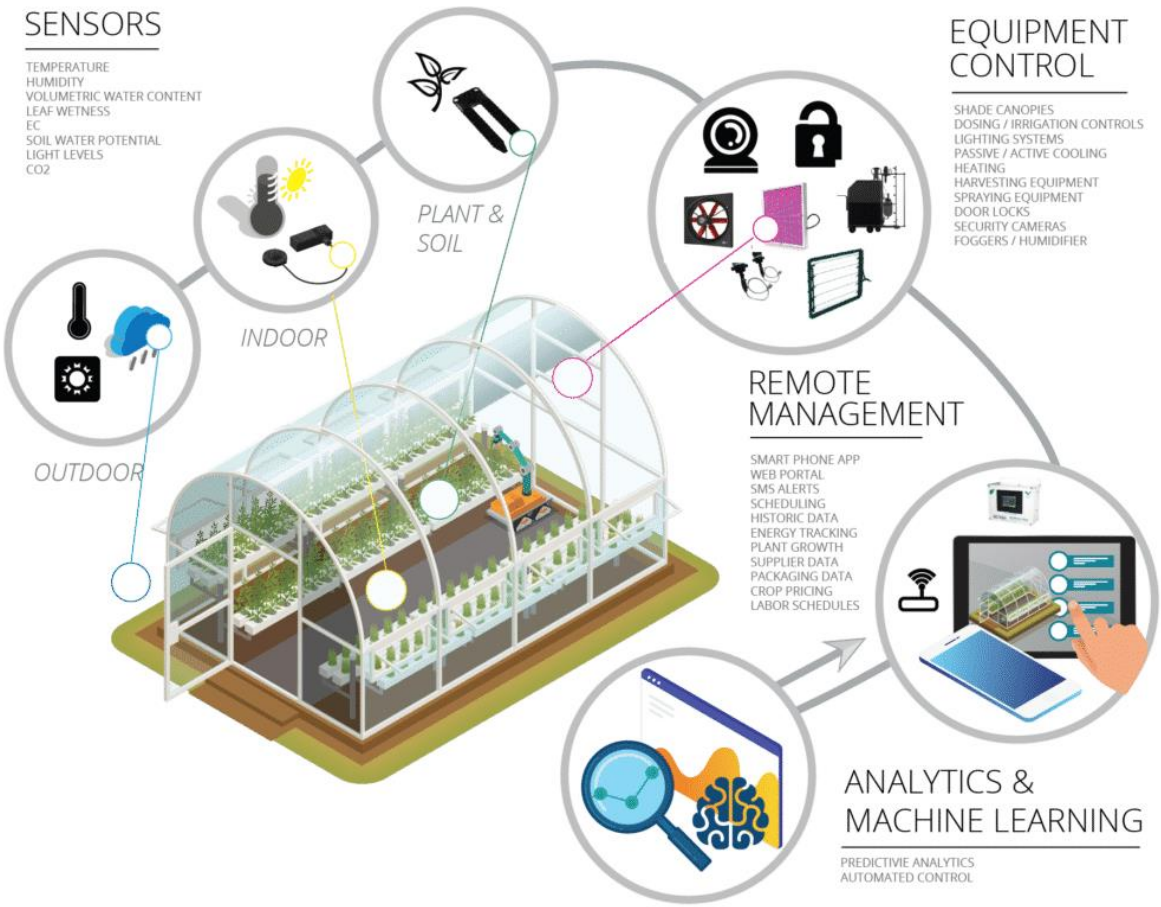

Рисунок 1.7 – Розумні системи віддаленого моніторингу теплиць

Важливо підтримувати контрольовану температуру в теплиці. Перепади температури можуть пошкодити або вбити ваші рослини всього за кілька годин. Системи дистанційного моніторингу захищають цінні рослини від екстремальних температурних коливань.

Для збереження здоров’я та росту рослин потрібні найкращі умови для вирощування. Але слідкувати за всіма змінами навколишнього середовища та станами або збоями обладнання може бути складним завданням.

Коли рослини знаходяться під загрозою, кожна секунда на рахунку. Чим раніше виявите падіння температури або несправність обладнання, тим більше рослин можна зберегти і підвищити продуктивність виробництва продукції. Системи віддаленого моніторингу забезпечують оновлення в реальному часі, тож можна швидко вжити заходів.

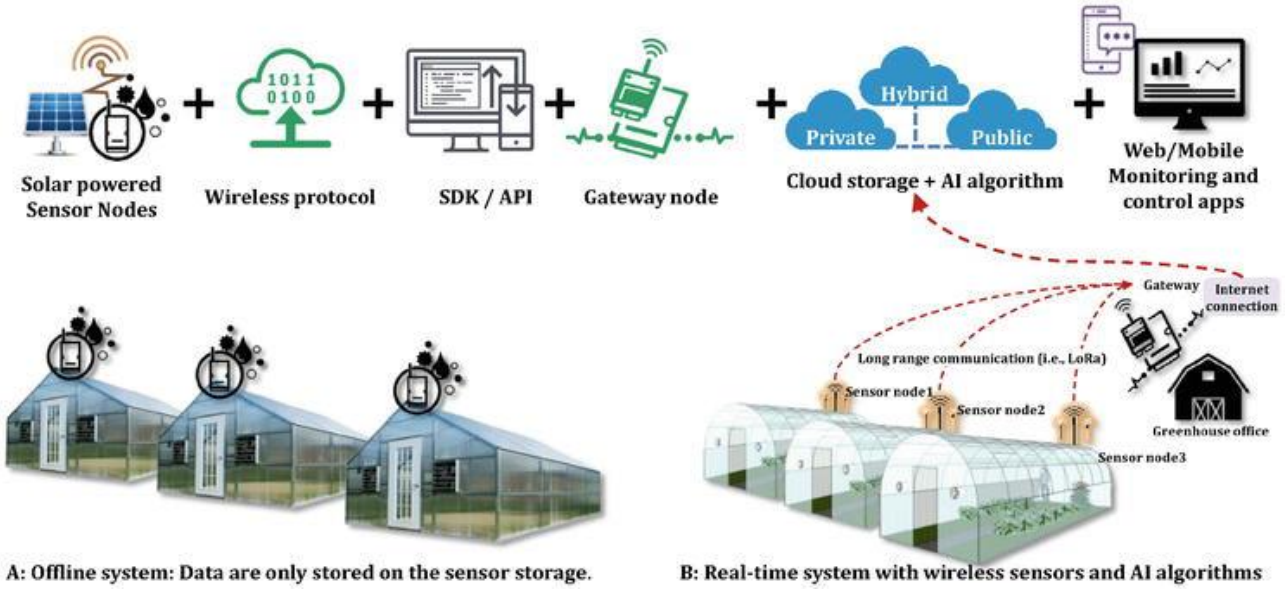

Рисунок 1.8 – Принципова схема бездротового зв'язку між вузлами датчиків теплиці та хмарним сховищем (Зображення Adaptive AgroTech)

На рисунку 1.8 показана загальна архітектура бездротового зв'язку для моніторингу та керування кількома теплицями через Інтернет речей. Основні аргументи для розгортання такої інфраструктури можна підсумувати як:

- ✓ забезпечення моніторингу змін і варіацій у режимі реального часу для забезпечення оптимального середовища зростання та мінімізації ризику несправності обладнання;
- ✓ обмін даними з хмарними системами. системи підтримки прийняття рішень;
- ✓ надсилати миттєві відповіді на бездротові приводи для зменшення вхідних витрат і підвищення врожайності та якості.

1.4. Аналіз розробок інтелектуальних систем управління із використанням генетичних алгоритмів

З розвитком комунікаційних технологій, мікроконтролерів, датчиків та Інтернету з'явився Інтернет речей, який застосовується в різних галузях промисловості, а також популяризуються і застосовуються «розумні» теплиці.

Компанія Maged NA представила імпульсний підвищувальний інвертор (SBI), придатний для сонячних фотоелектричних систем (PV) в якості джерела відновлюваної енергії [20].

У роботі [19] її автор запропонував метод оцінки мультиенергетичної комплементарної системи енергопостачання парку, заснований на вдосконаленому методі універсальної генеруючої функції (UGF).

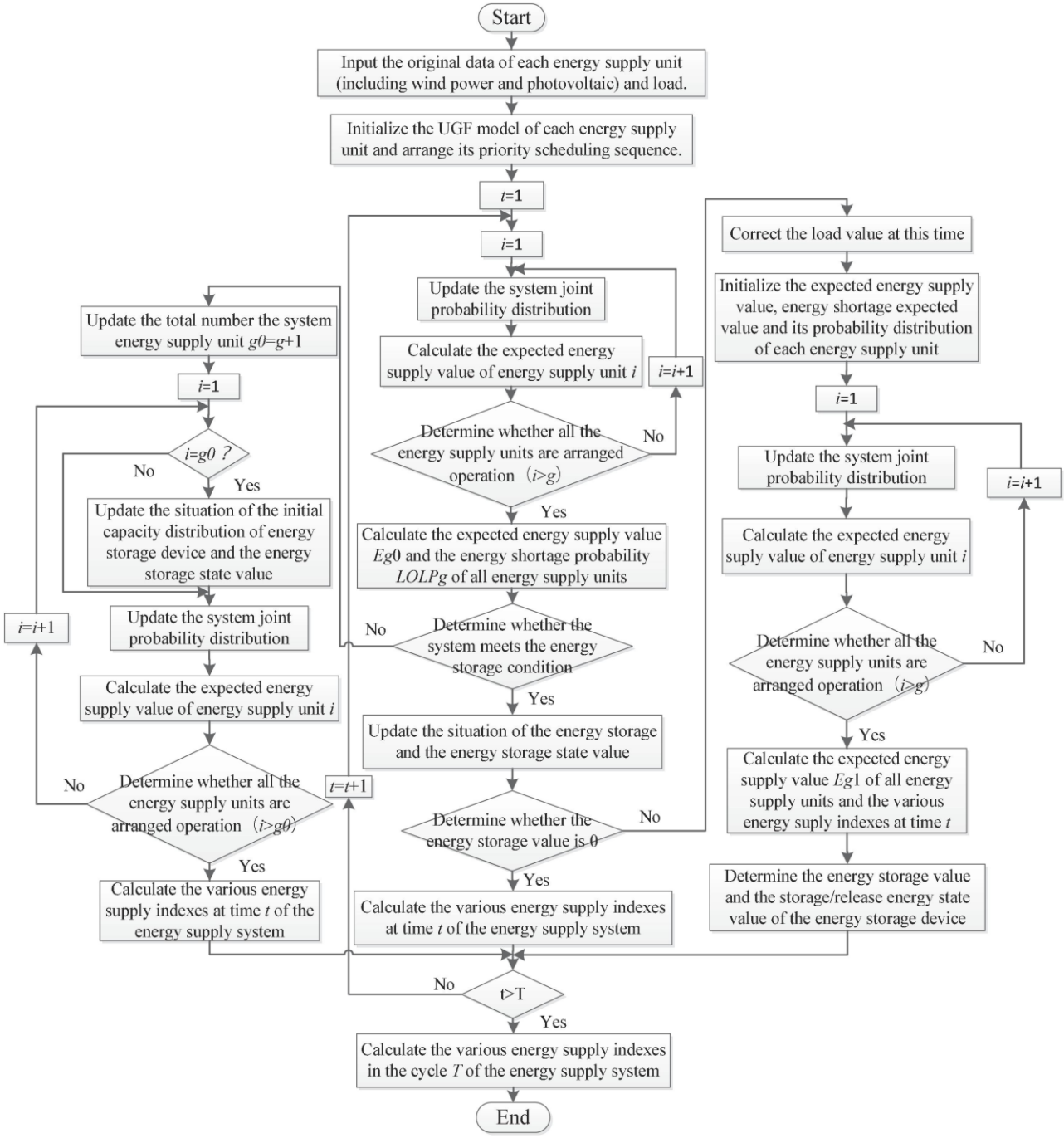

Рисунок 1.9 – Блок-схема оцінки системи енергопостачання в мультиенергетичному комплементарному парку на основі методу UGFPPS [19]

За допомогою ймовірності сталого стану блоку енергопостачання з чотирма станами можна отримати ймовірність еквівалентного блоку енергопостачання з трьома станами. Модель така:

$$P\,rate = P3\,, P\,derate = P1 + P4\,, P\,zero = P2\,, \qquad (1.1)$$

Вона представляє ймовірності стану номінальної потужності, стану зниженої потужності та стану нульової потужності еквівалентного блоку енергопостачання відповідно з трьома станами, (рис. 1.5).

Відома розподілена інтегрована мультиенергетична система (DIMS), яка зосереджена на глибокій інтеграції передових мультиенергетичних та інформаційних технологій, є найбільш вірогідною формою еластичного використання енергії в майбутньому людському суспільстві [15].

З розвитком соціальної економіки все більше науковців вивчають генетичні алгоритми. Для мульти мікромережевих систем з різними типами навантаження та потребами в електроенергії запропоновано економічну стратегію диспетчеризації для мульти-мікромереж на основі адаптивного мутаційного генетичного алгоритму [36].

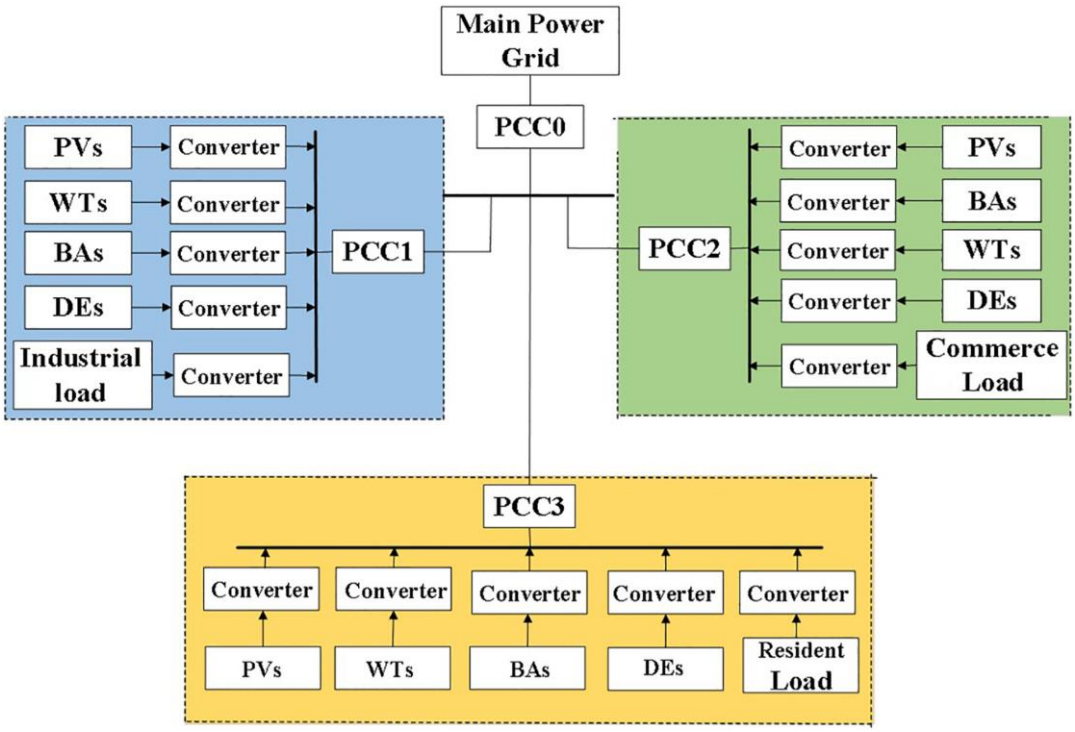

Рисунок 1.10 – Мультимікромережева система з різними типами навантаження та потребами в електроенергії [36]

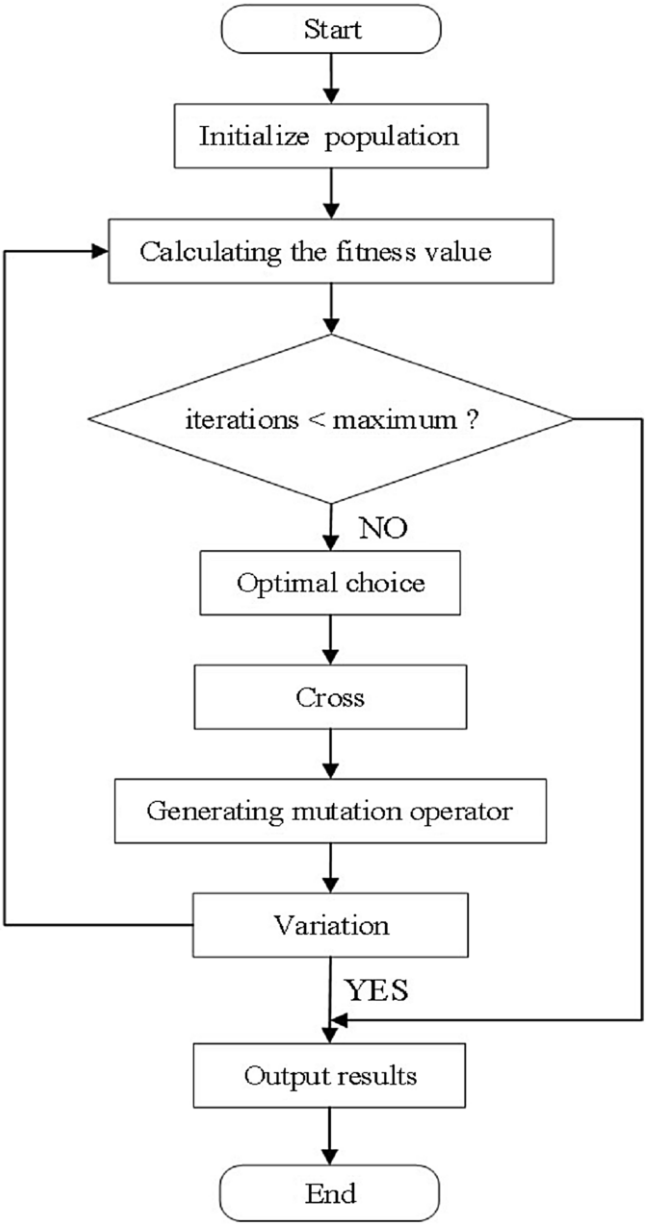

Рисунок 1.11 – Блок-схема генетичного алгоритму [36]

Етапи виконання алгоритму та його конкретна блок-схема показані на рис. 1.11.

1. У процесі оптимізації мікросітки ініціалізуються такі параметри, як розмір n популяції алгоритму, максимальне число ітерацій с пошуку алгоритму та ймовірність перетину P_c.

2. Розрахувати значення пристосованості окремої сукупності. Значення придатності, що відповідає кожному окремому, обчислюється та

оцінюється, чи досягає час ітерації заданого значення, якщо так, виводиться результат, інакше перетворюється на крок (3) для продовження виконання вниз.

3. Відповідно до розрахованої індивідуальної придатності вибираються особи з високим рівнем придатності.

4. Відповідно до заданого фактора кросинговеру та стратегії кросинговеру створюються нові особини.

5. Генерування фактора адаптивної мутації. Відповідно до рівняння, генерується фактор адаптивної мутації:

$$Q = A \cdot \left[\cos\left(\pi \cdot \frac{it-1}{T} \right) + 1 \right] + 0.2, \qquad (1.2)$$

де it – поточний номер ітерації алгоритму; T – максимальна кількість ітерацій, встановлена алгоритмом; A – контрольний параметр у діапазоні від 0 до 0,95, де вибрано A = 0,5.

6. Відповідно до факторів мутації, згенерованих на кроці (5), виконується операція мутації.

7. Нове покоління індивідуумів, утворене перехресною операцією та операцією мутації, повертається до кроку (2), щоб увійти до наступного циклу.

Енергетична взаємодія між мікромережами може бути досягнута через відповідні загальні точки з’єднання (PCC1, PCC2, PCC3), а вся багатомікромережа може взаємодіяти з основною мережею через загальні точки з’єднання на рівні групи PCC0. У кожній мікромережі фотоелектричні, вітряні, акумуляторні батареї та дизельні двигуни використовуються як джерела живлення для забезпечення енергією навантажень у відповідних діапазонах. Для того, щоб максимізувати ефективність використання відновлюваної енергії, енергетика вітру та фотоелектрична енергопостачання надаються пріоритетом у кожній зоні мікромережі.

З метою зменшення енергоспоживання в сушильній секції папероробної машини запропоновано метод оптимізації параметрів вентиляційної системи на основі вдосконаленого генетичного алгоритму.

Також вирішували проблему енергозберігаючої оптимізації технологічного маршруту обробного центру на основі вдосконаленого генетичного алгоритму, а також дослідив оптимізацію параметрів різання на основі вдосконаленого генетичного алгоритму [18].

З метою вирішення проблеми фрагментації спектру та високої надмірності ресурсів захисту в захисті з попередньо налаштованим періодом (P-цикл), запропоновано комбіновану стратегію захисту генетичного P-циклу для еластичної оптичної мережі на основі вдосконаленого генетичного алгоритму [16].

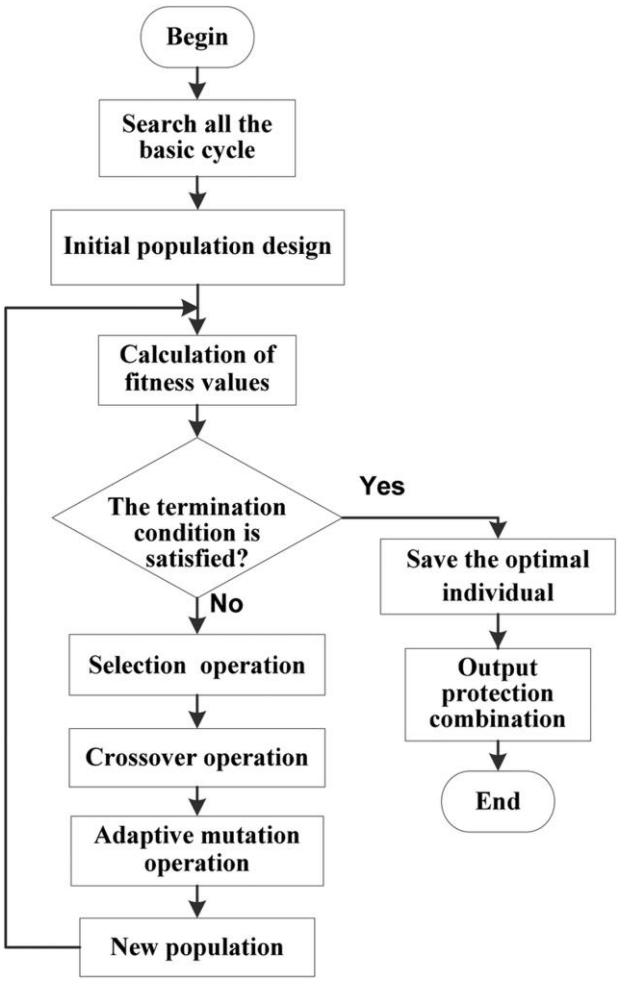

Рисунок 1.12 – Блок-схема алгоритму IGA [16]

На довжину шляху захисту зазвичай впливають стрибки P-циклів захисту. Довгий шлях захисту може призвести до відносно тривалого часу відновлення. Таким чином, поріг λ встановлюється для обмеження стрибків вибраних P-

циклів для запобігання тривалому відновленню. Це означає, що базові P-цикли видаляються з набору P, коли їхні стрибки перевищують поріг.

Кожна особина містить одну хромосому, і початкова хромосома побудована на основі методу десяткового кодування нефіксованої довжини. Зокрема, діаграма хромосоми та штучних хромосом наведена на рис. 1.13.

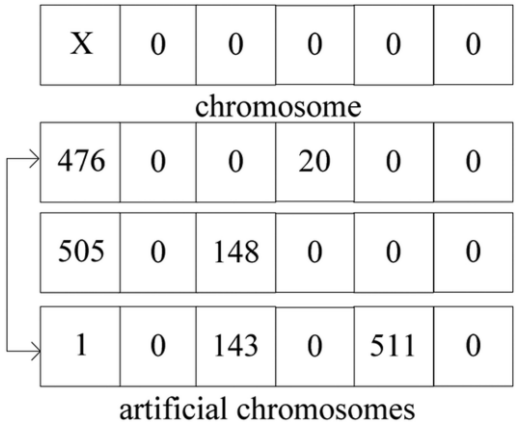

Рисунок 1.13 – Схема хромосоми та штучних хромосом [16]

Кожна хромосома представляє комбінаційну схему P-циклу і містить шість генів. Кожен ген є порядковим номером P-циклів. Крім того, для прискорення збіжності алгоритму з невисокою часовою складністю до вихідної популяції додають три штучні хромосоми. Штучна хромосома - це попередньо розрахована комбінована схема захисту P-циклу.

Недоліком відомих досліджень є те, що використання запропонованих алгоритмів не є достатньо всеосяжними для адаптації до більш складних ситуацій, а точність потребує покращення.

1.5. Ідентифікація ідеї проекту

У теперішніх умовах глобального зміни клімату та війни в Україні, аграрний сектор відчуває тиск на досягнення більшої продуктивності та ефективності. У цьому контексті, виникає необхідність розвинути і впровадити інноваційні технології, спрямовані на автоматизацію та оптимізацію процесів у

сільському господарстві. Однією з таких технологій є інтелектуальна система управління теплицями на основі генетичного алгоритму.

Ідея цього проекту виникла з потреби вдосконалення та автоматизації процесів керування тепличним господарством. Застосування інтелектуальних систем управління може значно підвищити продуктивність та якість вирощуваних культур, зменшити споживання ресурсів, а також покращити умови праці сільськогосподарських працівників.

Основною метою цього проекту є розробка та впровадження інтелектуальної системи, яка б забезпечувала автоматизоване керування кліматом у теплицях на основі генетичного алгоритму. Головні принципи такої системи полягатимуть у зборі даних з різноманітних датчиків у теплицях (температура, вологість, рівень CO2 тощо), їх аналізі та використанні для прийняття оптимальних рішень щодо керування умовами середовища для культурних рослин.

Такий підхід дозволить не лише автоматизувати процеси управління теплицями, а й забезпечить оптимальні умови для росту рослин у будь-який час, забезпечуючи максимальний врожай та економію ресурсів.

На базі розробленої інтелектуальної системи передбачається створення прототипу, який буде протестований на реальних об’єктах тепличного господарства з метою перевірки ефективності та доцільності її впровадження у виробничий процес.

РОЗДІЛ 2.
МАТЕМАТИЧНИЙ ОПИС ЗАДАЧІ ПРОЕКТУ ТА ВИБІР ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1. Особливості використання генетичних алгоритмів для вирішення практичних задач

Генетичний алгоритм – це алгоритм пошуку та оптимізації, розроблений на основі теорії біологічного природного відбору та генетичної еволюції [1].

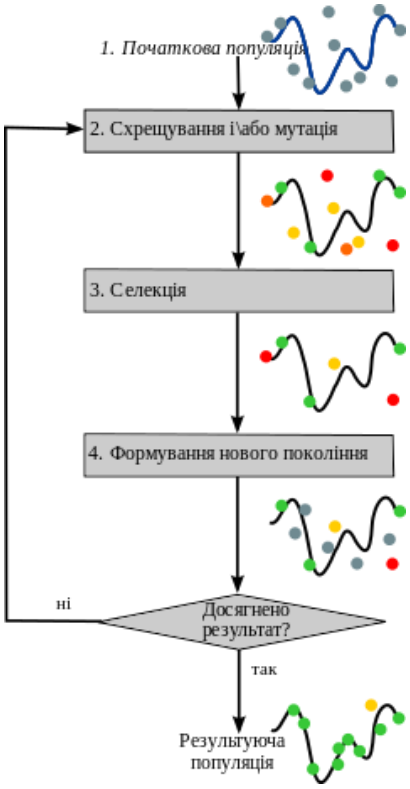

Рисунок 2.1 – Схема генетичного алгоритму [1]

Генетичні алгоритми використовуються для пошуку оптимальних або близьких до оптимальних рішень складних задач, особливо коли традиційні методи оптимізації можуть бути неефективними або непрактичними.

У природі види відбираються або знищуються природою через спадковість, мінливість, боротьбу за виживання та відповідно до їхньої пристосованості до навколишнього середовища. Генетичний алгоритм – це математичне представлення цього процесу. Основна ідея генетичного

алгоритму полягає в тому, щоб почати з початкової популяції, яка представляє рішення проблеми. Відповідно до пристосованості особин у популяції, проводяться операції відбору, кросинговеру та мутації, щоб сформувати нове покоління популяцій. Особини нової популяції не тільки успадковують гени попереднього покоління, але й перевершують попереднє покоління за загальними показниками.

Порівняно з іншими традиційними алгоритмами, генетичний алгоритм має переваги самоорганізації, самоадаптації та паралелізму. Він широко використовується в оптимізації функцій, комбінаторній оптимізації, плануванні виробництва, автоматичному управлінні та інших сферах.

Але він також має певні недоліки [13]. Базовий генетичний алгоритм використовує попередньо встановлені параметри керування, що має сильну залежність від параметрів. Розмір параметрів часто визначається досвідом і повторними експериментами. Якщо параметри підібрані неправильно, це вплине на продуктивність алгоритму. Чи зможе генетичний алгоритм отримати оптимальне рішення, залежить від правильного вибору розміру популяції.

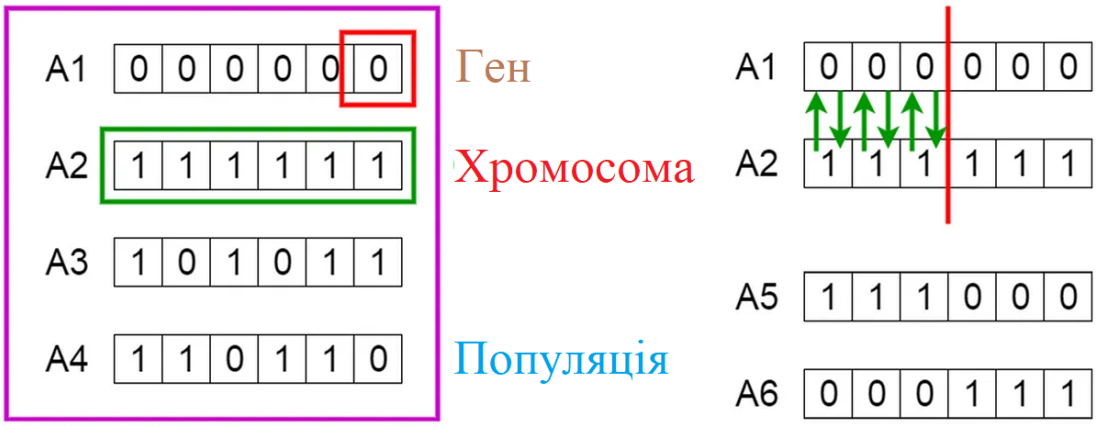

Рисунок 2.2 – Процес природного відбору найпристосованіших особин з метою отримання нащадків наступного покоління

Вибирається більш масштабна початкова популяція, що покращує різноманітність популяції і збільшує ймовірність отримання глобального оптимального рішення. Однак, якщо популяція занадто велика, час обчислень

алгоритму збільшується, а ефективність обчислень знижується. Якщо розмір популяції занадто малий, різноманітність популяції буде зменшена, і алгоритм буде збігатися занадто швидко, так що глобальний оптимальний розв’язок не може бути знайдений. Таким чином, розмір популяції має великий вплив на збіжність алгоритму. Базовий генетичний алгоритм фіксує розмір популяції, і основною причиною передчасного явища є те, що популяція втрачає свою різноманітність до того, як буде отримано глобальне оптимальне рішення, що змушує алгоритм потрапляти в локальне оптимальне рішення.

Збіжність генетичного алгоритму пов’язана не тільки з розміром популяції, але також залежить від генетичних операторів. Розмір ймовірності кросинговеру та ймовірності мутації безпосередньо впливає на швидкість генетичного алгоритму для генерації нових особин та збіжність алгоритму. Базовий генетичний алгоритм приймає незмінну ймовірність кросинговеру та ймовірність мутації, так що ці контрольні параметри не можуть саморегулюватися відповідно до процесу еволюції, що впливає на здатність алгоритму шукати оптимальне рішення [13].

Основні способи покращення адаптивного генетичного алгоритму можна узагальнити наступним чином:

- ✓ покращуються компоненти генетичного алгоритму або методи, що використовуються;
- ✓ використовується гібридний генетичний алгоритм;
- ✓ використовується динамічна адаптивна технологія для коригування параметрів управління в процесі еволюції;
- ✓ використовуються нестандартні генетичні оператори. По-п’яте, застосовано паралельний генетичний алгоритм.

2.2. Математичний опис генетичного алгоритму з адаптивним генетичним оператором

Ефективність генетичного алгоритму тісно пов'язана з підбором частоти кросинговеру H_c та частоти мутацій H_m. З цією метою запропоновано адаптивний генетичний алгоритм (АГА). Оптимальна швидкість кросинговеру H_c та швидкість мутації H_m відносно особини адаптивно призначається відповідно до рівня індивідуальної пристосованості.

Адаптивний генетичний алгоритм забезпечує збіжність генетичного алгоритму при збереженні різноманітності популяції. В адаптивному генетичному алгоритмі H_c і H_m адаптивно налаштовуються відповідно до наступних формул.

$$H_c = \left\{ \begin{array}{c} \dfrac{k_1\left(g_{\max} - g'\right)}{g_{\max} - g_{avg}}, g \geq g'_{avg} \\ k_2, g < g_{avg} \end{array} \right\}, \tag{2.1}$$

$$H_c = \left\{ \begin{array}{c} \dfrac{k_3\left(g_{\max} - g'\right)}{g_{\max} - g_{avg}}, g \geq g'_{avg} \\ k_4, g < g_{avg} \end{array} \right\}, \tag{2.2}$$

де $g_{\max}$ та g_{avg} – відповідно представляють максимальну пристосованість та середню пристосованість у популяції, відповідно; g' – представляє більшу пристосованість серед двох особин, що беруть участь у схрещуванні; g' позначає пристосованість особин, що беруть участь у мутації.

$$k_1, k_2, k_3, k_4 \in (0,1), \tag{2.3}$$

Таке коригування дозволяє особинам з нижчим за середній рівнем пристосованості отримати фіксовану більшу швидкість кросинговеру та мутацій. Особини з вищою за середню пристосованістю отримують відповідну швидкість кросинговеру і мутацій відповідно до їхньої пристосованості [22]. За такої стратегії на ранній стадії еволюції у пристосованих особин практично немає шансів брати участь в операціях кросинговеру і мутації. У цей час ці

особини з високою пристосованістю не обов'язково є глобальним оптимальним рішенням, що збільшує ймовірність отримання алгоритмом локального оптимального рішення. Таким чином, можна покращити швидкість кросинговеру H_c і швидкість мутацій H_m:

$$H_c = \left\{ \begin{array}{c} H_{c1} - \dfrac{(H_{c1} - H_{c2})(g_{\max} - g')}{g_{\max} - g_{avg}}, g' \geq g_{avg} \\ H_{c1}, g' < g_{avg} \end{array} \right\}, \quad (2.4)$$

$$H_m = \left\{ \begin{array}{c} H_{m1} - \dfrac{(H_{m1} - H_{m2})(g_{\max} - g)}{g_{\max} - g_{avg}}, g \geq g_{avg} \\ H_{m1}, g < g_{avg} \end{array} \right\}, \quad (2.5)$$

Це покращення збільшує частоту кросинговеру та мутацій особин з високим рівнем пристосованості в популяції, завдяки чому вони більше не залишаються майже в стані стагнації.

2.3. Використання у теплицях додаткової системи сонячного та вітрового електропостачання

З цієї причини в цій конструкції сонячна енергія та енергія вітру використовуються для забезпечення електроенергією електричних обігрівачів, освітлення та водяних насосів, а частина енергії зберігається для використання під час погіршених погодних умов. Крім того, система електропостачання також може задовольнити основну потребу у споживанні електроенергії керівним персоналом.

Пропонується здійснювати управління теплицею із використанням поновлюваних джерел енергії. Вона включає систему використання енергії вітром, систему додаткового освітлення, фотоелектричну систему електропостачання, систему вентиляції, систему підживлення рослин, систему

циркуляційної води та систему збору даних та контролю. На рис. 2.2 представлена схематична діаграма моделі теплиці.

Теплиця використовує сонячну фотоелектричну енергію та вітрову енергію для освітлення сільськогосподарських культур і регулювання температури в теплиці протягом дня. Коли температура низька, необхідна температура поступово досягається за рахунок електричного нагрівача та циркуляційного водопостачання. Якщо джерело живлення не відповідає температурним вимогам або не працює нормально, температуру підвищують шляхом спалювання біогазу. При високій температурі в теплиці включається вентилятор. Якщо температура все одно не знижується, вмикають вентилятори за межами теплиці.

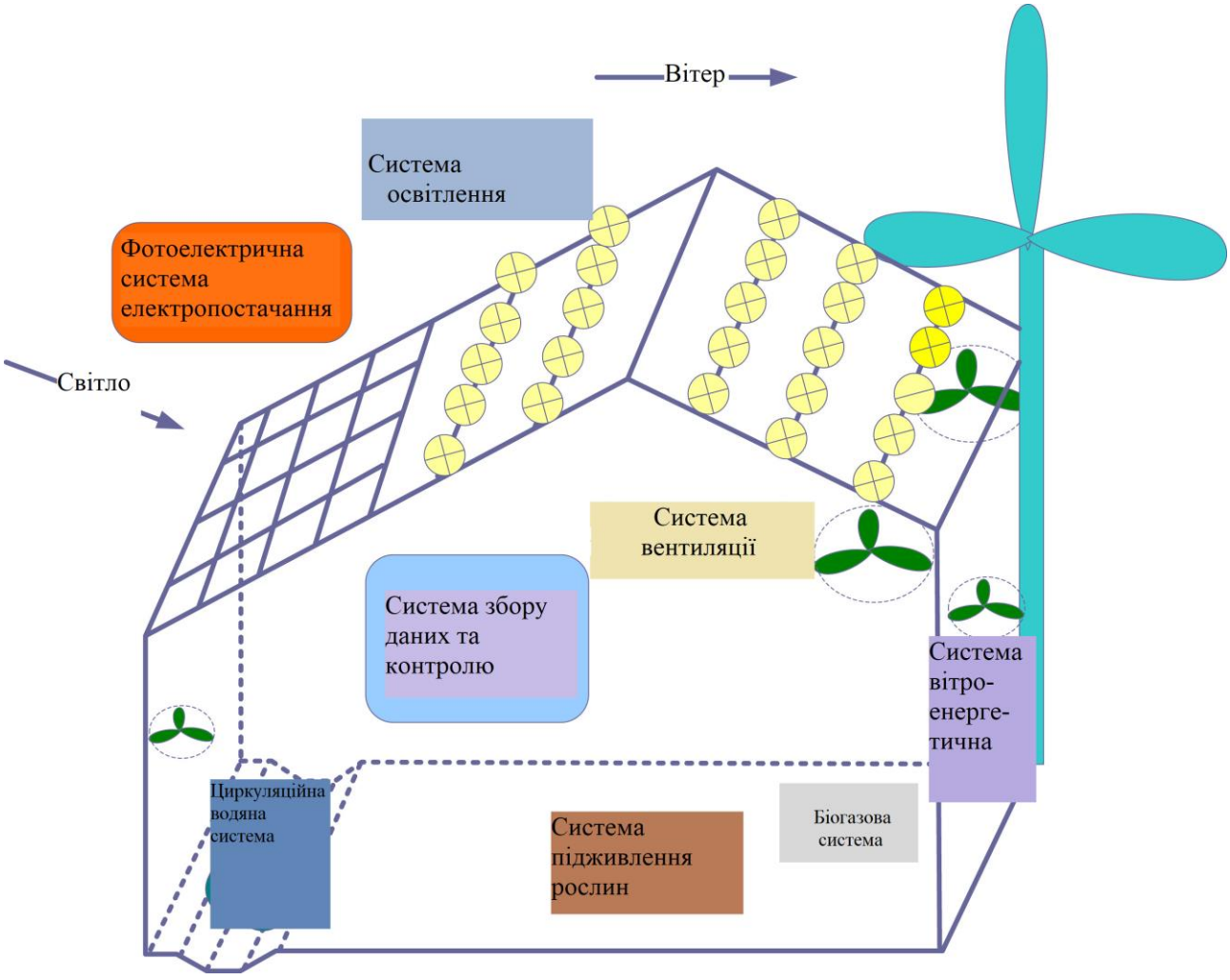

Рисунок 2.2 – Схема теплиці із об’єктами, які потребують управління

Якщо вищезазначені функції виконуються, але все ще є надлишок електроенергії, частина її буде зберігатися в акумуляторній батареї як резервна електроенергія для задоволення потреб теплиці в електроенергії, а інша частина використовуватиметься для накопичення енергії в циркулюючій воді.

2.4. Використання нечіткого підходу до управління теплицями

Нечіткий підхід добре адаптований для управління складними системами і менш чутливий до змін параметрів. Такий контролер не може запобігти побічним ефектам. Можна розглянути два рішення: або застосування модального керування, або впровадження нечіткого регулятора на основі простого PID (SPID) [14]. Контроль SPID дозволяє керувати стабільністю високих частот шляхом введення компенсаторів фазової затримки (рис. 2.3).

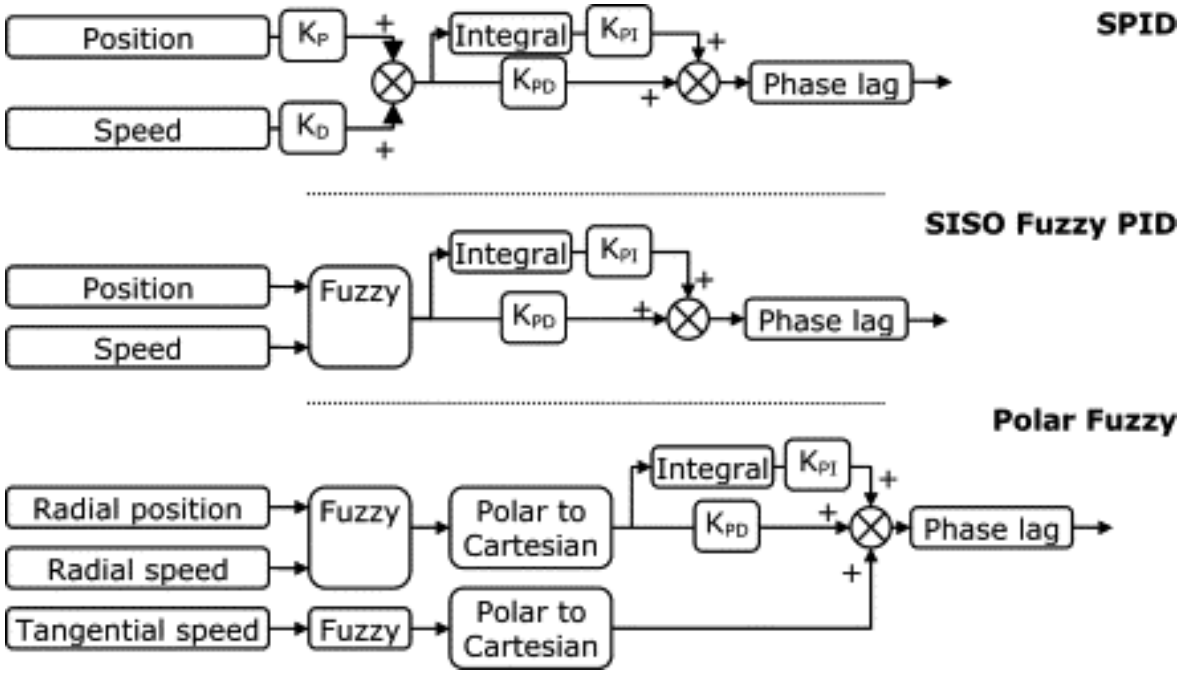

Рисунок 2.3 – Схема контролерів [14]

Основні принципи нечіткого підходу полягають у тому, що вхідні дані спочатку розбиваються на лінгвістичні величини, наприклад, високі та низькі. Кожному вхідному стану відповідає математична функція належності. По-

друге, механізм виведення реалізує набір лінгвістичних правил, заснованих на поведінці системи. Ці правила є умовними і повинні описувати всі можливі події, які можуть відбутися. Така оцінка вимагає ґрунтовних знань про систему. Кілька простих правил можуть бути достатніми для опису поведінки системи, оскільки обмежена кількість правил спрощує процес проектування і може призвести до оптимізації продуктивності. Нарешті, управління отримують після дефазифікації. Це полягає в агрегуванні висновків кожного правила.

Запропоновано два контролери на основі нечіткої логіки (рис. 2.3). Першим був SISO-регулятор, для якого вхідним сигналом є переміщення, виміряне вздовж лінії дії, а вихідним – сила, прикладена в тому ж напрямку. Він має безперервну нелінійну поведінку. Тому був запропонований другий контролер, який розглядав кожен вузол як єдину MIMO-систему. Вхідними даними є радіальне положення і швидкість, а також тангенціальна швидкість, розрахована на основі переміщень, виміряних у двох напрямках. На виході – сили, обчислені в полярних координатах, а потім перетворені в командні струми.

Першим кроком нечіткого підходу було визначення простих коефіцієнтів ПІД-регулятора, добровільно прості характеристики якого знаходяться в низькочастотному діапазоні. Зазвичай, характеристики визначаються як функція динамічної поведінки та кількості режимів, включених в умови експлуатації.

Для того, щоб узагальнити процес проектування, коефіцієнти були обрані таким чином, щоб демпфування було майже постійним і з низьким постійним нахилом для жорсткості для розглянутого робочого діапазону. Таким чином, регулятор став менш ефективним, але більш багатофункціональним.

Використаний нечіткий ПІД SISO має загальну структуру, описану в [25]. Структура була адаптована для керування гнучким ротором. Це була сума нечіткого ПІ з нечітким PD-регулятором. Переміщення вимірювалося, а швидкість обчислювалася шляхом чисельного виведення. Ці два дані

(переміщення і швидкість) використовувалися як вхідні дані для нечіткого контролера.

На підставі вище сказаного нами сформульовано правила багатопараметричного нечіткого регулятора (табл. 2.1).

Таблиця 2.1 – Імпакт-фактора обладнання теплиці

Екологічні параметри		Температура повітря	Відносна вологість	Інтенсивність світла
Контрольне обладнання	Сонцезахисна штора	0,5	0,2	0,8
	обігрівач	0,8	0,4	0,1
	Мансардне вікно	0,2	0,3	0,1
	Бокове вікно	0,5	0,3	0,1
	Мокра фіранка	0,7	0,8	0,1
	Заповнює світло	0,1	0,1	0,5
	Вентилятор вентиляції	0,7	0,7	0,1
	Циркуляційний вентилятор	0,2	0,2	0,1

Наприклад, при контролі температури повітря доступне обладнання для охолодження включає сонцезахисні жалюзі, мансардні вікна, бічні вікна, вологі штори та вентилятори. Згідно з таблицею 2.1, фактори впливу сонцезахисної завіси, мокрої завіси та вентиляційного вентилятора є відносно великими в деяких обладнаннях. Тому, коли різниця температур велика, комбінація цих пристроїв контролю є кращою.

2.5. Сегментований контроль температури

Управління змінною температурою тепличних культур має велике значення для покращення економічної вигоди, якості та стану

сільськогосподарських культур. Регулювання змінної температури полягає в тому, щоб розділити день на чотири періоди часу для контролю температури.

Рослини здійснюють довготривале дихання вночі, при цьому органічні речовини в організмі майже витрачені, а в повітря виділяється велика кількість вуглекислого газу. Коли інтенсивність світла починає зростати, рослини потребують і в основному здійснюють фотосинтез для накопичення великої кількості органічної речовини. У цей час температура в приміщенні стає умовою обмеження інтенсивності світла, тому вранці потрібна найвища температура.

Таблиця 2.2 – Оптимальна температура для контролю змінної температури на стадії росту кількох поширених тепличних культур

Види рослин	Відповідна температура			
	Ранок	Післяобідній час	Перша половина вечора	Друга половина нічна
помідор	26	19	10	4
огірок	31	26	15	5
перець	31	25	22	15
Баклажани	31	25	22	15

У другій половині дня, коли органічні речовини в рослині повільно накопичуються, фотосинтез поступово слабшає. Крім того, інтенсивність дихання не знижується, і воно знаходиться в стадії накопичення органічних речовин.

Перша половина ночі – це процес від заходу до темряви. Накопичена за день органічна речовина шляхом дихання поступово транспортується до частин, де інтенсивно ростуть квіткові бруньки та плоди. У цей час необхідна відповідна температура для сприяння процесу транспортування. Рослини

переважно дихають у другій половині ночі. Оскільки зі зниженням температури інтенсивність дихання може слабшати, у цей час необхідно лише підтримувати безпечну температуру для досягнення мети сповільнення дихання та зменшення споживання. За законом росту рослин день поділяється на:

1. ранок (6.00-12:00),
2. післяобідній час (12:00-17:00),
3. перша половина вечора (17:00-21:00),
4. друга половина нічна (21:00-6:00 наступного дня).

Щоб сприяти фотосинтезу, сповільнюючи дихання рослин, а також певною мірою запобігти шкідникам і хворобам, цільові температури повинні змінюватися від високої до низької протягом чотирьох періодів дня (табл. 2.2).

2.6. Вибір засобів для виконання проекту

Для розробки інтелектуальної системи управління теплицею з мультиенергетичним енергопостачанням та іншими описаними функціями, необхідно вибрати мови програмування та бібліотеки, які забезпечать ефективну роботу всіх модулів і компонентів системи. Нами здійснено вибір мов програмування, бібліотек та платформ для кожного з модулів.

На мові програмування C/C++ будуть написані:

1. Модуль збору даних – для роботи з мікроконтролерами та низькорівневим обладнанням, включаючи Zigbee модулі.
2. Модуль зв'язку – для реалізації інтерфейсу RS232.
3. Модуль контролю – для реалізації PID-алгоритмів і генетичного алгоритму.

На мові програмування Python будуть написані:

4. Модуль обробки даних – для обробки даних, інтеграції алгоритмів і роботи з базою даних SQLite.

5. Модуль взаємодії з користувачем – для розробки графічного інтерфейсу (GUI) та обробки даних.

На мові програмування JavaScript (Node.js) будуть написані:

6. Модуль веб-інтерфейсу – для створення веб-інтерфейсу, якщо потрібен доступ через веб-браузер.

Візуалізація додатку у середовищі Qt (C++):

7. Модуль взаємодії з користувачем – розробки графічного інтерфейсу на сенсорному екрані.

Також нами використано бібліотеки та платформи. Модуль збору даних:

1. Zigbee – бібліотеки для роботи з Zigbee (наприклад, zigpy для Python).
2. C/C++ бібліотеки для мікроконтролерів – Arduino IDE, mbed, STM32 HAL.

Модуль обробки даних:

3. NumPy, Pandas (Python) – для обробки і аналізу даних.
4. SQLite – для управління базою даних.

Модуль контролю та управління:

5. SciPy (Python) – для реалізації PID-контролю.
6. DEAP (Python) – для реалізації генетичних алгоритмів.

Модуль взаємодії з користувачем:

7. PyQt/PySide (Python) – для розробки графічного інтерфейсу.
8. Qt (C++) – для розробки GUI на сенсорному екрані.
9. Matplotlib, Plotly (Python) – для візуалізації даних.

Модуль зв'язку:

10. pySerial (Python) – для роботи з інтерфейсом RS232.
11. Boost.Asio (C++) – для реалізації асинхронного вводу-виводу.

Модуль бази даних:

12. SQLite – вбудована база даних для зберігання та запиту даних.
13. SQLAlchemy (Python) – ORM для роботи з SQLite.

Документація і довідка:

14. Sphinx (Python) – для створення документації.

Модуль оновлення програмного забезпечення:

15. OTA (Over-the-Air) Update Libraries – наприклад, Mender або Balena для оновлення прошивок.

Пропонується використання вбудованої операційної системи Linux на ARM-платі для забезпечення стабільності і можливості розширення. Використання Docker для контейнеризації окремих модулів, що полегшить розгортання та обслуговування системи. Використання Git для контролю версій та автоматизованого тестування для забезпечення якості коду.

Вибір цих мов програмування, бібліотек та платформ дозволить забезпечити ефективну роботу всіх компонентів системи управління теплицею, а також спростить інтеграцію та подальший розвиток системи.

РОЗДІЛ 3.
ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ ТЕПЛИЦЯМИ ІЗ ВИКОРИСТАННЯМ ПОНОВЛЮВАНИХ ДЖЕРЕЛ ЕНЕРГІЇ

3.1. Розробка принципової схеми системи керування теплицями

Під час війни в Україні спостерігаються часті відключення електроенергії. Це зумовлює змінювати системи енергозабезпечення теплиць завдяки використанню поновлюваних джерел енергії. Для цього слід під кліматичні умови регіону виконувати трансформацію енергопостачання теплиць, а також інтегровано виконувати збір та аналіз даних про навколишнє середовище, енергопостачання, системи світлодіодного освітлення та системи управління.

Відповідно із врахуванням кліматичних та ресурсних умов, пропонується використовувати мультиенергетичну систему енергопостачання для сільськогосподарських теплиць. Сонячна фотоелектрична система буде генерувати електроенергію, вітрова система також буде генерувати електроенергію, система зберігання енергії буде здійснюватися завдяки системи циркулюючої води, а також біогазова система використовується для енергопостачання теплиць. Збір та контроль параметрів навколишнього середовища теплиць (включаючи температуру, вологість, світло тощо) здійснюється за допомогою бездротової сенсорної мережі, а параметри навколишнього середовища та умови росту рослин у теплиці можна переглядати в режимі реального часу на головному комп’ютері.

Дані передаються в ARM для обробки, і центр обробки ARM використовує нечіткий алгоритм самонастроювання PID для визначення вихідної команди. Інструкції передаються на виконавче обладнання для реалізації контролю інформації теплиці.

Окрім того, система забезпечує обробку та зберігання даних у вбудованій операційній системі Linux. Передбачається використання SQLite (полегшена реляційна система керування базами даних). Окрім того, розробляється інтерактивний інтерфейс та РК-дисплея в режимі реального часу. Система управління теплицею показана на рис. 3.1.

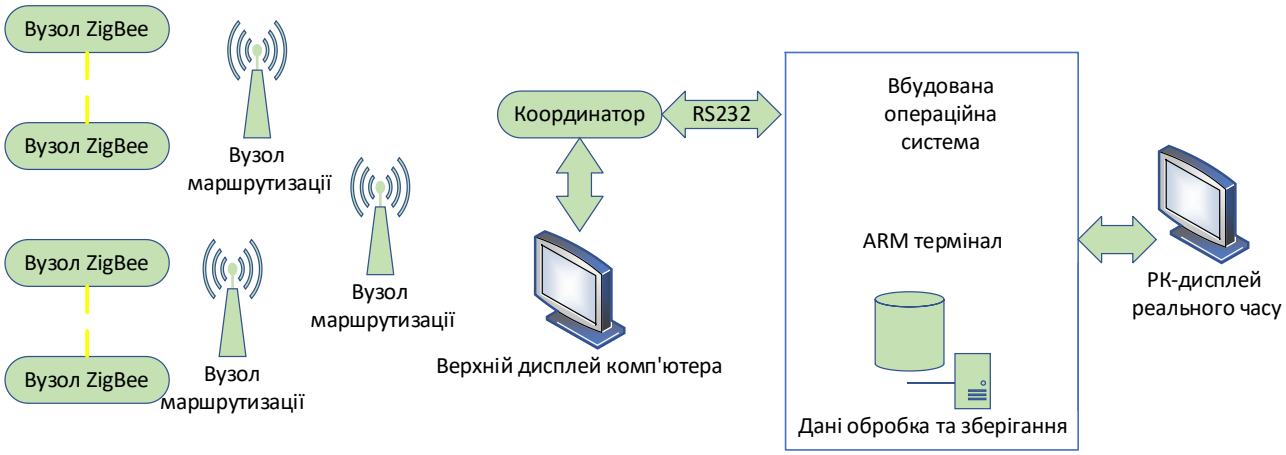

Рисунок 3.1 – Принципова схема системи керування теплицями

Рівень бездротового сенсорного модуля

Дані збираються датчиками і надсилаються, їх можна переглянути в програмному забезпеченні хост-комп’ютера. Це комп’ютер або інший пристрій, який спілкується з іншими хостами в мережі. Також відомі як мережеві хости, хости включають клієнтів і сервери, які надсилають або отримують дані, служби та програми.

Хости зазвичай не містять проміжних мережевих пристроїв, таких як комутатори та маршрутизатори , які замість цього часто класифікуються як вузли. Вузол – це ширший термін, який включає все, що підключено до мережі, тоді як для хоста потрібна IP-адреса. Іншими словами, усі хости є вузлами, але мережеві вузли не є хостами, якщо їм для функціонування не потрібна IP-адреси.

Хост-комп’ютер також взаємодіє з ARM через RS232 для отримання даних та інструкцій, а також виконує інструкції управління, видані терміналом ARM. Послідовний зв’язок, або протокол RS-232, – це метод передачі даних по

одному біту за один раз у послідовному режимі. Кабель RS-232 дозволяє здійснювати цей тип зв’язку, передаючи дані від одного пристрою до іншого. Протокол включає просте налаштування лінії передачі (лінія TX), лінії прийому (лінія RX) і лінії заземлення (лінія GND), один біт даних за раз надсилається через лінію TX і приймається через лінію RX.

Рівень модуля ARM

Підключена система збору даних Zigbee, а для реалізації обробки даних використовується нечіткий алгоритм самонастроювання PID.

Zigbee – це бездротова технологія, розроблена як відкритий стандарт зв’язку на світовому ринку для задоволення унікальних потреб недорогих, малопотужних бездротових мереж передачі даних IoT. Вона призначена для підтримки бездротового зв’язку, моніторингу та управління пристроями, що працюють на батареях, і сенсорними мережами. Zigbee працює за специфікацією IEEE 802.15.4 і широко використовується в системах домашньої автоматизації, збору медичних даних та промислових системах управління.

Після прийняття рішення подається команда управління, і зовнішній сенсорний РК-дисплей може відображати, запитувати та керувати в режимі реального часу. Система використовує вузол Zigbee для підключення необхідних датчиків: DHT11, MQ-2, фоторезистор тощо. Координатор CC2530 підключений до ARM-плати через RS232. Плата ARM використовує операційну систему Linux. Інтерфейс Qt/E розроблений на сенсорному екрані, а база даних SQlite використовується для перегляду і запитів даних про навколишнє середовище у теплиці.

3.2. Архітектура системи керування теплицями

Архітектура системи керування теплицями передбачає використання підсистеми мультиенергетичного енергопостачання та підсистеми управління теплицями. Вона складається з наступних трьох частин: 1)

мультиенергетичного енергопостачання; 2) бездротової сенсорної мережі; 3) платформи керування та програмного забезпечення.

Запропоновано використовувати мультиенергетичну систему енергопостачання для теплиць. Відповідно до регіональних умов, сонячні та вітрові ресурси раціонально використовуються для створення фотоелектричної та вітрової генерації електроенергії, накопичення енергії у циркулюючій воді та біогазової системи, що забезпечує енергопостачання теплиць. За відсутності електромережі вона може задовольнити основні потреби теплиці в електроенергії. Світло, необхідне культурам на різних стадіях росту, забезпечує система, що створює червоне та синє світлодіодне підсвічування.

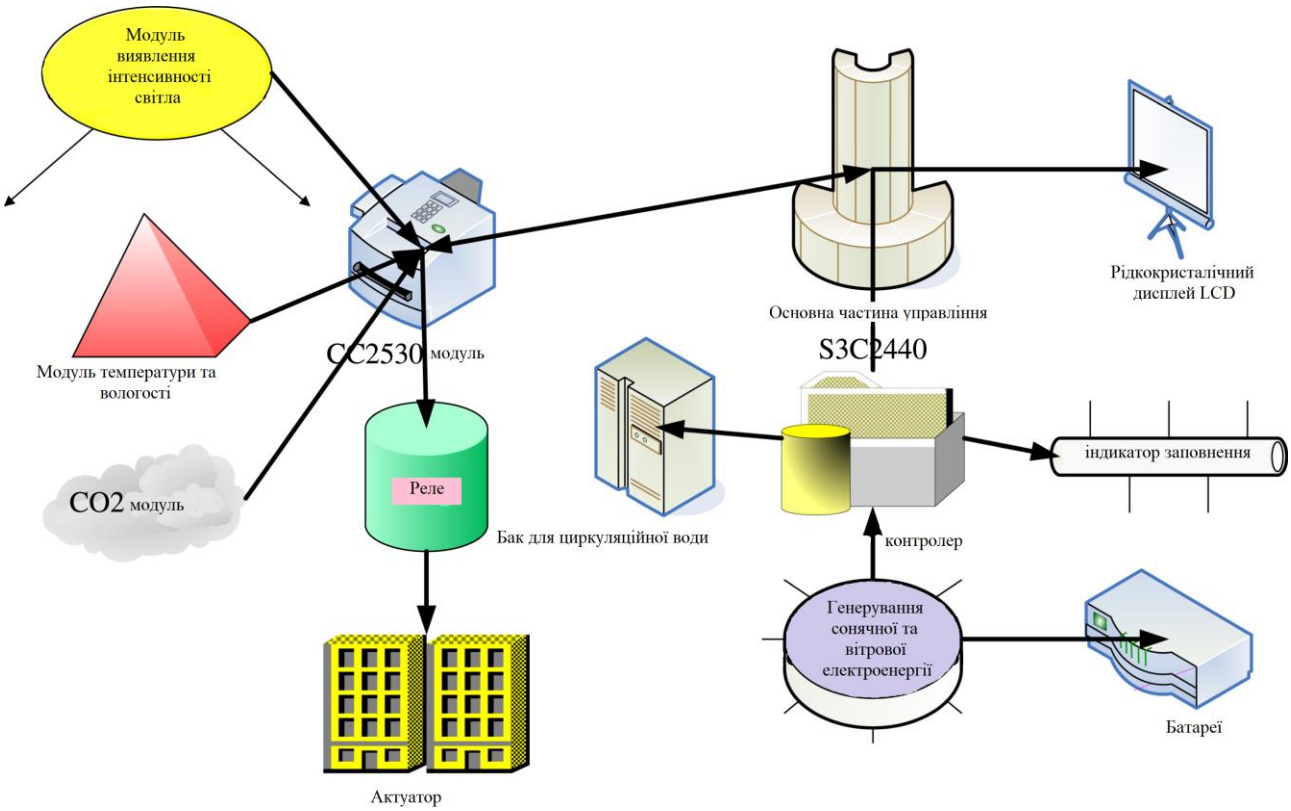

Рисунок 3.2 – Архітектура системи керування теплицями

Бездротова сенсорна мережа використовується для збору та контролю даних у середовищі теплиць. Зібрані дані надсилаються на термінал ARM. Термінал ARM обробляє дані про навколишнє середовище теплиці за допомогою нечіткого алгоритму самонастроювання PID і видає команди

управління після прийняття рішення для здійснення контролю параметрів мікроклімату теплиці.

Система передбачає вбудовану платформу та розробку пов’язаних з нею програм. Вбудовані складові включають: крос-компілятор, Qt, адаптацію ядра та дизайну бази даних SQlite тощо. РК-дисплей ARM-терміналу відображає інформацію про мікроклімат теплиці в режимі реального часу, створює графіки відображення інформації про стан мікроклімату у теплиці та зберігає дані про навколишнє середовище. Зручний інтерфейс Qt використовується для реалізації ручного керування теплицею.

Нами пропонується використовувати удосконалений генетичний алгоритм для управління мікрокліматом теплиці (див. п. 3.3). Генетичні алгоритми – це потужні математичні методи, які можуть бути корисними для оптимізації складних систем, таких як теплиці. Запропонований алгоритм забезпечує оптимізацію температури, вологості та освітленості для сприяння кращому росту та врожаю рослин. Зменшує витрати на енергію за рахунок більш ефективного управління ресурсами. Автоматизовує управління теплицею за рахунок динамічної адаптації до мінливих умов навколишнього середовища.

Архітектура, розроблена на рис. 3.2, використовує бездротову передачу даних Zigbee, що вирішує проблему незручної проводки в теплиці. Вузли можна вільно додавати і видаляти, а зону моніторингу можна збільшувати або зменшувати, що приносить зручність для моніторингу стану мікроклімату теплиці.

3.3. Удосконалений адаптивний генетичний алгоритм

Процес вдосконалення адаптивного генетичного алгоритму виглядає наступним чином, як показано на рис. 3.3 [11].

Удосконалення полягає у наступному:

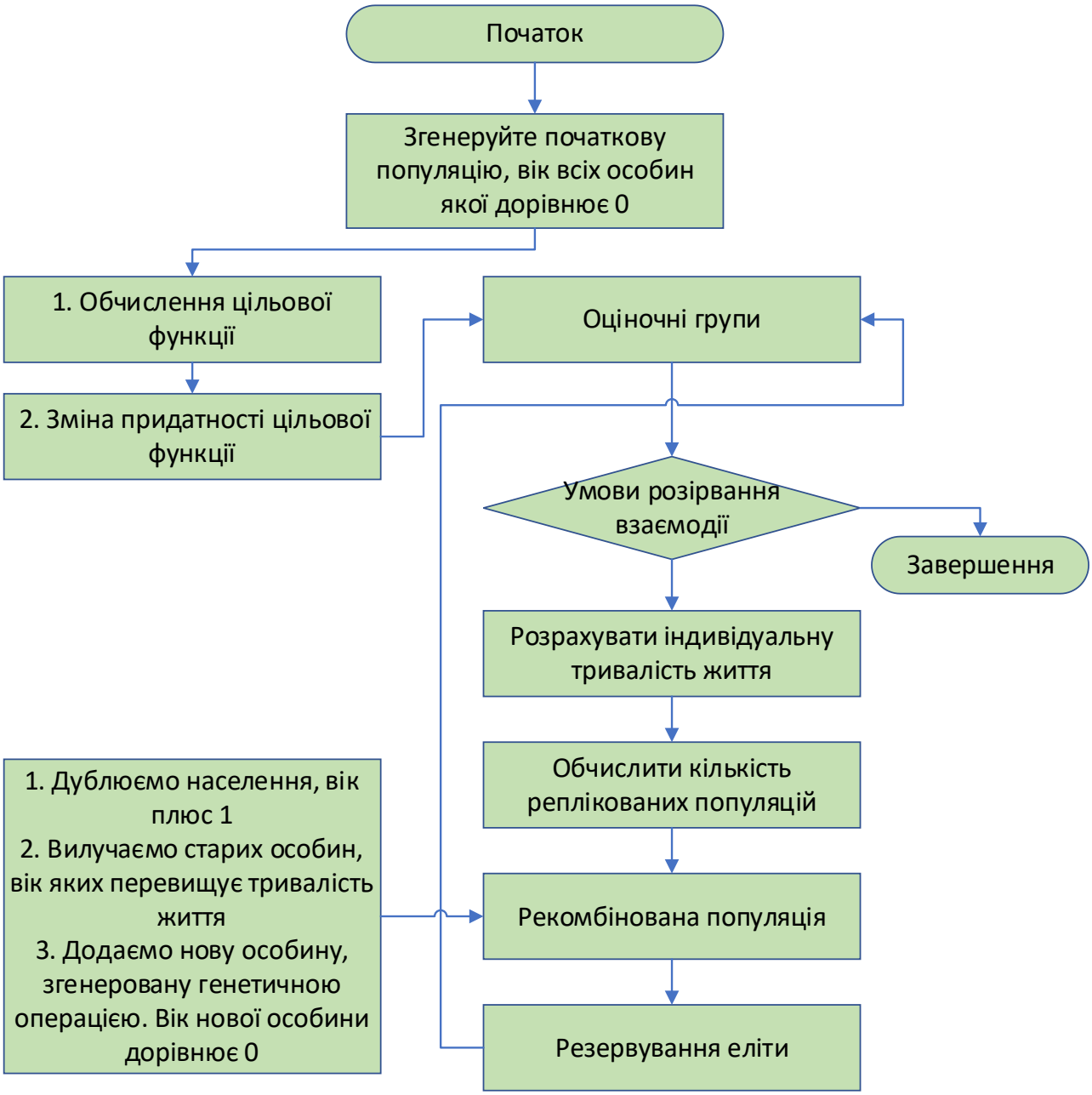

Рисунок 3.3 – Удосконалений адаптивний генетичний алгоритм

a. Відповідно до фактичної проблеми, особини кодуються для створення початкової популяції, а вік усіх особин встановлюється рівним 0;

b. Значення цільової функції особин поточної популяції обчислюється і перетворюється у фітнес;

c. Алгоритм оцінює, чи виконується умова припинення. Якщо так, то виводиться результат, якщо ні, то переходимо до кроку (d);

d. Відповідно до фітнесу особини розраховується її тривалість життя;

e. Відповідно до розміру популяції обчислюється кількість реплікованих популяцій;

f.

g. Популяція реорганізується:

1. Поточна популяція реплікується, а вік реплікованої особини збільшується на 1;

2. Особи, вік яких перевищує очікувану тривалість життя, видаляються;

3. Відповідно до алгебри особини, що залишилася, обчислюється частота кросинговеру та частота мутацій, операції кросинговеру та мутацій виконуються над реплікованою популяцією, а отримана нова особина додається до популяції, при цьому вік нової особини встановлюється рівним 0;

h. Зберігається еліта, тобто порівнюється пристосованість особин до і після рекомбінації, і найкращі особини зберігаються.

i. Повертаємось до кроку (c).

РОЗДІЛ 4.
РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ ТЕПЛИЦЯМИ ІЗ ВИКОРИСТАННЯМ ГЕНЕТИЧНОГО АЛГОРИТМУ

4.1. Розробка модуля для створення діалового вікна користувача інтелектуальної системи

Для створення графічного інтерфейсу на Python використано бібліотеку PyQt або PySide, яка є офіційним набором Python для фреймворку Qt. У додатку А наведено код для розробки аналогічного додатку на Python з використанням PyQt.

Насамперед нами написано код для імпорту необхідних бібліотек та модулів, що забезпечують створення графічного інтерфейсу користувача (GUI) з використанням PyQt5, для відображення графіків за допомогою Matplotlib та Plotly, а також для зберігання даних у форматі JSON (рис. 4.1).

```
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QLabel, QVBoxLayout, QWidget, QPushButton, QDialog,
    QLineEdit, QFormLayout, QMessageBox, QFileDialog
)
from PyQt5.QtCore import QTimer
import random
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import plotly.io as pio
import json
```

Рисунок 4.1 – Фрагмент коду для імпорту необхідних бібліотек та модулів, що забезпечують створення графічного інтерфейсу користувача (GUI)

Для імпорту бібліотек використовується sys, що забезпечує доступ до аргументів командного рядка. Для створення графіків використано matplotlib.pyplot і FigureCanvasQTAgg. Для створення інтерактивних графіків використано plotly.graph_objs, plotly.subplots, plotly.io. Окрім того передбачено використання json для роботи з JSON-файлами. Для створення графічного інтерфейсу використовується PyQt5.QtWidgets і PyQt5.QtCore.

Передбачено створення різних діалогових вікон в графічному інтерфейсі користувача для системи управління теплицею.

Діалогове вікно, що забезпечує вибір рослин (Select Plants) (рис. 4.2).

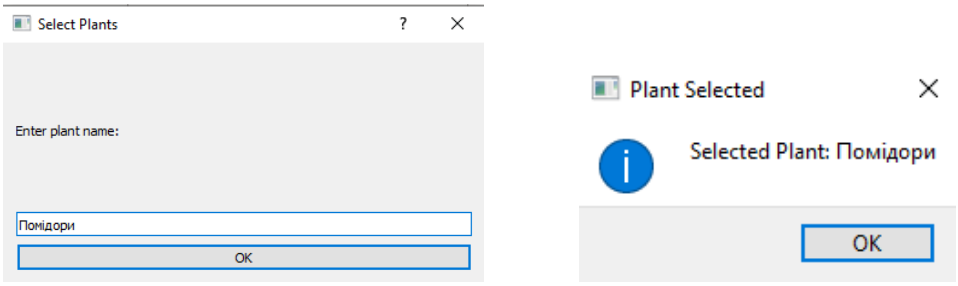

Рисунок 4.2 – Діалогове вікно, що забезпечує вибір рослин (Select Plants)

Запропоновано діалогове вікно дозволяє користувачеві ввести назву рослини, яку потрібно вибрати для управління. Для цього слід натиснути кнопку "Select Plants" на головному екрані. Відкриється діалогове вікно з назвою "Select Plants". Введіть назву рослини у текстове поле. Натисніть кнопку "OK" для підтвердження. Відобразиться повідомлення з назвою вибраної рослини. Приклад, ввести "Помідори" у текстове поле і натисніть "OK". Ви побачите повідомлення "Selected Plant: Помідори ".

Діалогове вікно, що забезпечує видалення рослини (Remove Plant) (рис. 4.3).

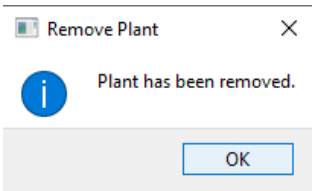

Рисунок 4.3 – Діалогове вікно, що забезпечує видалення рослини (Remove Plant)

Ця функція дозволяє видалити вибрану рослину. Для цього слід натиснути на кнопку "Remove Plant" на головному екрані. Відобразиться повідомлення, що рослину було видалено. Наприклад, натиснувши на кнопку "Remove Plant", побачите повідомлення "Plant has been removed."

Діалогове вікно, що забезпечує ручне введення параметрів (Set Parameters Manually) (рис. 4.4).

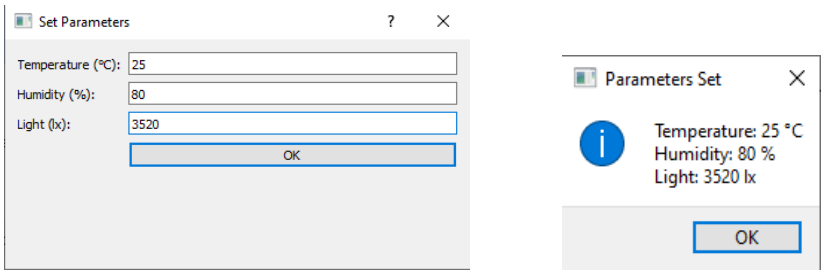

Рисунок 4.4 – Діалогове вікно, що забезпечує ручне введення параметрів (Set Parameters Manually)

Це діалогове вікно дозволяє користувачеві вручну ввести параметри теплиці, такі як температура, вологість та освітленість. Для цього слід натиснути на кнопку "Set Parameters Manually" на головному екрані. Відкриється діалогове вікно з назвою "Set Parameters". Після цього слід ввести значення для температури, вологості та освітленості у відповідні текстові поля. Натисніть кнопку "OK" для підтвердження. Відобразиться повідомлення з введеними значеннями параметрів. Наприклад, ввівши "25" у поле "Temperature (°C)", "80" у поле "Humidity (%)" та "3520" у поле "Light (lx)" та натиснувши на "OK" побачимо повідомлення:

"Temperature: 25 °C

Humidity: 80 %

Light: 3520 lx".

Також є можливість отримання звіту (Get Report). Ця функція дозволяє користувачеві отримати звіт з поточними даними про параметри теплиці (рис. 4.5). Для цього слід натиснути кнопку "Get Report" на головному екрані. Відобразиться діалогове вікно з поточними значеннями температури, вологості та освітленості. Наприклад, натиснувши кнопку "Get Report" побачимо звіт з поточними даними – "Temperature Data: [дані], Humidity Data: [дані], Light Data: [дані]".

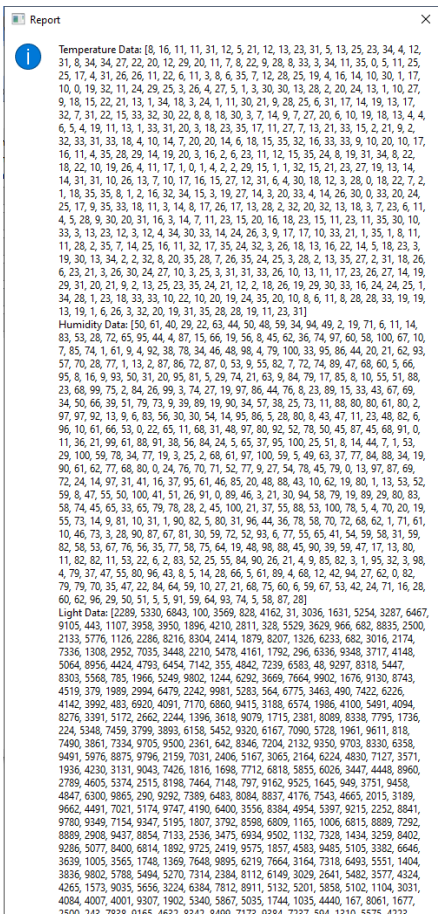

Рисунок 4.5 – Діалогове вікно, що забезпечує отримання звіту (Get Report)

Діалогове вікно, що забезпечує збереження даних (Save Data) представлено на рис. 4.6.

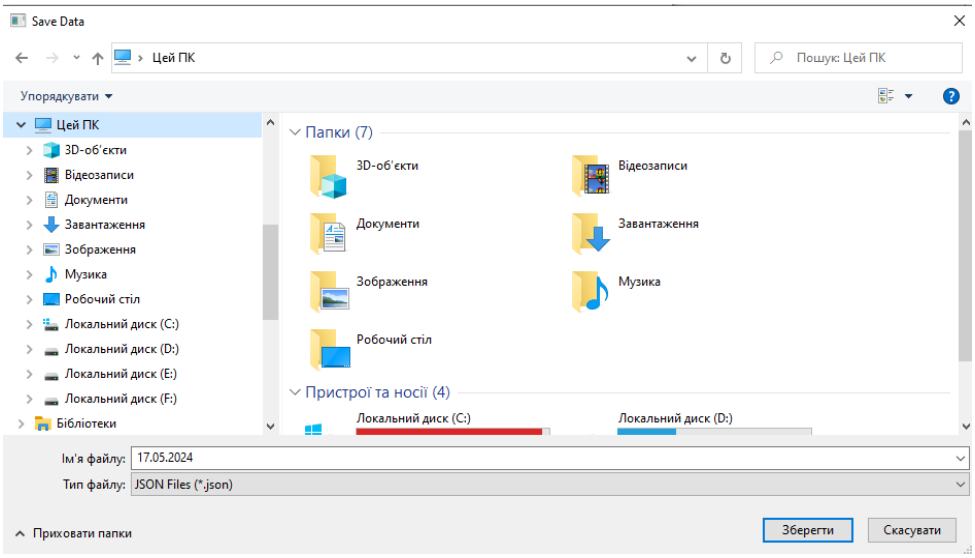

Рисунок 4.6 – Діалогове вікно, що забезпечує збереження даних (Save Data)

Панель користувача інтелектуальної системи представлено на рис. 4.7.

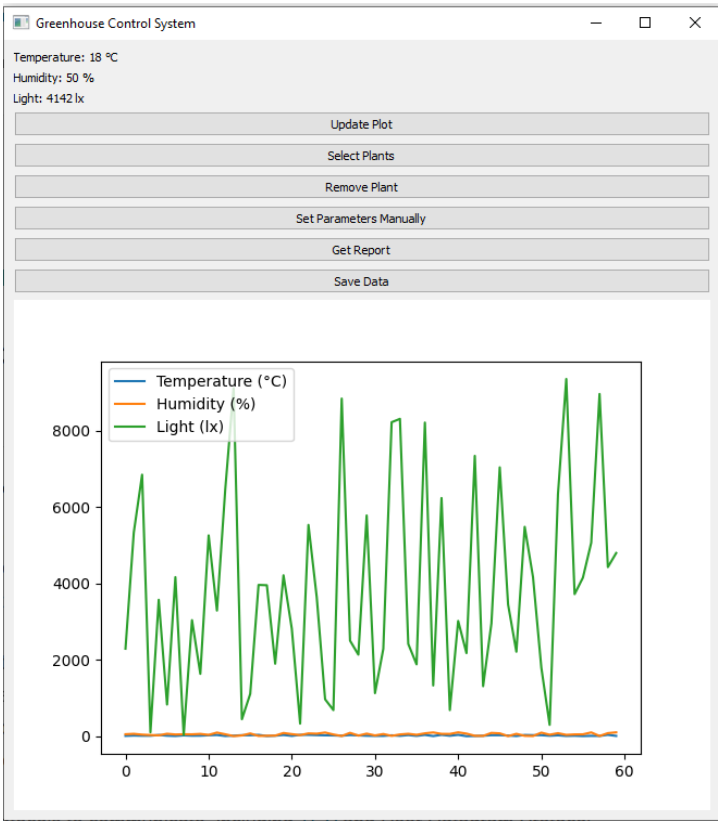

Рисунок 4.7 – Панель користувача інтелектуальної системи управління теплицею

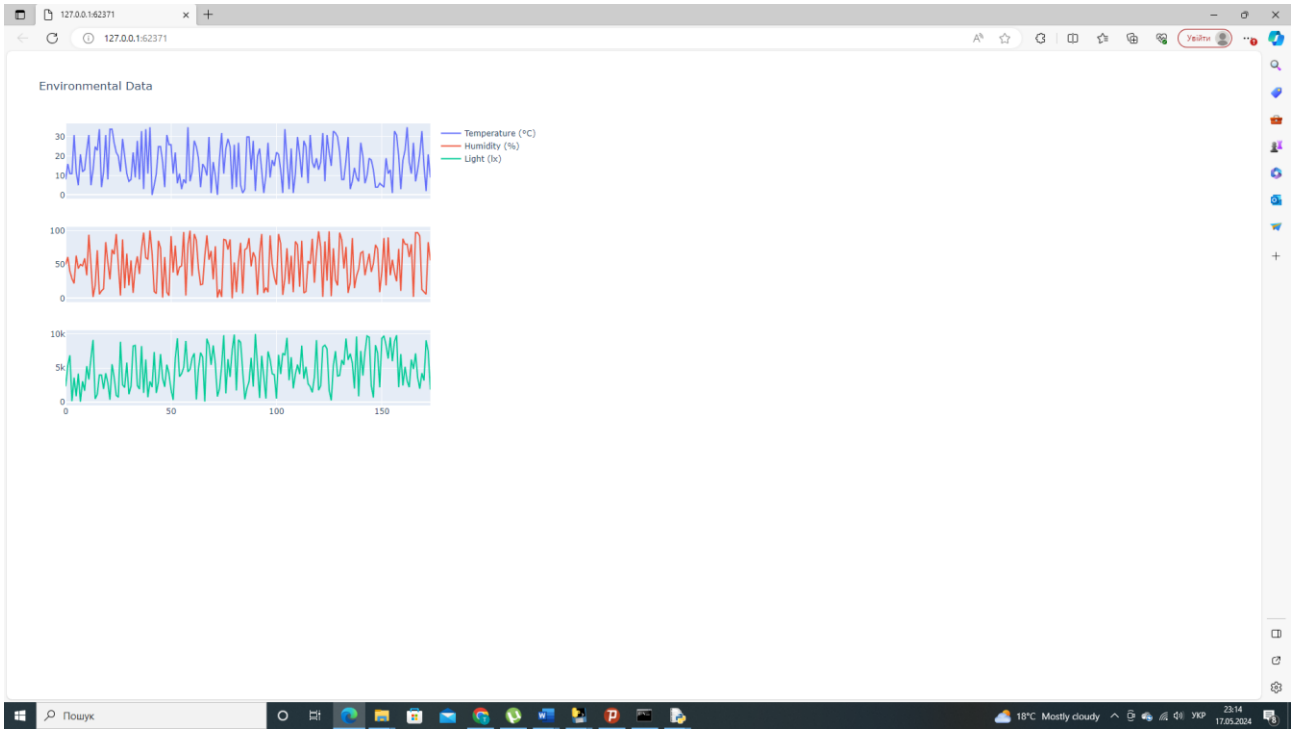

Рисунок 4.8 – Панель із графіками, що відображають динаміку зміни показників мікроклімату теплиці

Ця функція дозволяє користувачеві зберегти зібрані дані у файл у форматі JSON. Для її використання слід натиснути кнопку "Save Data" на головному екрані. Відкриється діалогове вікно для збереження файлу. Потрібно вибрати місце збереження файлу та введіть ім'я файлу. Натисніть кнопку "Save" для підтвердження. Відобразиться повідомлення, що дані було успішно збережено.

4.2. Розробка модуля контролю для реалізації PID-алгоритмів і генетичного алгоритму

Щоб створити модуль для реалізації PID-алгоритмів і генетичного алгоритму, нами розділено код на дві частини: PID-контролер і генетичний алгоритм. Для цього використано Python і бібліотеки NumPy для числових обчислень та DEAP (Distributed Evolutionary Algorithms in Python) для реалізації генетичного алгоритму.

```python
import numpy as np

class PIDController:
    def __init__(self, kp, ki, kd, setpoint):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.setpoint = setpoint
        self.integral = 0
        self.previous_error = 0

    def update(self, measured_value, dt):
        error = self.setpoint - measured_value
        self.integral += error * dt
        derivative = (error - self.previous_error) / dt

        output = self.kp * error + self.ki * self.integral + self.kd * derivative

        self.previous_error = error
        return output
```

Рисунок 4.9 – Фрагмент коду PID-контролер

Модуль для реалізації PID-алгоритмів передбачає клас PIDController (рис. 4.9). Ініціалізується з коефіцієнтами пропорційного, інтегрального і диференціального компонентів (kp, ki, kd) та встановленою точкою (setpoint).

Використовується метод update, що розраховує вихід контролера на основі поточної помилки, інтегралу помилки та похідної помилки.

Для управління теплицею із використанням генетичного алгоритму написано код, фрагмент якого представлено на рис. 4.10.

```
from deap import base, creator, tools, algorithms
import random

# Create types
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

# Parameters
BOUND_LOW, BOUND_UP = 0, 10
NUM_GENERATIONS = 40
POPULATION_SIZE = 50
MUTATION_PROB = 0.2
CROSSOVER_PROB = 0.5

# Evaluation function
def evaluate(individual):
    kp, ki, kd = individual
    pid = PIDController(kp, ki, kd, setpoint=1.0)
    setpoint = 1.0
    dt = 0.1
    total_error = 0
    measured_value = 0

    for _ in range(100):
        control = pid.update(measured_value, dt)
        measured_value += control  # Simulate process response
        total_error += abs(pid.setpoint - measured_value)

    return total_error,
```

Рисунок 4.10 – Фрагмент коду для управління теплицею із використанням генетичного алгоритму

Використовується DEAP для створення класів для індивідуумів та їхньої відповідної фізичної підготовки. Функція evaluate оцінює індивідуумів на основі їхньої здатності мінімізувати помилку контролю PID.

Для налаштування генетичного алгоритму виконується ініціалізація населення та операторів кросовера, мутації та відбору. Генетичний алгоритм використовується також для налаштування параметрів PID-контролера. Головна функція запускає генетичний алгоритм, збирає статистику та виводить найкращого індивідуума (найкращий набір параметрів PID).

4.3. Створення модуля для обробки даних, інтеграції алгоритмів і роботи з базою даних SQLite

Нами написано код Python для створення модуля, який обробляє дані, інтегрує PID-алгоритми та генетичний алгоритм, а також працює з базою даних SQLite (рис. 4.11).

```
# SQLite Database Interaction
class Database:
    def __init__(self, db_name):
        self.conn = sqlite3.connect(db_name)
        self.cursor = self.conn.cursor()
        self.create_table()

    def create_table(self):
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS greenhouse_data (
                id INTEGER PRIMARY KEY,
                timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
                temperature REAL,
                humidity REAL,
                light REAL
            )
        ''')
        self.conn.commit()

    def insert_data(self, temperature, humidity, light):
        self.cursor.execute('''
            INSERT INTO greenhouse_data (temperature, humidity, light)
            VALUES (?, ?, ?)
        ''', (temperature, humidity, light))
        self.conn.commit()

    def fetch_data(self):
        self.cursor.execute('SELECT * FROM greenhouse_data')
        return self.cursor.fetchall()

    def close(self):
        self.conn.close()

# Integration and Data Processing
class GreenhouseControl:
    def __init__(self, db_name):
        self.db = Database(db_name)
        self.pid = PIDController(0.0, 0.0, 0.0, setpoint=1.0)

    def tune_pid(self):
        best_individual = genetic_algorithm()
        self.pid.kp, self.pid.ki, self.pid.kd = best_individual
        print(f'Tuned PID parameters: Kp={self.pid.kp}, Ki={self.pid.ki}, Kd={self.pid.kd}')
```

Рисунок 4.11 – Фрагмент коду для обробки даних, інтеграції алгоритмів і роботи з базою даних SQLite

Структура модуля наступна:

- ✓PID-контролер;
- ✓Генетичний алгоритм для налаштування PID;
- ✓Робота з базою даних SQLite;
- ✓Інтеграція та обробка даних.

Клас PIDController реалізує PID-контролер з методами для оновлення контролю на основі виміряного значення і заданого часу.

Функція evaluate оцінює індивідуумів на основі їх здатності мінімізувати помилку контролю PID. Функція genetic_algorithm використовує бібліотеку DEAP для налаштування параметрів PID-контролера.

Клас Database забезпечує взаємодію з базою даних SQLite, включаючи створення таблиці, вставку даних і отримання даних. Клас GreenhouseControl інтегрує PID-контролер, генетичний алгоритм і базу даних для управління параметрами теплиці.

Метод tune_pid використовує генетичний алгоритм для налаштування параметрів PID. Метод update_parameters оновлює параметри теплиці та записує дані в базу даних. Метод get_report отримує дані з бази даних. Метод close закриває з'єднання з базою даних.

4.4. Використання інтелектуальної системи управління теплицями на основі генетичного алгоритму

Використання інтелектуальної системи управління теплицями на основі генетичного алгоритму дозволяє досягти наступних результатів, які забезпечують підвищення ефективності діяльності у теплицях.

За допомогою генетичного алгоритму система налаштовує параметри PID-контролера (Kp, Ki, Kd), що забезпечує:

- ✓ Підвищення ефективності контролю температури, вологості та освітлення в теплиці.
- ✓ Завдяки оптимальним параметрам, знижується ймовірність перерегулювання і виникнення нестабільних коливань параметрів.
- ✓ Система швидше досягає стабільного стану після змін у навколишньому середовищі.

Окрім того, покращуються умови для росту рослин. Оптимізовані параметри PID-контролера забезпечують більш точний і стабільний контроль умов у теплиці. Підтримка оптимальної температури сприяє здоровому росту

рослин і зменшує ризик теплового стресу. Висока точність у підтримці вологості сприяє запобіганню розвитку захворювань рослин, таких як грибкові інфекції. Забезпечує необхідний рівень освітленості для фотосинтезу, що важливо для росту і розвитку рослин.

Інтелектуальна система дозволяє ефективніше використовувати ресурси. Оптимальне управління системами опалення, охолодження та освітлення зменшує енергоспоживання. Точний контроль вологості дозволяє використовувати воду раціональніше, що важливо в умовах обмежених водних ресурсів.

Система забезпечує безперервний моніторинг та зберігання даних про умови в теплиці. Збереження даних у базі даних дозволяє аналізувати історичні зміни параметрів і приймати обґрунтовані рішення. Можливість створювати звіти та візуалізувати дані для оцінки ефективності роботи системи і умов вирощування рослин.

Інтелектуальна система зменшує необхідність ручного втручання. Генетичний алгоритм автоматично налаштовує PID-параметри без потреби в ручній корекції. Система безперервно відстежує та регулює параметри, забезпечуючи стабільні умови вирощування.

Загалом, використання інтелектуальної системи управління теплицями на основі генетичного алгоритму призводить до значного покращення контролю умов у теплиці, підвищення продуктивності та якості рослин, ефективного використання ресурсів, а також зменшення витрат на енергоспоживання та воду. Система забезпечує автоматизацію процесів, збір та аналіз даних, що дозволяє аграріям приймати більш обґрунтовані рішення та підвищувати загальну ефективність вирощування.

РОЗДІЛ 5.
ОХОРОНА ПРАЦІ

Розглянемо основні аспекти охорони праці в контексті впровадження інтелектуальної системи управління теплицями. Автоматизація багатьох процесів зменшує необхідність ручної праці, що знижує ризик фізичних травм працівників. Інтелектуальна система забезпечує стабільні умови роботи, зменшуючи вплив несприятливих факторів (наприклад, різких коливань температури). Автоматизовані системи знижують ймовірність помилок, викликаних людським фактором.

Інтелектуальні системи управління теплицями на основі генетичних алгоритмів не лише підвищують ефективність аграрного виробництва, але й покращують умови праці та безпеку персоналу. Впровадження таких систем вимагає ретельного планування та навчання персоналу, але в довгостроковій перспективі це приносить значні вигоди.

5.1. Аналіз стану шуму у приміщенні

Характеристика джерел шуму у приміщенні наведена в таблиці 5.1.

Таблиця 5.1 – Характеристика джерел шуму

№ п/п	Найменування	Джерело небезпеки	Причини небезпек	Наслідки небезпеки
1	Сервер Dell PowerEdge T330	Елемент охолодження сервера (куллер)	Шум елемента охолодження	Емоційна роздратованість працівника, що може призвести до зменшення продуктивності його роботи

Реальні та нормативні значення рівня шуму в приміщенні наведені в таблиці 5.2.

Таблиця 5.2 – Реальні та нормативні значення рівня шуму

№ п/п	Назва параметра	Реальне значення	Нормативне значення
1	Рівень шуму, дБ	56-60	50

Засоби та заходи захисту від впливу шуму наведено у таблиці 5.3.

Таблиця 5.3 – Засоби та заходи захисту від небезпеки

№ п/п	Вид заходу	Зміст заходу	Результат заходу
1	Технічні	підтримка серверної шафи у закритому вигляді; встановлення шумоізоляційної перегородки між серверною та робочою зоною; додаткова шумоізоляція серверної шафи за допомогою ущільнювачів дверей і швів та резинових підставок під ноги шафи.	Мінімізування рівня шуму
2	Організаційні	перерви для працівника; перевірка на належний стан куллерів та шумоізоляційних матеріалів перегородки і серверної шафи.	Запобігання впливу на організм працівника шуму
3	Експлутаційні	своєчасний ремонт куллерів та заміна шумоізоляційних елементів	Захист від високого рівня шуму

5.2. Забезпечення електробезпеки

Приміщення програміста, який працює над розробленням інтелектуальної інформаційної системи планування витрат кормів та використання технічного оснащення сільськогосподарського підприємства, не відноситься до приміщень з підвищеною небезпекою. Обладнання не завдає великого навантаження на мережу. Джерела небезпеки наведено у табл. 5.4.

Таблиця 5.4 – Джерела електронебезпеки

№ п/п	Назва	Джерело небезпеки	Причини небезпеки	Наслідки небезпеки
1	Персональний комп'ютер Zevs PC i7 2600	Блок живлення	Пошкодження блоку живлення, кабеля живлення.	Ураження струмом
2	Сервер Dell PowerEdge T330	Блок живлення, деталі сервера, ща знаходяться під напругою	Пошкодження блоку живлення, кабеля живлення.	Ураження струмом
3	Роутер Asus RT-N11P	Блок живлення	Пошкодження блоку живлення, кабеля живлення	Ураження струмом
4	Джерело безперебійного живлення LogicPower LPT-W-10000RD	Блок вхідного живлення, деталі, що знаходяться під напругою	Пошкодження блоку живлення, вхідного кабеля живлення.	Ураження струмом

Дані про споживання напруги наведено у таблиці 5.5.

Таблиця 5.5 – Реальні та нормативні фактори небезпеки

№ п/п	Чинник небезпеки	Реальне значення	Нормативні значення
1	Максимальний струм	>1 А	50 мА
		220 В	45 В

Для зниження ймовірності настання небезпечної ситуації, необхідно дотримуватись заходів безпеки, які наведені в таблиці 5.6.

Таблиця 5.6 – Засоби захисту від електротравм

№ п/п	Група номенклатурних заходів з ОП	Вид заходу	Результат заходу
1	Технічні	Використання ізоляційних матеріалів для сервера та серверної шафи (ізоляційні лаки для усунення витоків струсу). Захисні заземлення (наземні комунікації). Захисне розділення електромереж (використання розділювального трансформатору). Увімкнення техніки в мережу через заземлені фільтри (мережевий фільтр Defender DFS 151). Своєчасна заміна елементів, що працюють під напругою, які вийшли з ладу	Уникнення пробою, витоків струму уникнення контакту зі струмопровідними частинами
2	Організаційні	Інструктаж з правил електробезпеки. Підтримка сухого, незапиленого приміщення з вологістю не вище 75%	Доступність знань щодо безпеки експлуатації
3	Режимні	Перевірка несправностей техніки та мережі тільки у відключеному стані;	Уникнення контакту з елементами під напругою
4	Експлуатаційні заходи	Своєчасна заміна будь-яких пошкоджених елементів	Забезпечення безпечної роботи з об’єктом

У приміщенні програміста, який працює над розробленням інтелектуальної інформаційної системи планування витрат кормів та використання технічного оснащення сільськогосподарського підприємства, виявлена наявність електронебезпеки, яка може проявлятися у вигляді витоку струму з електромережі та надання травм організму при контакті з джерелом небезпеки.

5.3. Покращення пожежної безпеки

Характеристика джерел небезпеки, які наявні у кабінеті особи, яка приймає управлінські рішення, наведено в таблиці 5.7.

Таблиця 5.7 – Джерела пожежної небезпеки

№ п/п	Найменування	Джерело небезпеки	Причини небезпек	Наслідки небезпеки
1	Персональний комп’ютер Zevs PC i7 2600	Блок живлення, деталі під напругою	Коротке замикання	Виникнення пожежі
2	Сервер Dell PowerEdge T330	Блок живлення, деталі під напругою	Коротке замикання	
3	Роутер Asus RT-N11P	Блок живлення	Коротке замикання	
4	Джерело безперебійного живлення LogicPower LPT-W-10000RD	Блок вхідного живлення, деталі під напругою	Коротке замикання	
5	Матеріали і речовини, що схильні до займання	Загорання матеріалів	Зовнішнє загорання	
6	Щільність проводки	Оплавлення ізоляції	Коротке замикання	

Характеристика вибухонебезпечності та пожежної небезпеки у кабінеті особи, яка приймає управлінські рішення, наведено у таблиці 5.8.

Таблиця 5.8 – Характеристика вибухонебезпечності та пожежної небезпеки

№ п/п	Назва	Значення
1	Клас пожежі	A,E
2	Клас зони приміщення по пожежній безпеці	Клас II-IIa
3	Категорія пожежної небезпеки	B

Засоби та заходи захисту від вибухонебезпечності та пожежі у кабінеті особи, яка приймає управлінські рішення, наведено у таблиці 5.9.

Таблиця 5.9 – Засоби та заходи захисту від пожежної небезпеки

№ п/п	Заходи	Реалізація	Результат заходу
1	Технічні	Розташування у кімнаті порошкового вогнегасника ОПУ-10. Встановлення протипожежної теплового сповіщувача FT-B	Гасіння первинних осередків займання
2	Організаційні	Проведення навчань, інструктажів з пожежної безпеки. Створення планів евакуації.	Навчання з питань безпеки при пожежі, запобігання людських жертв
3	ЗІЗ	Протигази, респіратори та маски, захисний одяг.	Запобігання отруєнню та опіків
4	Експлуатаційні	Своєчасний ремонт та заміна обладнання.	Захист від пожеж, що можуть бути викликані технічними несправностями
5	Режимні	Встановлення пропускного режиму для серверної	Захист від зовнішніх осередків займання

Для забезпечення пожежної безпеки у кабінеті особи, яка приймає управлінські рішення, присутній порошковий вогнегасник ОПУ-10, та протипожежний тепловий сповіщувач FT-B.

5.4. Інструкція з охорони праці під час використання інтелектуальної системи управління теплицями

1. Загальні положення

1.1. Ця інструкція встановлює основні вимоги з охорони праці при використанні інтелектуальної системи управління теплицями (ІСУТ).

1.2. Інструкція призначена для працівників, які безпосередньо працюють з ІСУТ, а також для керівників, відповідальних за організацію та безпеку праці в теплицях.

1.3. Всі працівники повинні пройти відповідний інструктаж з охорони праці перед початком роботи з ІСУТ.

2. Вимоги до працівників

2.1. Працівники повинні мати необхідні знання та навички для роботи з комп'ютерною технікою та програмним забезпеченням ІСУТ.

2.2. Працівники повинні бути ознайомлені з основними правилами безпеки при роботі з електрообладнанням та системами автоматизації.

3. Підготовка до роботи

3.1. Перед початком роботи необхідно перевірити стан робочого місця та обладнання, впевнитись у відсутності пошкоджень та несправностей.

3.2. Перевірити наявність та справність засобів індивідуального захисту (ЗІЗ).

3.3. Ознайомитися з планом евакуації на випадок надзвичайної ситуації.

4. Основні вимоги безпеки під час роботи

4.1. Використання електрообладнання

Переконатися у відсутності видимих пошкоджень кабелів, розеток та іншого електрообладнання.

Використовувати тільки справні електроприлади та дотримуватися інструкцій виробника.

4.2. Робота з комп'ютерною технікою та програмним забезпеченням

Регулярно оновлювати програмне забезпечення для забезпечення його стабільної роботи.

Вести облік всіх змін, що вносяться в налаштування системи.

4.3. Контроль параметрів мікроклімату

Систематично контролювати роботу датчиків температури, вологості, освітлення та інших параметрів.

Перевіряти правильність калібрування датчиків та їх функціонування.

4.4. Робота з хімічними речовинами

Дотримуватись правил зберігання та використання хімічних речовин.

Використовувати ЗІЗ при роботі з добривами та іншими хімікатами.

4.5. Технічне обслуговування

Проводити планове технічне обслуговування обладнання згідно з графіком.

У разі виявлення несправностей негайно повідомляти керівництво та відповідальних осіб.

5. Дії у разі аварійних ситуацій

5.1. При виникненні пожежі негайно повідомити відповідні служби, вимкнути електроживлення та евакуюватися згідно з планом евакуації.

5.2. У разі витоку хімічних речовин слід негайно евакуюватися, повідомити керівництво та викликати спеціальні служби для ліквідації аварії.

5.3. При виявленні несправностей у роботі ІСУТ негайно припинити роботу, повідомити керівництво та викликати технічного спеціаліста.

6. Заключні положення

6.1. Всі працівники повинні дотримуватися даної інструкції протягом всього робочого часу.

6.2. Керівництво зобов'язане забезпечити регулярний інструктаж з охорони праці та контроль за виконанням вимог безпеки.

6.3. Інструкція підлягає перегляду та оновленню не рідше одного разу на рік або в разі внесення змін у технологічний процес або обладнання.

Дотримання цієї інструкції сприятиме забезпеченню безпечних умов праці та збереженню здоров'я працівників під час використання інтелектуальної системи управління теплицями.

ВИСНОВКИ І ПРОПОЗИЦІЇ

Інтелектуальні системи управління (ІСУ) є одним із перспективних напрямків розвитку сільського господарства. Вони можуть збирати та аналізувати дані про навколишнє середовище, стан рослин та інші фактори, що впливають на ріст і розвиток рослин. Наша кваліфікаційна робота стосується обґрунтування складових та розробки інтелектуальної системи управління теплицями на основі генетичного алгоритму. Використання генетичних алгоритмів у сільському господарстві відкриває широкі перспективи для оптимізації процесів, підвищення врожайності та зниження витрат.

Сучасні інтелектуальні інформаційні системи теплиць використовуються для моніторингу, збору даних про мікрокліматичні умови та терміни, які впливають на ріст культур (рис. 1.2). Основні характеристики автоматизованих тепличних систем представлено на рис. 1.3.

Основними перевагами використання ІСУ є: підвищення врожайності та якості продукції; економія ресурсів; зниження ризиків; полегшення роботи персоналу та збільшення прибутку. ІСУ можуть допомогти фермерам збільшити свій прибуток за рахунок підвищення врожайності, економії ресурсів та зниження ризиків.

Нами виконано огляд існуючих інтелектуальних систем управління сільськогосподарськими теплицями. Датчики є основними пристроями в розумній теплиці, яка використовує IoT. Доступні різні типи, включаючи бездротові та дротові. Комерційні виробники створюють інтелектуальні системи автоматизації теплиць (рис. 1.6), щоб покращити якість і виробництво. Також відомі розумні системи віддаленого моніторингу теплиць (рис. 1.7). Використовують бездротовий зв’язок між вузлами датчиків теплиці та хмарним сховищем.

Нами проаналізовано розробки інтелектуальних систем управління із використанням генетичних алгоритмів. Недоліком відомих систем є те, що

використання запропонованих алгоритмів не є достатньо всеосяжними для адаптації до більш складних ситуацій, а точність потребує покращення.

Основною метою цього проекту є розробка та впровадження інтелектуальної системи, яка б забезпечувала автоматизоване керування кліматом у теплицях на основі генетичного алгоритму. Головні принципи такої системи полягатимуть у зборі даних з різноманітних датчиків у теплицях (температура, вологість, освітленість тощо), їх аналізі та використанні для прийняття оптимальних рішень щодо керування умовами середовища для культурних рослин.

Генетичні алгоритми використовуються для пошуку оптимальних або близьких до оптимальних рішень складних задач, особливо коли традиційні методи оптимізації можуть бути неефективними або непрактичними. Пропонується удосконалити цей алгоритм завдяки використанню динамічної адаптивної технології для коригування параметрів управління.

Нами здійснено математичний опис генетичного алгоритму з адаптивним генетичним оператором.

Пропонується здійснювати управління теплицею із використанням поновлюваних джерел енергії. Вона включає систему використання енергії вітром, систему додаткового освітлення, фотоелектричну систему електропостачання, систему вентиляції, систему підживлення рослин, систему циркуляційної води та систему збору даних та контролю. На рис. 2.2 представлена схематична діаграма моделі теплиці.

Запропоновано два контролери на основі нечіткої логіки (рис. 2.3). Першим був SISO-регулятор, для якого вхідним сигналом є переміщення, виміряне вздовж лінії дії, а вихідним – сила, прикладена в тому ж напрямку. Він має безперервну нелінійну поведінку. Тому був запропонований другий контролер, який розглядав кожен вузол як єдину MIMO-систему. Вхідними даними є радіальне положення і швидкість, а також тангенціальна швидкість, розрахована на основі переміщень, виміряних у двох напрямках.

На підставі виже сказаного нами сформульовано правила багатопараметричного нечіткого регулятора (табл. 2.1).

Управління змінною температурою тепличних культур має велике значення для покращення економічної вигоди, якості та стану сільськогосподарських культур. Регулювання змінної температури полягає в тому, щоб розділити день на чотири періоди часу для контролю температури. Щоб сприяти фотосинтезу, сповільнюючи дихання рослин, а також певною мірою запобігти шкідникам і хворобам, цільові температури повинні змінюватися від високої до низької протягом чотирьох періодів дня (табл. 2.2).

Для розробки інтелектуальної системи управління теплицею з мультиенергетичним енергопостачанням та іншими описаними функціями, нами здійснено вибір мов програмування, бібліотек та платформ для кожного з модулів.

Нами запропоновано принципову схему системи керування теплицями. система забезпечує обробку та зберігання даних у вбудованій операційній системі Linux. Передбачається використання SQLite (полегшена реляційна система керування базами даних). Окрім того, розробляється інтерактивний інтерфейс та РК-дисплея в режимі реального часу.

Архітектура системи керування теплицями передбачає використання підсистеми мультиенергетичного енергопостачання та підсистеми управління теплицями. Вона складається з наступних трьох частин: 1) мультиенергетичного енергопостачання; 2) бездротової сенсорної мережі; 3) платформи керування та програмного забезпечення.

Нами розроблено блок-схему удосконаленого адаптивного генетичного алгоритму для управління теплицею.

Для створення графічного інтерфейсу на Python використано бібліотеку PyQt або PySide, яка є офіційним набором Python для фреймворку Qt. Здійснено опис діалогового вікна користувачів.

Щоб створити модуль для реалізації PID-алгоритмів і генетичного алгоритму, нами розділено код на дві частини: PID-контролер і генетичний

алгоритм. Для цього використано Python і бібліотеки NumPy для числових обчислень та DEAP (Distributed Evolutionary Algorithms in Python) для реалізації генетичного алгоритму.

Нами написано код Python для створення модуля, який обробляє дані, інтегрує PID-алгоритми та генетичний алгоритм, а також працює з базою даних SQLite (рис. 4.11).

Використання інтелектуальної системи управління теплицями на основі генетичного алгоритму дозволяє досягти наступних результатів, які забезпечують підвищення ефективності діяльності у теплицях.

Нами розроблені заходи із безпеки праці виконавців, які працюють над створенням та використанням інтелектуальної системи управління теплицями, що забезпечують покращення умов праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вікіпедія. Генетичний алгоритм. URL: https://uk.wikipedia.org/wiki/

2. Інноваційні технології в управлінні складними біотехнологічними об’єктами агропромислового комплексу / А.П. Ладанюк, В.М. Решетюк, В.Д. Кишенько, Я.В. Смітюх. Київ : Центр учбової літератури, 2014. 280 с.

3. Корчемний М.О., Лисенко В.П. Нейронні мережі. Київ : НАУ, 2008. 156 с.

4. Лисенко В.П., Дудник А.О. Оптимальне управління: стан та перспективи розвитку в тепличній галузі. Науковий вісник Національного університету біоресурсів і природокористування України. 2011. Вип. 166. С. 53–56.

5. Програмування числових методів мовою Python : підруч. / А. В. Анісімов, А. Ю. Дорошенко, С. Д. Погорілий, Я. Ю. Дорогий ; за ред. А. В. Анісімова. К. : Видавничо-поліграфічний центр «Київський університет», 2014. 640 с.

6. Прокопенко Т.О. Інтелектуальна система керування температурно-вологісним режимом у теплиці. Науковий вісник Національного університету біоресурсів і природокористування України. Серія «Техніка та енергетика АПК». 2015. Вип. 209. Ч. І. С. 140–147.

7. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. СПб.: БХВ-Петербург, 2012. 704 с.

8. Розумні теплиці. Які можливості дає Інтернет речей теплицям. URL: https://iotji.io/solutions/rozumni-teplytsi/

9. Tryhuba A., Boyarchuk V., Tryhuba I., Ftoma O., Padyuka R., Rudynets M., Forecasting the Risk of the Resource Demand for Dairy Farms Basing on Machine Learning, Proceedings of the 2nd International Workshop on Modern Machine Learning Technologies and Data Science (MoMLeT+DS 2020). Volume I: Main Conference. Lviv-Shatsk, Ukraine, June 2-3, 2020. pp.327-340.

10. Шаманська О.І. Застосування інформаційних систем та технологій як пріоритетного напряму ефективного функціонування та розвитку дорадчої діяльності в Україні. Ефективна економіка. 2015. № 4.

11. Acuna Y., Sun Y. An efficiency-improved genetic algorithm and its application on multimodal functions and a 2D common reflection surface stacking problem. Geophys. Prospect., 68 (4), 2020, P. 1189-1210.

12. Automated Greenhouse System with Profound Analytics. URL: https://intellias.com/automated-greenhouse-system/

13. Chen Y. Location and path optimization of green cold chain logistics based on improved genetic algorithm from the perspective of low carbon and environmental protection. Fresenius Environ. Bull., 30 (6), 2021, P. 5961-5973.

14. Defoy, B., Alban, T., Mahfoud, J. Experimental assessment of a new fuzzy controller applied to a flexible rotor supported by Active Magnetic Bearings. Institution of Mechanical Engineers - 10th International Conference on Vibrations in Rotating Machinery, 2012, P. 379–388.

15. Gao David Wenzhong, Mujadi E. Guest editorial: Special section on distributed integrated multi-energy system(DIMS). J. Mod. Power Syst. Clean Energy, v.8 (05), 2020, P. 3-6.

16. Guo X., Huang J., Liu H., Chen Y. An efficient P-cycle combination protection strategy based on improved genetic algorithm in elastic optical networks. IET Optoelectron., 12 (2), 2017, P. 73-79.

17. Koval N., Tryhuba A., Kondysiuk I., Grabovets V., Onyshchuk V., Forecasting the fund of time for performance of works in hybrid projects using machine training technologies. CEUR Workshop Proceedings, 2021, 2917, pp. 196–206.

18. Li B. Energy saving optimization of machining center process route based on improved genetic algorithm. Acad. J. Manuf. Eng., 15 (3), 2017, P. 35-42.

19. Li H., Wang X., Gao Y., Liang H. Evaluation research of the energy supply system in multi-energy complementary park based on the improved universal generating function method. Energy Convers. Manage., 2018, 174, P. 955-970.

20. Maged N.A., Naser E., Bendary F. A novel approach of a single-input multi-outputs converter for a modified DC nanogrid within an open energy system. J. Electr. Eng., 3 (3), 2018, P. 90-101.

21. Malanchuk O., Tryhuba A., Tryhuba I., Sholudko R., Pankiv O., A Neural Network Model-based Decision Support System for Time Management in Pediatric Diabetes Care Projects. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2023.

22. Morelli Stefano, Cossio Filippo, Monarca Danilo, Marucci Alvaro, Selli Sara, Pierini Daniele, Carlini Maurizio. Parametric sweep simulation for greenhouse temperature field optimization: An Italian case study. Energy Rep., 8 (Supplement 9), 2022, P. 881-895.

23. Python (programming language) [Електронний ресурс]. Режим доступу: https://bitly.su/paHJe. (дата звернення: 28.04.2024)

24. Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/. (дата звернення: 12.03.2024)

25. Qiao W.Z., Mizumoto M.. PID type fuzzy controller and parameters adaptive method. Fuzzy Sets and Systems, 1996, 78, P. 23-35.

26. Tryhuba A., Ratushny R., Bashynsky O., Ptashnyk V., Development and Usage of a Computer Model of Evaluating the Scenarios of Projects for the Creation of Fire Fighting Systems of Rural Communities. 11th International Scientific and Practical Conference on Electronics and Information Technologies, ELIT 2019 - Proceedings, 2019, pp. 34–39, 8892320.

27. Smart Greenhouse Automation System. URL: https://controlbyweb.com/blog/smart-greenhouse-automation-system/

28. Smart Greenhouse Solutions: IoT-Based Environmental Monitoring and Control. URL: https://webbylab.com/blog/smart-greenhouse-solutions-iot-based-environmental-monitoring-and-control/

29. Tang W., Yang R.S., Sun Z.Y. Optimization of exhaust system parameters in drying section of paper machine based on improved genetic algorithm. China Pulp Pap. [Chung-Kuo Tsao Chih], 36 (10), 2017, P. 44-49.

30. Tryhuba A., Kondysiuk I., Tryhuba I., Boiarchuk O., Tatomyr A., Intellectual information system for formation of portfolio projects of motor transport enterprises, CEUR Workshop Proceedings, 2022, 3109, pp. 44–52.

31. Tryhuba A., Koval N., Tryhuba I., Boiarchuk O., Application of Sarima Models in Information Systems Forecasting Seasonal Volumes of Food Raw Materials of Procurement on the Territory of Communities. CEUR Workshop Proceedings, 2022, 3295, pp. 64–75.

32. Tryhuba A., Ratushny R., Horodetskyy I., Molchak Y., Grabovets V., The configurations coordination of the projects products of development of the community fire extinguishing systems with the project environment. CEUR Workshop Proceedings, 2021, 2851, pp. 238–248.

33. Tryhuba A., Ratushny R., Tryhuba I., Koval N., Androshchuk I., The model of projects creation of the fire extinguishing systems in community territories. Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis, 2020, 68(2), pp. 419–431.

34. Tryhuba A., Tryhuba I., Ftoma O., Boyarchuk O., Method of quantitative evaluation of the risk of benefits for investors of fodder-producing cooperatives. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2019, 3, pp. 55–58, 8929788.

35. Xiao L. Parameter tuning of PID controller for beer filling machine liquid level control based on improved genetic algorithm. Comput. Intell. Neurosci., 2021 (2), 2021, P. 1-10.

36. Zjup C.I., Jiang H., Ning S., Ge Q., Yu B. Optimal economic dispatching of multi-microgrids by an improved genetic algorithm. IET Cyber-Syst. Robot., 3 (1), 2021, P. 68-76.

Додатки

Додаток А

Фрагмент програмного коду створення вікна користувачів інтелектуальної системи

```
import sys
from PyQt5.QtWidgets import (
    QApplication, QMainWindow, QLabel, QVBoxLayout, QWidget, QPushButton,
QDialog,
    QLineEdit, QFormLayout, QMessageBox, QFileDialog
)
from PyQt5.QtCore import QTimer
import random
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import plotly.io as pio
import json

class PlantSelectionDialog(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Select Plants")
        self.setGeometry(100, 100, 400, 200)
        layout = QVBoxLayout()
        self.plantName = QLineEdit(self)
        layout.addWidget(QLabel("Enter plant name:"))
        layout.addWidget(self.plantName)
        okButton = QPushButton("OK", self)
        okButton.clicked.connect(self.accept)
        layout.addWidget(okButton)
        self.setLayout(layout)

class ParameterSettingDialog(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Set Parameters")
        self.setGeometry(100, 100, 400, 200)
        layout = QFormLayout()
        self.tempInput = QLineEdit(self)
        self.humidInput = QLineEdit(self)
        self.lightInput = QLineEdit(self)
        layout.addRow("Temperature (°C):", self.tempInput)
        layout.addRow("Humidity (%):", self.humidInput)
        layout.addRow("Light (lx):", self.lightInput)
        okButton = QPushButton("OK", self)
        okButton.clicked.connect(self.accept)
        layout.addWidget(okButton)
        self.setLayout(layout)

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Greenhouse Control System")

        self.temperatureLabel = QLabel("Temperature: -- °C")
        self.humidityLabel = QLabel("Humidity: -- %")
        self.lightLabel = QLabel("Light: -- lx")
```

```
        self.plotButton = QPushButton("Update Plot")
        self.selectPlantsButton = QPushButton("Select Plants")
        self.removePlantButton = QPushButton("Remove Plant")
        self.setParamsButton = QPushButton("Set Parameters Manually")
        self.getReportButton = QPushButton("Get Report")
        self.saveDataButton = QPushButton("Save Data")

        layout = QVBoxLayout()
        layout.addWidget(self.temperatureLabel)
        layout.addWidget(self.humidityLabel)
        layout.addWidget(self.lightLabel)
        layout.addWidget(self.plotButton)
        layout.addWidget(self.selectPlantsButton)
        layout.addWidget(self.removePlantButton)
        layout.addWidget(self.setParamsButton)
        layout.addWidget(self.getReportButton)
        layout.addWidget(self.saveDataButton)

        container = QWidget()
        container.setLayout(layout)

        self.setCentralWidget(container)

        self.timer = QTimer()
        self.timer.timeout.connect(self.updateData)
        self.timer.start(1000)  # Update every second

        self.plotButton.clicked.connect(self.updatePlot)
        self.selectPlantsButton.clicked.connect(self.selectPlants)
        self.removePlantButton.clicked.connect(self.removePlant)
        self.setParamsButton.clicked.connect(self.setParamsManually)
        self.getReportButton.clicked.connect(self.getReport)
        self.saveDataButton.clicked.connect(self.saveData)

        self.figure, self.ax = plt.subplots()
        self.canvas = FigureCanvas(self.figure)
        layout.addWidget(self.canvas)

        self.plotData = {
            'temperature': [],
            'humidity': [],
            'light': [],
        }

    def updateData(self):
        temperature = random.randint(0, 35)
        humidity = random.randint(0, 100)
        light = random.randint(0, 10000)

        self.plotData['temperature'].append(temperature)
        self.plotData['humidity'].append(humidity)
        self.plotData['light'].append(light)

        self.temperatureLabel.setText(f"Temperature: {temperature} °C")
        self.humidityLabel.setText(f"Humidity: {humidity} %")
        self.lightLabel.setText(f"Light: {light} lx")

    def updatePlot(self):
        self.ax.clear()
        self.ax.plot(self.plotData['temperature'], label='Temperature (°C)')
        self.ax.plot(self.plotData['humidity'], label='Humidity (%)')
        self.ax.plot(self.plotData['light'], label='Light (lx)')
        self.ax.legend()
        self.canvas.draw()
```

```
        self.updatePlotly()

    def updatePlotly(self):
        fig = make_subplots(rows=3, cols=1, shared_xaxes=True)

        fig.add_trace(go.Scatter(y=self.plotData['temperature'], mode='lines',
name='Temperature (°C)'), row=1, col=1)
        fig.add_trace(go.Scatter(y=self.plotData['humidity'], mode='lines',
name='Humidity (%)'), row=2, col=1)
        fig.add_trace(go.Scatter(y=self.plotData['light'], mode='lines',
name='Light (lx)'), row=3, col=1)

        fig.update_layout(height=600, width=800, title_text="Environmental
Data")
        pio.show(fig)

    def selectPlants(self):
        dialog = PlantSelectionDialog()
        if dialog.exec_() == QDialog.Accepted:
            plant_name = dialog.plantName.text()
            QMessageBox.information(self, "Plant Selected", f"Selected Plant:
{plant_name}")

    def removePlant(self):
        QMessageBox.information(self, "Remove Plant", "Plant has been removed.")

    def setParamsManually(self):
        dialog = ParameterSettingDialog()
        if dialog.exec_() == QDialog.Accepted:
            temp = dialog.tempInput.text()
            humid = dialog.humidInput.text()
            light = dialog.lightInput.text()
            QMessageBox.information(self, "Parameters Set", f"Temperature:
{temp} °C\nHumidity: {humid} %\nLight: {light} lx")

    def getReport(self):
        report = (
            f"Temperature Data: {self.plotData['temperature']}\n"
            f"Humidity Data: {self.plotData['humidity']}\n"
            f"Light Data: {self.plotData['light']}"
        )
        QMessageBox.information(self, "Report", report)

    def saveData(self):
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getSaveFileName(self, "Save Data", "", "JSON
Files (*.json);;All Files (*)", options=options)
        if fileName:
            with open(fileName, 'w') as file:
                json.dump(self.plotData, file)
            QMessageBox.information(self, "Save Data", "Data has been saved
successfully.")

if __name__ == "__main__":
    app = QApplication(sys.argv)

    window = MainWindow()
    window.show()

    sys.exit(app.exec_())
```

Фрагмент програмного коду PID Controller

```
import numpy as np

class PIDController:
    def __init__(self, kp, ki, kd, setpoint):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.setpoint = setpoint
        self.integral = 0
        self.previous_error = 0

    def update(self, measured_value, dt):
        error = self.setpoint - measured_value
        self.integral += error * dt
        derivative = (error - self.previous_error) / dt

        output = self.kp * error + self.ki * self.integral + self.kd *
derivative

        self.previous_error = error
        return output
```

Фрагмент програмного коду Genetic Algorithm for PID Tuning

```
from deap import base, creator, tools, algorithms
import random

# Create types
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", list, fitness=creator.FitnessMin)

# Parameters
BOUND_LOW, BOUND_UP = 0, 10
NUM_GENERATIONS = 40
POPULATION_SIZE = 50
MUTATION_PROB = 0.2
CROSSOVER_PROB = 0.5

# Evaluation function
def evaluate(individual):
    kp, ki, kd = individual
    pid = PIDController(kp, ki, kd, setpoint=1.0)
    setpoint = 1.0
    dt = 0.1
    total_error = 0
    measured_value = 0

    for _ in range(100):
        control = pid.update(measured_value, dt)
        measured_value += control  # Simulate process response
        total_error += abs(pid.setpoint - measured_value)

    return total_error,

# Genetic Algorithm Setup
toolbox = base.Toolbox()
toolbox.register("attr_float", random.uniform, BOUND_LOW, BOUND_UP)
toolbox.register("individual", tools.initRepeat, creator.Individual,
toolbox.attr_float, 3)
```

```
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("mate", tools.cxBlend, alpha=0.5)
toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=1, indpb=0.2)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", evaluate)

def main():
    population = toolbox.population(n=POPULATION_SIZE)
    hof = tools.HallOfFame(1)

    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("avg", np.mean)
    stats.register("std", np.std)
    stats.register("min", np.min)
    stats.register("max", np.max)

    population, log = algorithms.eaSimple(population, toolbox,
cxpb=CROSSOVER_PROB, mutpb=MUTATION_PROB,
                                          ngen=NUM_GENERATIONS, stats=stats,
halloffame=hof, verbose=True)

    return population, log, hof

if __name__ == "__main__":
    population, log, hof = main()
    print("Best individual is:", hof[0])
```