Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи
Назва теми

Рівень вищої освіти *другий (магістерський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 240118.24.01.35 ПЗ

Виконав здобувач ІІ курсу, група КІ2м-24-1 _____ Вадим ЛІСОВИЙ
Підпис Ініціали, прізвище

Керівник д. техн. наук, професор _____ Василь ЯЦКІВ
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер д. техн. наук, професор _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
« 07 » травня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ДРУГИЙ (МАГІСТЕРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 12 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Лісовому Вадиму Миколайовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи
Керівник проекту (роботи) Яцків Василь Васильович, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 12.01.2026 р. № 6

2. Термін подання здобувачем роботи на кафедру 01.05.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

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

Аналіз предметної області, існуючих методів керування мікрокліматом в агропромисловому секторі та обґрунтування використання прогнозуючих моделей штучного інтелекту

Проєктування загальної архітектури гібридної Cloud-Edge системи, розробка апаратного забезпечення граничного вузла та розподілу енергетичних ліній з гальванічною ізоляцією навантажень

Розробка математичного та алгоритмічного забезпечення системи: розробка багатокритеріального нечіткого регулятора для упереджувального керування та імплементація моделі машинного навчання ARIMA для прогнозування часових рядів

Програмно-апаратна реалізація розробленої системи та експериментальне дослідження її ефективності, енергозбереження і алгоритмів відмовостійкості

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

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання « 12 » 01 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	23.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	23.01.2026	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	03.03.2026	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	06.03.2026	виконано
5	Робота над науковими тезами	07.03.2026	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	27.03.2026	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	07.05.2026	виконано
8	Оформлення пояснювальної записки згідно вимог	08.05.2026	виконано
9	Попередній захист ДРМ	12.05.2026	виконано
10	Захист ДРМ на засіданні ЕК	До 15.05.2026	

Здобувач _____ Вадим ЛІСОВИЙ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Василь ЯЦКІВ
Підпис Ім'я, ПРІЗВИЩЕ

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи.

Автор роботи: Лісовий Вадим Миколайович.

Керівник роботи: Яцків Василь Васильович.

Пояснювальна записка: 98 с., 37 рис., 3 табл., 6 дод., 80 джерел.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ: ІНТЕРНЕТ РЕЧЕЙ, МІКРОКЛІМАТ ТЕПЛИЦІ, НЕЧІТКА ЛОГІКА, ПРОГНОЗУЮЧЕ КЕРУВАННЯ, ARIMA, ESP32, ХМАРНІ ОБЧИСЛЕННЯ, ВІДМОВОСТІЙКІСТЬ.

Об’єктом дослідження є процес автоматизованого керування мікрокліматичними параметрами теплиці.

Предметом дослідження є апаратно-програмні засоби, моделі машинного навчання та алгоритми нечіткої логіки для прогнозування й оптимізації мікроклімату теплиці.

Метою кваліфікаційної роботи магістра є підвищення енергоефективності та надійності системи автоматизованого керування мікрокліматом теплиці шляхом розробки Cloud-Edge архітектури з використанням методів прогнозуючою аналітики та упереджувального нечіткого керування.

Для розв’язання поставлених задач використовувалися методи: теорії автоматичного керування (для розробки контурів регулювання), теорії нечітких множин (для реалізації алгоритму Мамдані), статистичного аналізу та машинного навчання (для прогнозування часових рядів за моделлю ARIMA), системного аналізу та об’єктно-орієнтованого програмування (для розробки багаторівневого програмного забезпечення).

Наукова новизна отриманих результатів:

− набув подальшого розвитку метод прогнозуючого керування мікрокліматом, який, на відміну від існуючих, використовує двопортовий нечіткий регулятор (з одночасною оцінкою поточної та прогнозованої похибки від моделі

ARIMA), що дозволяє реалізувати упереджувальне керування інерційними виконавчими механізмами;

– набула подальшого розвитку інформаційна технологія забезпечення відмовостійкості в IoT-системах за рахунок впровадження алгоритму безпечної деградації прогнозу та локального збереження уставок у незалежній пам'яті, що гарантує автономне виживання біологічних об'єктів при втраті зв'язку з хмарним сервером.

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення гібридної Cloud-Edge системи, що включає хмарне інтелектуальне ядро (на базі Flask/Waitress, СУБД MySQL та брокера MQTT) для ресурсоємної аналітики, а також апаратно-програмний граничний вузол (на базі мікроконтролера ESP32) із розробленою архітектурою неблокуючого кінцевого автомата.

Практична значимість отриманих результатів полягає у створенні готового до дослідно-промислової експлуатації програмно-апаратного комплексу керування теплицею, який знижує пікові енергозатрати, алгоритмічно захищає силову електроніку за рахунок математично обґрунтованого ШІМ-обмеження та надає агроному зручний HMI-дашборд для інструментального контролю за процесом вирощування.

У першому розділі проведено комплексний аналіз предметної області, розглянуто існуючі підходи до автоматизації теплиць, визначено недоліки класичних релейних систем керування та обґрунтовано доцільність використання IoT-інфраструктури в поєднанні з прогнозуючими алгоритмами ШІ.

У другому розділі розроблено загальну Cloud-Edge архітектуру системи, обґрунтовано вибір апаратної платформи (ESP32, датчики, MOSFET-модулі), спроектовано топологію централізованого живлення з гальванічною ізоляцією індуктивних навантажень та розраховано апаратне обмеження споживаної потужності.

У третьому розділі розроблено математичне забезпечення системи: розроблено базу правил нечіткого регулятора Мамдані для керування "за

відхиленням", описано алгоритм машинного навчання ARIMA для аналізу часових рядів та формалізовано методи статистичної оцінки точності.

У четвертому розділі описано практичну програмну реалізацію хмарного бекенду та граничного вузла, розроблено інтерактивний HMI-інтерфейс оператора. Наведено результати експериментального SITL-тестування, які підтвердили високу точність прогнозуючої моделі та алгоритмічну стійкість розроблених методів енергозбереження.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

TTL – Time to Live (термін придатності)

MPC – Model Predictive Control (модельно-прогнозуваьне керування)

IoT – Internet of Things (інтернет речей)

ПІД – Пропорційно-Інтегрально-Диференціальний

CV – Computer Vision (комп’ютерний зір)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

CFD – Computational Fluid Dynamics (обчислювальна гідродинаміка)

RNN – Reccurent Neural Network (рекурентна нейронна мережа)

LSTM – Long Short-Term Memory (довга короткочасна пам’ять)

GA – Genetic Algorithm (генетичний алгоритм)

EECSI – Energy-Efficient Clustered Smart Irrigation (енергоефективний кластерний розумний полив)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

CoAP – Constrained Application Protocol (протокол обмеженого застосування)

MQTT – Message Queuing Telemetry Transport (мережевий протокол обміну повідомленнями)

ACF – AutoCorrelation Function (автокореляційна функція)

PACF – Partial AutoCorrelation Function (часткова автокореляційна функція)

AIC – Akaike Information Criterion (інформаційний критерій Акаіке)

BIC – Bayesian Information Criterion (Байєвський інформаційний критерій)

FLC – Fuzzy Logic Control (регулятор на базі нечіткої логіки)

CoG – Center of Gravity (центр ваги)

RMSE – Root Mean Square Error (середньоквадратична похибка)

MAE – Mean Absolute Error (середня абсолютна похибка)

ADF – Augmented Dickey-Fuller Test (розщирений тест Дікі-Фулера)

IAE – Integral Absolute Error (інтегральний критерій абсолютної похибки)

HMI – Human-Machine Interface (людино-машинний інтерфейс)

ВСТУП

В умовах глобальних кліматичних змін та стрімкого зростання населення планети, забезпечення продовольчої безпеки стає одним із найважливіших проблем для людства. Традиційне землеробство є вразливим до погодних аномалій, що зумовлює інтенсивний розвиток сільського господарства та теплиць. Однак, сучасні теплиці є надзвичайно енергоємними об'єктами, а мікроклімат у них – це складна динамічна система з високим ступенем інерційності та нелінійності взаємозв'язків між температурою, вологістю та освітленням.

Класичні системи автоматизації, побудовані на базі релейних регуляторів, призводять до значних перевитрат електроенергії, зносу обладнання та температурних коливань, що викликають стрес у рослин. Вирішення цієї проблеми лежить у сфері Інтернету речей та штучного інтелекту, зокрема – у переході від реактивного до упереджувального керування. Впровадження гібридних Cloud-Edge архітектур дозволяє поєднати потужні хмарні обчислення для аналітики великих даних із локальною надійністю мікроконтролерів, створюючи енергоефективні та відмовостійкі біотехнічні системи нового покоління.

Актуальність роботи полягає в необхідності розробки сучасних апаратно-програмних комплексів керування мікрокліматом, які здатні не лише фіксувати відхилення параметрів, але й прогнозувати їх, завчасно адаптуючи роботу виконавчих механізмів. Використання машинного навчання у поєднанні з нечіткою логікою на межі мережі дозволяє оптимізувати енерговитрати та усунути критичні вразливості централізованих хмарних систем при втраті зв'язку.

Метою кваліфікаційної роботи магістра є підвищення енергоефективності та надійності системи автоматизованого керування мікрокліматом теплиці шляхом розробки Cloud-Edge архітектури з використанням методів прогнозуючо аналітики та упереджувального нечіткого керування.

Поставлена мета досягається розв'язанням таких основних завдань:

- провести аналіз існуючих методів та технологій керування мікрокліматом в агропромисловому секторі;

- розробити загальну архітектуру Cloud-Edge системи, спроектувати апаратне забезпечення граничного вузла та оптимізувати розподіл ліній живлення з ізоляцією індуктивних навантажень;

- розробити метод та математичне забезпечення системи: розробити прогнозуючу модель машинного навчання для аналізу часових рядів та двопортовий нечіткий регулятор керування "за відхиленням";

- розробити програмне забезпечення хмарного сервера та мікроконтролера;

- провести експериментальне дослідження розробленого методу для оцінки точності прогнозування та ефективності алгоритмів відмовостійкості.

Об’єктом дослідження є процес автоматизованого керування мікрокліматичними параметрами в умовах теплиці.

Предметом дослідження є методи, апаратно-програмні засоби, моделі машинного навчання та алгоритми нечіткої логіки для прогнозування й оптимізації мікроклімату теплиці.

Наукова новизна отриманих результатів:

- набув подальшого розвитку метод прогнозуючого керування мікрокліматом, який, на відміну від існуючих, використовує двопортовий нечіткий регулятор, що дозволяє реалізувати упереджувальне керування інерційними виконавчими механізмами;

- набула подальшого розвитку інформаційна технологія забезпечення відмовостійкості в IoT-системах за рахунок алгоритму безпечної деградації прогнозу та збереження уставок у незалежній пам’яті, що гарантує автономне виживання біологічних об’єктів при втраті зв’язку.

Практична цінність отриманих результатів полягає у створенні готового до дослідно-промислової експлуатації програмно-апаратного комплексу. Розроблено відмовостійкий граничний вузол на базі ESP32 з математично обґрунтованим ШІМ-обмеженням потужності Пельтьє та хмарний веб-додаток із багатопотоковою архітектурою, захистом ШІ від математичного зависання та інтерактивним інструментарієм аналізу енергоефективності.

Для розв’язання поставлених задач використовувалися методи: теорії автоматичного керування, теорії нечітких множин, статистичного аналізу та машинного навчання, системного аналізу та об’єктно-орієнтованого програмування.

За темою кваліфікаційної роботи опубліковано наукові тези [55] у 17-й Міжнародній студентській науковотехнічній конференції «Перспективні мережеві та комп’ютерні технології» – ПЕРСИК 2026 (м. Харків, 23 квіт. 2026). Харків, 2026.

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ СИСТЕМИ КЕРУВАННЯ МІКРОКЛІМАТОМ ТЕПЛИЦІ З ПРОГНОЗУВАННЯМ ТА ОПТИМІЗАЦІЄЮ РЕЖИМІВ РОБОТИ

1.1 Аналіз особливостей керування складними нелінійними об’єктами на прикладі теплиць

Сучасні тепличні комплекси належать до класу складних динамічних систем з розподіленими параметрами, керування якими пов’язане з низкою суттєвих труднощів. На відміну від простих замкнутих просторів, мікроклімат теплиці формується під впливом біологічних процесів та агресивного впливу зовнішнього середовища.

Як зазначають у дослідженні [1, 56, 57, 58, 59], забезпечення оптимальних умов для росту рослин вимагає підтримання температури та вологості у вузьких діапазонах, специфічних для кожної культури. Проте, досягнення цих показників ускладнюється тим, що теплиця є нелінійною системою, параметри якої змінюються в часі. Автори підкреслюють, що традиційні методи керування часто не здатні ефективно компенсувати різкі зміни погодних умов, що призводить до температурного стресу у рослин і зниження врожайності.

Однією з найскладніших проблем автоматизації теплиць є наявність сильних перехресних зв’язків між параметрами мікроклімату. Згідно з дослідженнями [2], температура та відносна вологість повітря є взаємозалежними змінними. Наприклад, активація системи вентиляції для зниження температури у спекотний день неминуче призводить до різкого падіння вологості, що може бути критичним для рослин, приклад зображено на рисунку 1.1. Використання незалежних контурів регулювання (наприклад, окремих ПІД-регуляторів для температури та вологості) у таких випадках є неефективним, оскільки керуючі впливи починають конфліктувати між собою, розхитуючи систему. Це вимагає застосування методів багатокритеріального керування, які розглядають мікроклімат як єдиний вектор станів [2].

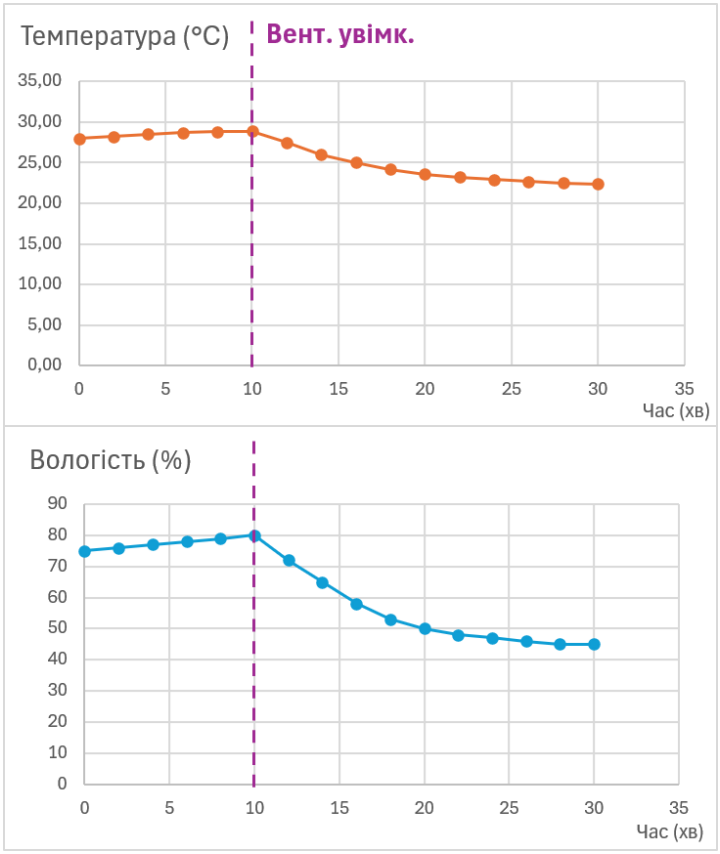


Рисунок 1.1 – Графік взаємозв’язку

Ще одним критичним фактором є енергоефективність та вплив стохастичних збурень (рис 1.2). У роботі [3] вказується, що теплиці є надзвичайно енергоємними об’єктами. Класичні системи керування працюють за реактивним принципом (реагують на відхилення, що вже сталося), що призводить до значних перевитрат енергії через інерційність систем опалення. Окрім того, на об’єкт постійно діють зовнішні збурення (сонячне випромінювання, швидкість вітру, зміна хмарності), які мають випадковий характер і важко піддаються точному моделюванню фізичними формулами. Саме тому автори пропонують переходити до стратегій керування на основі даних та методів прогнозуючого керування, які здатні враховувати невизначеність моделі та оптимізувати енерговитрати [3].

Реалізація такого упереджувального підходу вимагає застосування гнучких алгоритмів машинного навчання та нечіткої логіки для аналізу мікрокліматичних тенденцій.

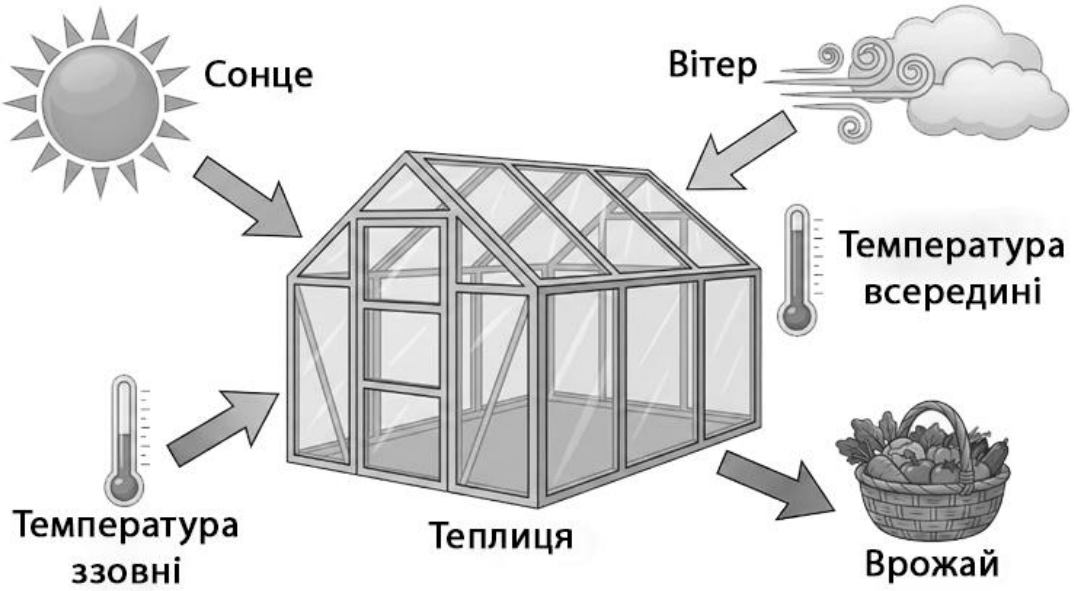


Рисунок 1.2 – Схема впливу збурень

1.2 Огляд методів штучного інтелекту в задачах агрономії

Впровадження технологій штучного інтелекту у сільське господарство є ключовим етапом переходу до концепції "Точного землеробства". На відміну від традиційних автоматизованих систем, які діють за жорсткими алгоритмами, системи на базі штучного інтелекту здатні аналізувати складні нелінійні залежності та адаптуватися до змінних умов навколишнього середовища (рис 1.3).

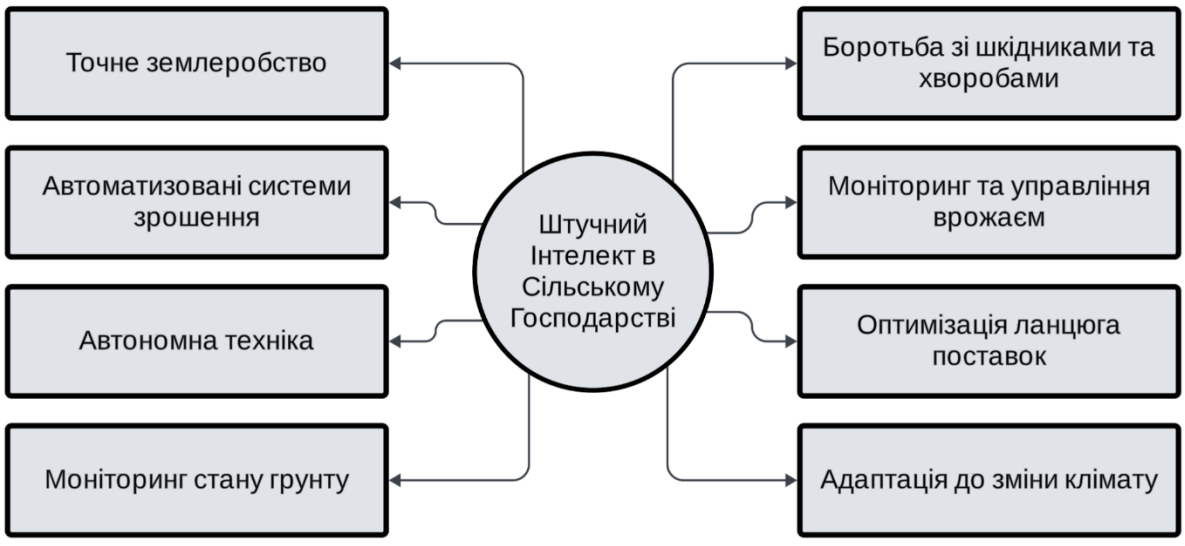


Рисунок 1.3 – Основні сфери застосування методів штучного інтелекту в агрономії

У оглядовому дослідженні [4, 60, 61, 62, 63] класифікує застосування інтелектуальних систем в агрономії за кількома ключовими напрямками: управління властивостями ґрунту, моніторинг стану врожаю, а також боротьба з бур’янами та захворюваннями рослин. Автор зазначає, що головною перевагою штучного інтелекту є здатність обробляти великі масиви даних, які людина не в змозі проаналізувати оперативно. Це дозволяє мінімізувати вплив людського фактору, знизити ризик помилок при прийнятті рішень та підвищити загальну продуктивність агроекосистеми. Зокрема, експертні системи та алгоритми машинного навчання дозволяють виявляти дефіцит поживних речовин або початок захворювання ще до появи видимих ознак, що є критичним для збереження врожаю [4].

Особливого значення набуває інтеграція методів штучного інтелекту з технологіями Інтернету речей, особливо в умовах тепличних господарств. Як зазначається у роботі [5], традиційні методи керування теплицями часто є неефективними через затримки у реакції на зміни мікроклімату та нераціональне використання ресурсів. Автори пропонують гібридний фреймворк, де IoT-пристрої виступають у ролі інструментів збору даних, а алгоритми машинного навчання – у ролі "мозку" системи, що приймає рішення, приклад такої архітектури зображено на рисунку 1.4.

Застосування такої концепції вимагає впровадження багаторівневої структури обробки даних, де хмарний сервіс бере на себе найбільш ресурсомісткі задачі предиктивної аналітики. Використання складних математичних моделей для аналізу часових рядів дозволяє системі ідентифікувати приховані закономірності, які неможливо відстежити за допомогою класичних алгоритмів. Це створює фундамент для переходу від реактивного реагування на поточні події до стратегічного планування режимів роботи всього комплексу, що значно підвищує його загальну ефективність.

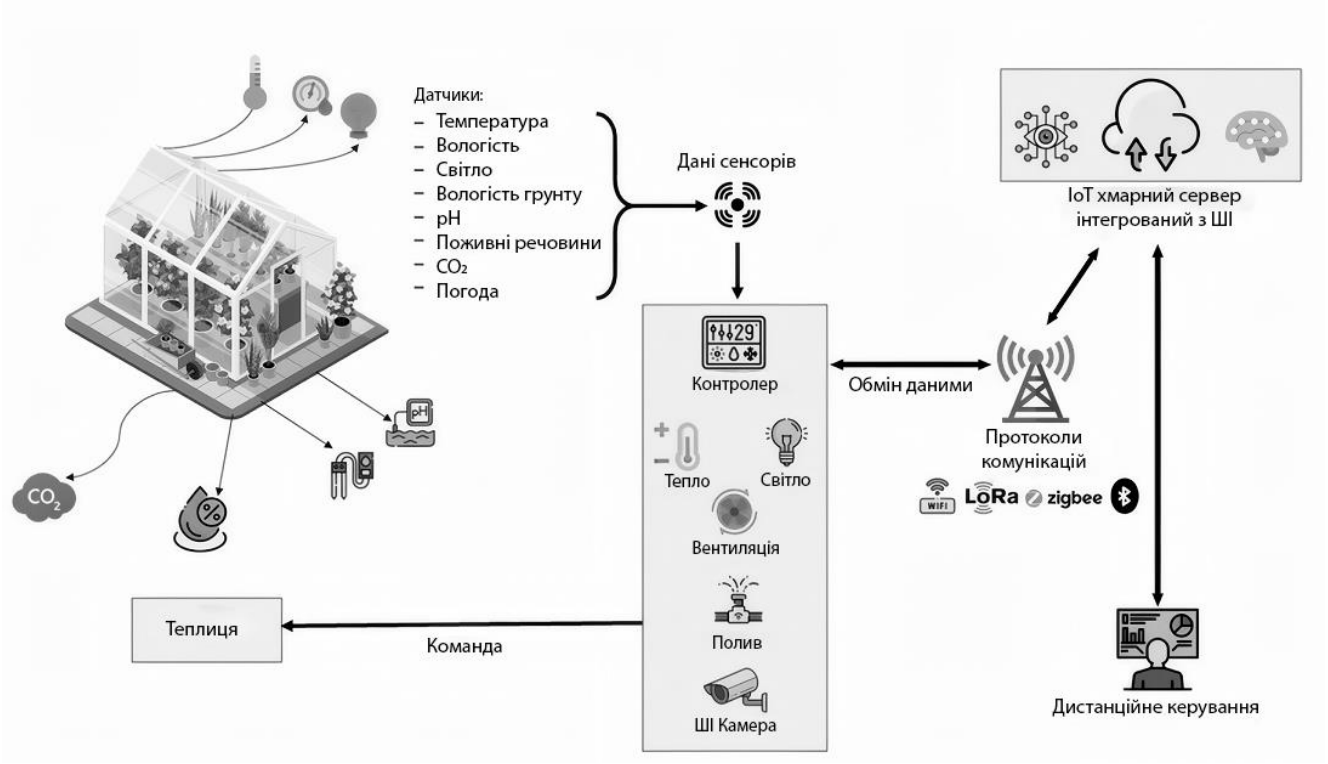


Рисунок 1.4 – Архітектура гібридної системи керування теплицею на базі IoT та машинного навчання

Згідно з дослідженнями [5], така гібридна архітектура дозволяє реалізувати прогнозуюче керування: система не просто реагує на поточні значення датчиків, а прогнозує потреби рослин, використовуючи історичні дані. Це дозволяє оптимізувати роботу виконавчих механізмів (насосів, вентиляторів, обігрівачів), забезпечуючи сталий розвиток та значну економію енергоресурсів.

1.2.1 Застосування нечіткої логіки для прийняття рішень

Одним із найбільш ефективних підходів до керування складними нелінійними об’єктами, до яких належить теплиця, є використання апарату нечіткої логіки (Fuzzy Logic). На відміну від класичної булевої логіки, яка оперує лише двома станами ("0" або "1", "Вимкнено" або "Увімкнено"), нечітка логіка дозволяє оперувати лінгвістичними змінними, наближаючи алгоритм керування до людського мислення.

У роботі [6] детально розглядають архітектуру контролера мікроклімату на базі IoT та нечіткої логіки. Автори зазначають, що процес керування складається з трьох етапів: фазифікації (перетворення чітких значень датчиків у нечіткі змінні), логічного виведення на основі бази правил та дефазифікації (отримання чіткого керуючого сигналу). Ключовим елементом такої системи є функції належності, які визначають ступінь приналежності фізичної величини до певного терму, наприклад, "Холодно", "Нормально" або "Жарко".

Графічна інтерпретація функцій належності, що використовуються для класифікації температурних режимів, наведена на рисунку 1.5.

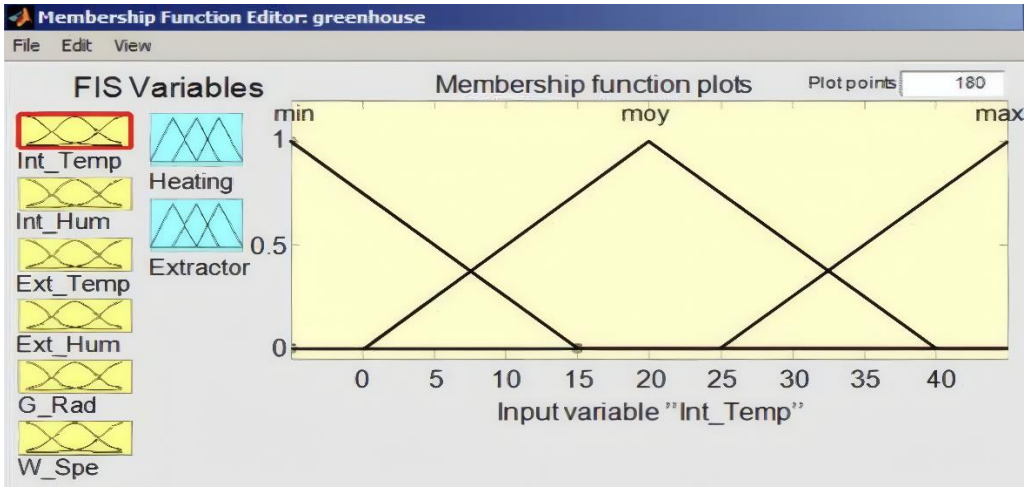


Рисунок 1.5 – Приклад трикутних функцій належності для вхідної змінної «Температура» у середовищі Matlab [6]

Як видно з графіка, використання перекриття між трикутниками дозволяє системі уникати різких стрибків при керуванні. Замість того, щоб просто вимкнути обігрів при досягненні порогового значення (що призводить до частих вмикань/вимикань реле), фазі-контролер плавно зменшує потужність, стабілізуючи мікроклімат.

Ефективність такого підходу порівняно з класичними ПІД-регуляторами підтверджується у дослідженні [7]. Автори провели порівняльний аналіз роботи обох типів регуляторів в умовах термостабілізації інкубатора, фізика процесів якого схожа з теплицею. Результати експерименту показали, що класичний ПІД-

регулятор схильний до значного перерегулювання – ситуації, коли температура за інерцією перевищує задану норму, що може бути критичним для біологічних об’єктів (рис 1.6).

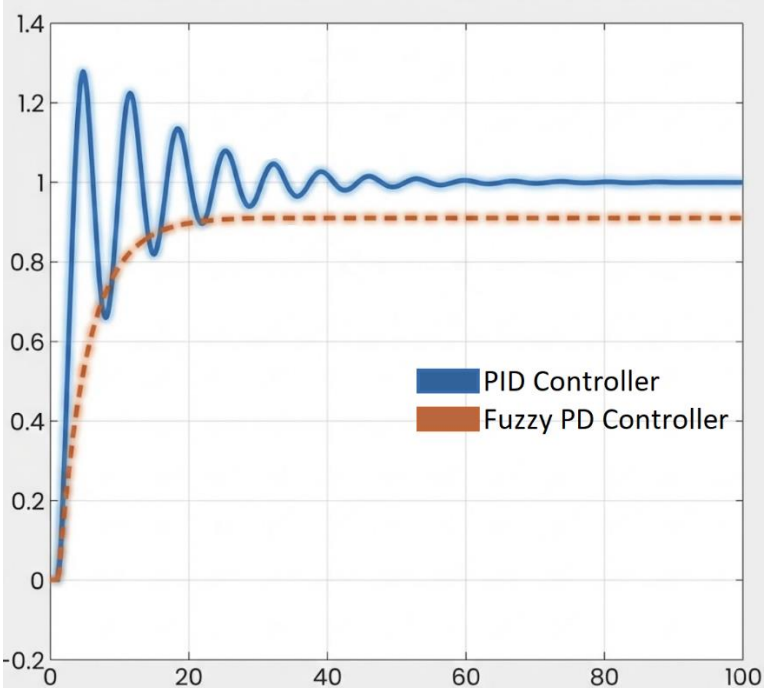


Рисунок 1.6 – Порівняння характеру перехідних процесів при використанні ПІД-регулятора та нечіткого контролера (узагальнено)

Натомість, як стверджується в дослідженні [7], система на базі нечіткої логіки продемонструвала кращу динаміку перехідних процесів: вихід на заданий режим відбувався плавно, без перегріву та з меншою кількістю коливань. Це дозволяє зробити висновок про доцільність використання інтелектуальних методів керування для забезпечення стабільності біотехнічної системи в умовах невизначеності.

1.2.2 Використання комп’ютерного зору для діагностики стану рослин

Традиційні системи керування мікрокліматом спираються виключно на показники датчиків навколишнього середовища (температури, вологості, освітлення). Проте, такий підхід є опосередкованим: система стабілізує параметри

повітря, але не отримує інформації про те, як на ці параметри реагує сама рослина. Інтеграція технологій комп’ютерного зору дозволяє створити біотехнічний контур зворотного зв’язку, де керуючий вплив формується на основі фізіологічного стану культури.

У огляді [8] детально проаналізовано методи виявлення та класифікації захворювань рослин за допомогою штучного інтелекту. Автори наголошують, що своєчасна діагностика хвороб на ранніх стадіях є критичною для запобігання втратам врожаю. Типовий алгоритм обробки зображень у таких системах складається з чотирьох етапів: отримання зображення, попередньої обробки, сегментації (виділення області листка) та класифікації ознак (рис 1.7).

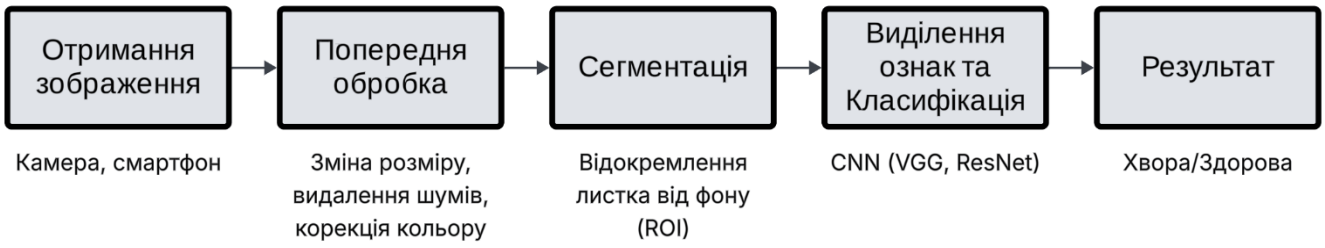


Рисунок 1.7 – Узагальнена структурна схема алгоритму діагностики захворювань рослин

Згідно з дослідженням [8], найбільш ефективним інструментом для задач класифікації є згорткові нейронні мережі (рис 1.8). На відміну від класичних методів обробки зображень, де ознаки (колір, текстура) виділяються вручну, CNN здатні автоматично формувати ієрархію ознак, що забезпечує високу точність діагностики (понад 95% для популярних архітектур типу ResNet або VGG).

Використання таких архітектур у тепличних комплексах дозволяє створювати надійні системи раннього попередження, здатні в режимі реального часу ідентифікувати візуальні маркери патологій. Інтеграція модулів комп'ютерного зору для безперервного моніторингу суттєво мінімізує вплив людського фактора та забезпечує можливість оперативного реагування на біологічні загрози ще до їх масового поширення.

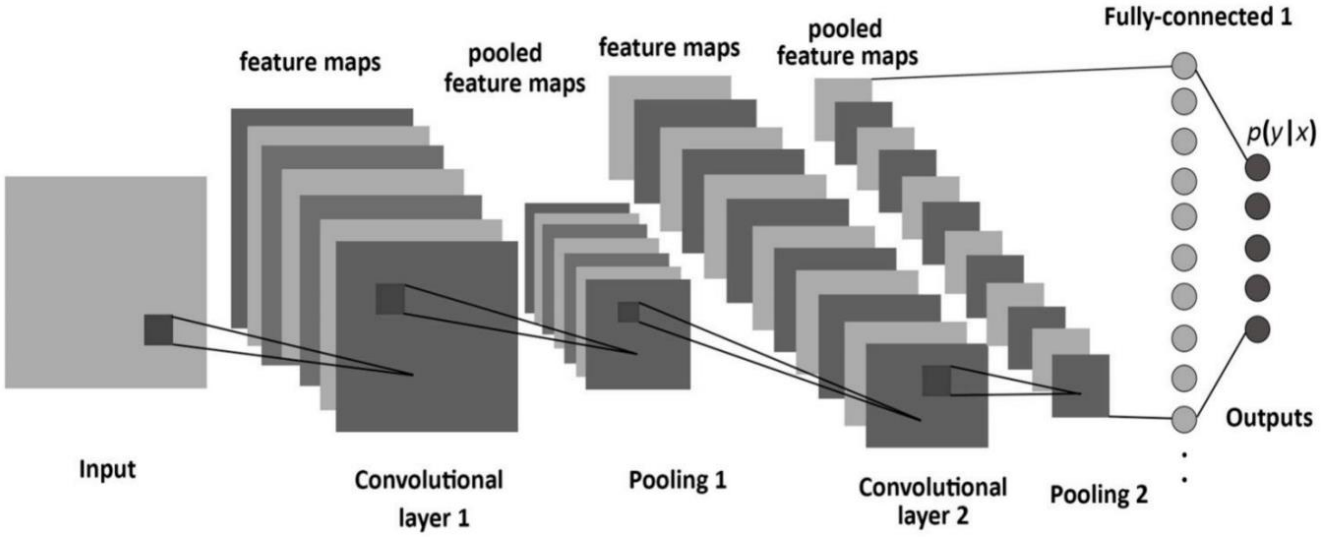


Рисунок 1.8 – Типова архітектура згорткової нейронної мережі для задач фенотипування рослин [9]

Однак, для задач керування мікрокліматом важливо не лише виявляти хвороби, а й оцінювати динаміку розвитку рослин – фенотипування. У дослідженні [9] розглядається застосування глибокого навчання для високопродуктивного фенотипування рослин. Автори виділяють такі ключові задачі CV в агрономії: підрахунок листя, вимірювання площі листкової поверхні, оцінка біомаси та виявлення стресових станів (наприклад, в’янення через нестачу води або опіки через надмірне освітлення).

Використання комп’ютерного зору як сенсора зворотного зв’язку дозволяє реалізувати адаптивні стратегії керування. Наприклад, якщо система CV фіксує в’янення листя, контролер може автоматично активувати систему поливу або затінення, навіть якщо датчики ґрунту ще не показують критичних значень. Такий підхід перетворює теплицю з простої автоматизованої системи на кіберфізичну систему, орієнтовану на біологічний об’єкт.

1.3 Класифікація методів математичного моделювання мікроклімату

Розробка ефективної системи керування мікрокліматом теплиці неможлива без наявності математичної моделі, яка описує динаміку зміни параметрів

середовища. Модель дозволяє прогнозувати реакцію системи на керуючі впливи та зовнішні збурення, що є основою для алгоритмів оптимізації.

У огляді [11, 64, 65, 66] наводиться загальноприйнята класифікація моделей динаміки теплиць, які поділяються на дві основні категорії:

1. Фізичні моделі. Ці моделі базуються на фундаментальних законах термодинаміки, балансі енергії та маси. Найбільш точним, але й найбільш обчислювально складним підходом тут є обчислювальна гідродинаміка. CFD дозволяє моделювати неоднорідність розподілу температурних полів у 3D-просторі, враховуючи конструкцію теплиці.

2. Моделі «чорної скриньки». Цей підхід не вимагає детального знання фізичних параметрів об’єкта (товщини стінок, коефіцієнтів теплопровідності матеріалів). Натомість, модель будується шляхом виявлення статистичних закономірностей між вхідними даними (керуючі сигнали, погода) та виходом системи (температура, вологість) на основі історичних даних.

Класифікацію методів математичного моделювання мікроклімату теплиць зображено на рисунку 1.9.

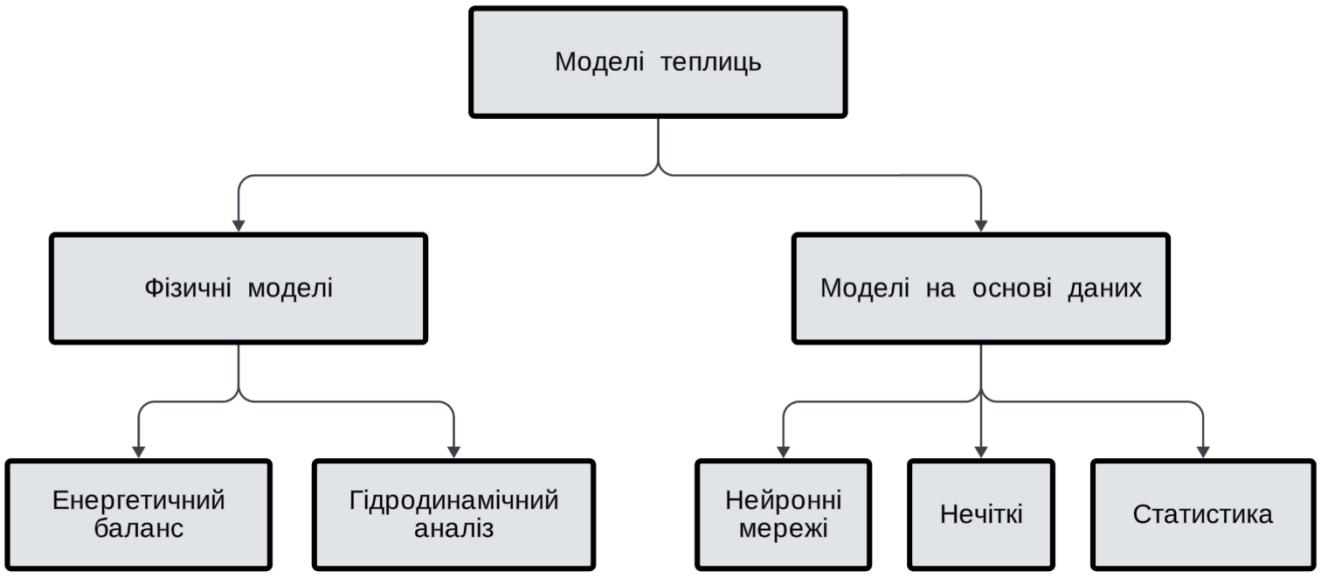


Рисунок 1.9 – Класифікація методів математичного моделювання мікроклімату теплиць

Як зазначається у роботі [10], хоча фізичні моделі забезпечують високу точність розуміння процесів, вони часто є надто громіздкими для використання в системах реального часу. Наявність великої кількості емпіричних коефіцієнтів, які важко виміряти точно, робить їх калібрування складним завданням.

Водночас, автори [10] вказують на зростаючу популярність методів штучного інтелекту та гібридних моделей. Інтеграція нейронних мереж з алгоритмами оптимізації дозволяє створювати адаптивні системи, які навчаються в процесі роботи. Це особливо актуально для задач енергозбереження, де необхідно знаходити баланс між витратами енергії та комфортом рослин в умовах невизначеності.

1.3.1 Методи математичного моделювання та прогнозування

Статистичний підхід до прогнозування мікроклімату базується на аналізі історичних даних як часових рядів, де майбутнє значення змінної розглядається як функція від її попередніх значень та помилок моделі. Серед лінійних стаціонарних моделей найбільш поширеним стандартом є метод ARIMA (AutoRegressive Integrated Moving Average) – авторегресійна інтегрована модель змінного середнього.

У дослідженні [12] детально розглянуто застосування гібридних підходів на основі ARIMA для прогнозування температури та вологості в теплиці. Автори пояснюють, що модель ARIMA(p, d, q) складається з трьох компонентів:

1. AR (AutoRegressive) вказує на залежність поточного значення від попередніх (p – порядок авторегресії).
2. I (Integrated) відповідає за приведення ряду до стаціонарного вигляду шляхом взяття різниць (d – порядок інтегрування).
3. MA (Moving Average) враховує залежність від попередніх помилок прогнозу (q – порядок ковзного середнього).

Структурну схему моделі прогнозу ARIMA зображено на рисунку 1.10.

Рисунок 1.10 – Структурна схема прогнозу в моделі ARIMA

Згідно з результатами [12], ARIMA демонструє високу точність на короткострокових інтервалах, коли зовнішні умови змінюються плавно. Однак, головним обмеженням методу є його лінійність. Оскільки мікроклімат теплиці є суттєво нелінійною системою, класична ARIMA може давати значні похибки в моменти пікових навантажень.

Проблематику використання часових рядів також досліджено в роботі [13], де проводилося порівняння різних моделей (ARIMA, Prophet, LSTM) для прогнозування концентрації CO_2 та інших параметрів середовища. Автори зазначають, що статистичні моделі вимагають ретельної попередньої обробки даних – перевірки на стаціонарність та сезонність. Хоча ARIMA показала прийнятні результати для загальних тенденцій, вона поступалася методам глибокого навчання у здатності адаптуватися до складних патернів даних.

1.3.2 Нейромережеві методи прогнозування

Зважаючи на обмеження статистичних методів при роботі з високонелінійними даними, сучасні системи керування все частіше інтегрують методи глибокого навчання. Особливу увагу дослідників привертають рекурентні

нейронні мережі, які, на відміну від мереж прямого поширення, мають зворотні зв’язки.

У дослідженні [15] розглядається застосування глибокого навчання для прогнозування агрономічних показників. Автори зазначають, що хоча звичайні RNN ефективно виявляють короткострокові залежності, при навчанні на довгих часових рядах вони страждають від проблеми "зникаючого градієнта". Це призводить до того, що мережа втрачає здатність враховувати кліматичні події, що відбулися значний час тому.

Для усунення цього недоліку найбільш доцільним є використання архітектури LSTM (Long Short-Term Memory). Як детально описано в роботі [14], LSTM вирішує проблему градієнта завдяки унікальній структурі комірки пам’яті. На відміну від простого нейрона RNN, комірка LSTM містить три регулюючі механізми (вентилі):

1. Вентиль забування (Forget Gate) відфільтровує застарілу інформацію (наприклад, нічні температури при настанні дня).
2. Вхідний вентиль (Input Gate) оновлює стан пам’яті новими даними з датчиків.
3. Вихідний вентиль (Output Gate) формує прогноз на наступний крок.

Структурна схема комірки LSTM, наведена у джерелі [14], представлена на рисунку 1.11.

Завдяки такій складній внутрішній організації комірок, модель набуває здатності відслідковувати та запам'ятовувати довгострокові часові залежності. Для тепличного комплексу це означає можливість враховувати глибоку теплову інерційність об'єкта: наприклад, алгоритм здатний математично запам’ятовувати обсяг сонячної енергії, акумульованої масивом ґрунту протягом усього дня, щоб максимально точно спрогнозувати швидкість його охолодження вночі.

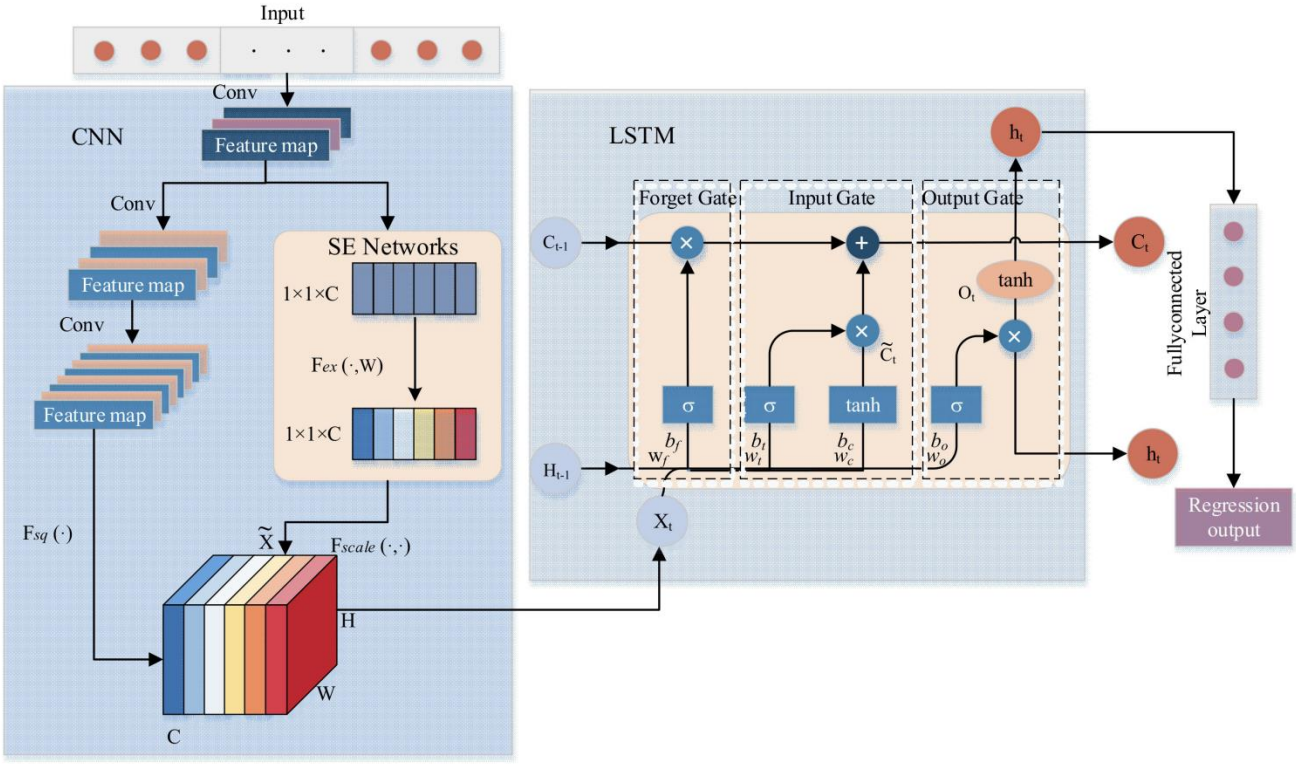


Рисунок 1.11 – Внутрішня архітектура блоку LSTM [14]

Ефективність застосування LSTM для прогнозування мікроклімату підтверджується експериментальними даними. У дослідженні [14] автори інтегрували LSTM з алгоритмом пошуку горобця для оптимізації гіперпараметрів мережі. Результати моделювання показали, що такий підхід дозволяє досягти високої точності прогнозування навіть в умовах різких змін зовнішнього середовища.

На рисунку 1.12 наведено порівняння реальних значень температури та прогнозу, згенерованого LSTM-моделлю. Видно, що крива прогнозу майже ідеально відтворює динаміку реального процесу.

Така висока точність наближення значень має критичне значення для розробки енергоефективних систем керування. Завдяки мінімізації похибки між очікуваними та фактичними показниками, локальний контролер отримує змогу формувати максимально плавні впливи на виконавчі механізми.

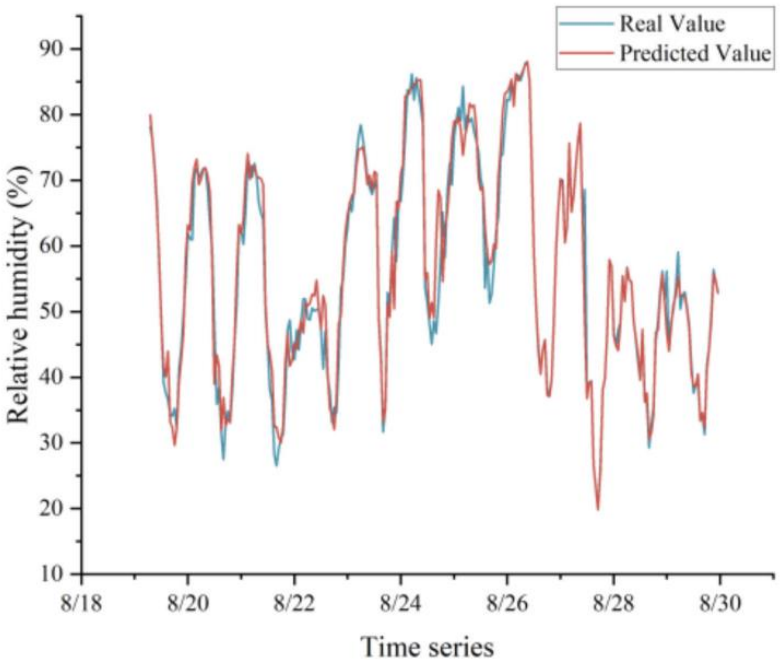


Рисунок 1.12 – Порівняння результатів прогнозування мікроклімату моделлю LSTM з реальними даними [14]

1.4 Алгоритми багатокритеріальної оптимізації режимів роботи

Інтеграція підсистем прогнозування у контур керування мікрокліматом відкриває шлях до переходу від класичного автоматичного регулювання до оптимального керування. Якщо задача звичайного регулятора полягає у підтримці заданої уставки (наприклад, стабільно 22°C), то задача системи оптимізації – динамічно змінювати цю уставку так, щоб мінімізувати витрати ресурсів, не шкодячи при цьому врожаю.

У дослідженні [16, 67, 68, 69, 70] наголошується, що сучасні «розумні» теплиці стикаються з проблемою конфлікту цілей. З одного боку, максимізація фотосинтезу та росту рослин вимагає підтримання ідеальних, часто дуже енергоємних умов, а з іншого – стрімке зростання цін на енергоносії робить критично важливою мінімізацію експлуатаційних витрат. Автори визначають вирішення цього конфлікту як задачу багатокритеріальної оптимізації, що математично зводиться до пошуку мінімуму комплексної цільової функції. Ця функція одночасно інтегрує декілька компонентів із різними ваговими

коефіцієнтами: енергетичну складову, яка відображає фінансову вартість ресурсів, витрачених на опалення, вентиляцію та освітлення; біологічну складову, що діє як математичний штраф за будь-яке відхилення параметрів мікроклімату від оптимальної зони комфорту рослини; а також екологічний аспект, орієнтований на жорстку мінімізацію викидів вуглекислого газу та раціональне використання водних ресурсів.

Згідно з висновками [16], традиційні методи керування не здатні ефективно вирішувати такі задачі в реальному часі через складну нелінійність теплових процесів у теплиці. Тому сучасний вектор досліджень спрямований на використання евристичних алгоритмів пошуку (таких як генетичні алгоритми) та методів керування з прогнозуючою моделлю, які здатні знаходити баланс між економією та продуктивністю. Дворівневу архітектуру керування теплицею зображено на рисунку 1.13.

Рисунок 1.13 – Дворівнева архітектура керування теплицею

1.4.1 Генетичні алгоритми та їх застосування в енергоменеджменті

Для вирішення задач глобальної оптимізації, де цільова функція є багатовимірною та має складну топологію з багатьма локальними екстремумами, широко застосовуються еволюційні методи, зокрема генетичні алгоритми. Ці алгоритми імітують процес природного відбору, оперуючи популяцією потенційних рішень, які еволюціонують через операції селекції, перехрещування та мутації.

У роботі [17] запропоновано систему інтелектуального керування сільськогосподарською теплицею на базі адаптивного покращеного генетичного алгоритму. Автори розглядають задачу оптимізації роботи мультиенергетичної системи, яка включає відновлювані джерела енергії. Метою оптимізації є мінімізація економічних витрат при забезпеченні заданих параметрів мікроклімату.

Класичний генетичний алгоритм часто стикається з проблемою "передчасної збіжності", коли пошук застряє в локальному оптимумі, не досягаючи найкращого глобального рішення. Для вирішення цієї проблеми в дослідженні [17] було розроблено адаптивний механізм: ймовірності перехрещування (P_c) та мутації (P_m) не є фіксованими константами, а динамічно змінюються залежно від пристосованості конкретної особини. У разі зупинки покращення, механізм автоматично підвищує коефіцієнт мутації для переходу від уточнення поточної області до пошуку нових областей простору станів.

Загальна логіка роботи алгоритму, реалізованого авторами для керування виконавчими механізмами теплиці, представлена на блок-схемі (рис. 1.14).

Як видно з наведеної блок-схеми, адаптивний алгоритм безперервно генерує та оптимізує керуючі впливи для виконавчих механізмів. Це дозволяє системі в режимі реального часу знаходити найкращий компроміс між суворим дотриманням заданих агротехнічних параметрів мікроклімату та мінімізацією загального енергоспоживання об'єкта.

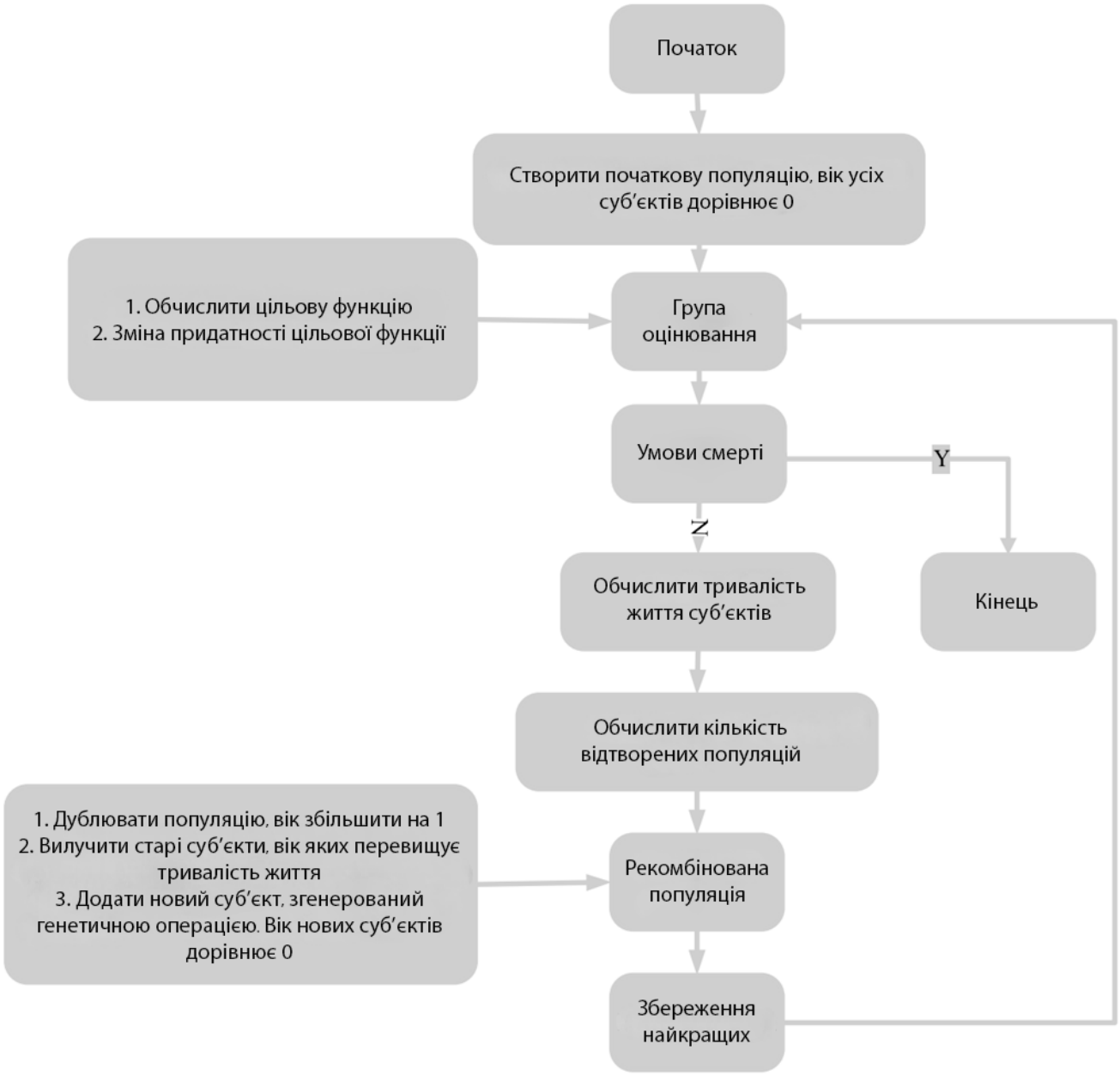


Рисунок 1.14 – Блок-схема роботи адаптивного генетичного алгоритму оптимізації [17]

Ефективність запропонованого підходу підтверджується експериментальними даними. На рисунку 1.15 наведено графіки зміни температури повітря та відносної вологості в теплиці під час роботи інтелектуальної системи керування.

Як видно з графіка, система забезпечує стабільну підтримку параметрів мікроклімату в заданих діапазонах, незважаючи на зовнішні збурення. Коливання температури мінімальні, що свідчить про те, що генетичний алгоритм успішно

підбирає оптимальні керуючі впливи, забезпечуючи комфортні умови для рослин при раціональному використанні енергоресурсів.

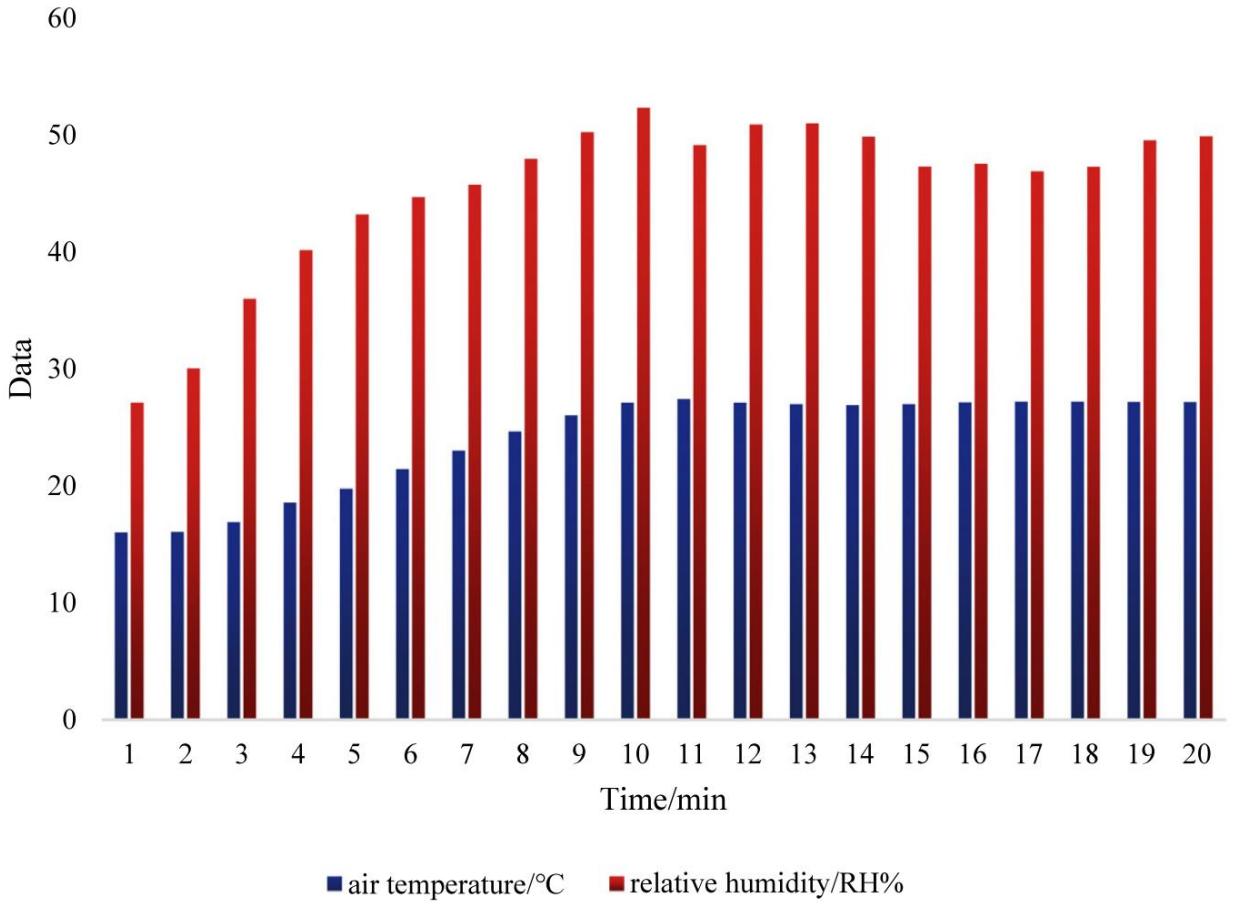


Рисунок 1.15 – Експериментальні дані зміни температури повітря та відносної вологості при керуванні на базі генетичного алгоритму [17]

1.4.2 Метод прогнозуючого керування

Серед сучасних стратегій оптимального керування найбільш перспективним для тепличних комплексів вважається метод керування з прогнозуючою моделлю (MPC). На відміну від класичних ПІД-регуляторів, MPC використовує математичну модель об’єкта для передбачення його поведінки на майбутньому горизонті планування.

Однак, класичний MPC чутливий до неточностей моделі та зовнішніх збурень. Тому в сучасних наукових працях, зокрема у дослідженні [3], пропонується використовувати робастне керування (RMPC). Цей підхід враховує

невизначеність параметрів (наприклад, похибки прогнозу погоди) і гарантує, що температура залишиться в допустимих межах навіть за найгіршого сценарію розвитку подій.

Структурна схема системи RMPC, запропонована авторами, наведена на рисунку 1.16. Вона включає такі ключові елементи:

1. Модель невизначеності. Враховує можливі відхилення зовнішніх факторів (сонячного випромінювання, температури ззовні).

2. Оптимізатор. Вирішує задачу оптимізації, знаходячи керуючі впливи, які мінімізують енерговитрати, одночасно задовольняючи жорсткі обмеження на мікроклімат.

3. Модель прогнозування. Базується на нейронній мережі, що дозволяє адаптувати систему до конкретної конструкції теплиці без складних фізичних розрахунків.

Рисунок 1.16 – Структурна схема системи робастного керування з прогнозуючою моделлю [3]

1.5 Архітектури IoT-систем

Впровадження інтелектуальних алгоритмів керування (таких як нейромережі LSTM та CV) висуває високі вимоги до обчислювальних потужностей системи. У сучасній інженерії Інтернету речей існує дві основні парадигми організації обчислень: хмарні обчислення та граничні обчислення.

У огляді [18, 71, 72, 73] детально проаналізовано застосування цих архітектур у «розумному» сільському господарстві. Автори зазначають, що традиційна хмарна архітектура, за якої всі дані з датчиків передаються на віддалений сервер, забезпечує суттєві переваги у вигляді необмеженої обчислювальної потужності для зберігання великих масивів даних та навчання складних моделей. Однак, при впровадженні у системи реального часу виникає низка критичних недоліків. Насамперед, час передачі сигналу до хмарного середовища та у зворотному напрямку створює мережеву затримку, яка може бути неприпустимо великою для оперативного реагування на аварійні ситуації. Додатковою вразливістю є абсолютна залежність від стабільності каналу зв'язку, оскільки будь-яка втрата інтернет-з’єднання призводить до повної зупинки контуру інтелектуального керування. Крім того, гостро постають питання інформаційної безпеки та конфіденційності, адже безперервна трансляція чутливої інформації, такої як відеопотоки з камер, на віддалені сервери створює потенційні ризики витоку даних.

На противагу цьому, граничні обчислення передбачають перенесення обчислень безпосередньо на пристрої, розташовані в теплиці. Це дозволяє приймати рішення миттєво, навіть за відсутності інтернету.

Оптимальним рішенням для сучасних теплиць, згідно з висновками [18], є гібридна трирівнева архітектура (Cloud-Fog-Edge), структура якої наведена на рисунку 1.17.

У такій системі відбувається чіткий розподіл обчислювального навантаження: граничний рівень забезпечує миттєве керування обладнанням, проміжні шлюзи здійснюють попередню агрегацію даних, а хмарний сервер виконує ресурсомісткі задачі машинного навчання. Такий синергетичний підхід

дозволяє досягти оптимального балансу між безперебійною швидкодією локальних контурів та глобальною інтелектуальною аналітикою.

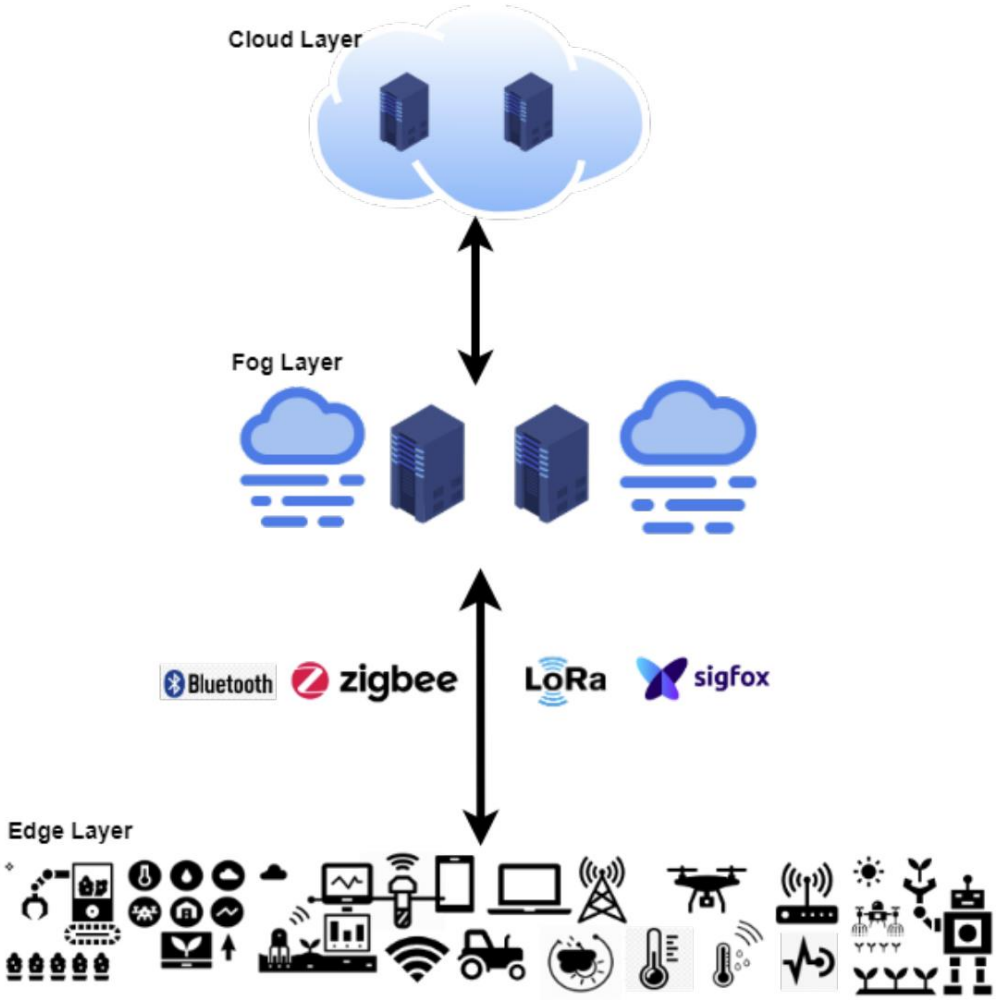


Рисунок 1.17 – Трирівнева архітектура IoT-системи: Cloud, Fog та Edge Computing [18]

Практичну реалізацію такої багаторівневої моделі розглянуто в роботі [19] на прикладі системи EECSI (Energy Efficient Clustered Smart Irrigation). Автори пропонують структуру, що складається з трьох шарів:

1. Фізичний рівень. Сенсори та виконавчі механізми.
2. Мережевий/Граничний рівень. Локальні контролери, які здійснюють первинну обробку даних та локальне керування.
3. Прикладний рівень. Хмарний сервер для глобальної аналітики та візуалізації.

Такий підхід дозволяє досягти балансу: критично важливі алгоритми (наприклад, підтримка температури) виконуються на Edge-рівні (локально), а навчання нейромереж та довгострокова аналітика – у Хмарі. Це забезпечує високу надійність системи та енергоефективність передачі даних.

1.6 Аналіз протоколів передачі даних в системах IoT

Ефективність функціонування розподілених систем керування мікрокліматом на базі Cloud-Edge архітектури критично залежить від обраної моделі комунікації між граничним вузлом та хмарним сервером. Збільшення обсягів сенсорних даних та необхідність передачі керуючих команд у режимі реального часу вимагають застосування оптимізованих протоколів прикладного рівня [34]. У сучасних IoT-системах найчастіше використовуються такі протоколи: HTTP/REST, CoAP, WebSocket та MQTT [30, 33].

Класичний протокол HTTP, який базується на архітектурі REST, тривалий час залишався стандартом для веб-сервісів. Проте його використання в IoT-системах, особливо на базі мікроконтролерів, супроводжується значними накладними витратами через великий розмір HTTP-заголовків та синхронну модель "запит-відповідь". Для вирішення цієї проблеми було розроблено протокол CoAP (Constrained Application Protocol), який функціонує поверх UDP і забезпечує низьке енергоспоживання [32]. Однак використання UDP знижує надійність доставки пакетів у нестабільних бездротових мережах, що є критичним недоліком для систем автоматичного керування актуаторами.

Для забезпечення двостороннього зв’язку без постійного розриву з’єднання часто застосовується протокол WebSocket. У дослідженні [31] проаналізовано масштабованість WebSocket у хмарному середовищі Amazon Web Services. Хоча цей протокол забезпечує мінімальну затримку після встановлення з’єднання, він вимагає постійного утримання відкритого TCP-сокету, що швидко виснажує оперативну пам’ять малопотужних мікроконтролерів при великій кількості клієнтів.

Найбільш адаптованим до вимог сучасних індустріальних та аграрних IoT-мереж є протокол MQTT. Цей легковаговий протокол працює за моделлю "публікація-підписка" через центральний брокер повідомлень [33, 74, 75, 76]. У дослідженні [29] проведено порівняльний аналіз продуктивності MQTT та WebSocket для потокової передачі даних у реальному часі. Результати експериментів довели, що MQTT демонструє значно меншу затримку, нижчий рівень джитера та вищу стійкість до втрати пакетів порівняно з WebSocket, особливо в умовах обмежених апаратних ресурсів.

Крім того, аналіз продуктивності різних IoT-протоколів, проведений у роботі [32], підтверджує, що MQTT забезпечує найкращий баланс між надійністю доставки та обсягом переданих даних. Структура пакета MQTT має мінімальний заголовок (від 2 байт), що ідеально підходить для передачі невеликих масивів даних, таких як показники температури та вологості у форматі JSON.

Враховуючи проаналізовані характеристики, для розроблюваної системи керування мікрокліматом теплиці найбільш доцільним є використання протоколу MQTT. Його застосування у синергії з платформою ESP32 дозволить забезпечити надійну, асинхронну та енергоефективну передачу телеметрії до хмарного сервера для подальшої обробки моделлю ARIMA, а також миттєву доставку керуючих сигналів назад на граничний вузол.

1.7 Аналіз методів машинного навчання та статистичного моделювання для прогнозування часових рядів

Реалізація прогнозуючого керування мікрокліматом вимагає використання надійних математичних апаратів для прогнозування часових рядів. Сучасний інструментарій включає класичні статистичні моделі (ARIMA, SARIMA), адитивні моделі машинного навчання (Prophet) та архітектури глибинного, зокрема рекурентні нейронні мережі LSTM [36].

Глибинні нейронні мережі, такі як LSTM, демонструють високу ефективність у моделюванні складних нелінійних залежностей у довготривалих періодах.

Наприклад, у дослідженнях [38, 40] доведено перевагу архітектури LSTM для прогнозування температурних режимів у великих містах (зокрема, на прикладі метеоданих Києва). Високу точність гібридних моделей на базі LSTM для динамічного прогнозування навантажень також підтверджується у роботі [42]. Однак, використання нейромереж має суттєві обмеження для IoT-систем: вони вимагають колосальних масивів історичних даних для початкового тренування, потребують значних обчислювальних ресурсів для перенавчання та є непрозорою, що знижує інтерпретованість результатів системою автоматики.

Популярною альтернативою складним нейромережам є модель Prophet, розроблена дослідниками з Meta. Ця адитивна модель оптимізована для роботи з часовими рядами, що мають яскраво виражену сезонність. У роботі [35] успішно застосували Prophet для прогнозування сезонних коливань температур, а у роботі [37] використали її у синергії із SARIMA для прогнозування метеорологічних посух. Згідно з результатами порівняльного аналізу [39], Prophet часто перевершує статистичні методи у макро-задачах прогнозування (наприклад, енергоспоживання) завдяки високій стійкості до пропущених даних. Разом з тим, для мікрорівня – прогнозування температури у закритому середовищі теплиці, де динаміка залежить від миттєвого включення актуаторів, а не від річної сезонності – Prophet може реагувати на короткочасні флуктуації гірше за авторегресійні алгоритми.

Класичні моделі інтегрованої авторегресії ковзного середнього (ARIMA/SARIMA) досі залишаються фундаментальним стандартом у наукових дослідженнях. У роботі [41] наголошують на критичній важливості сезонно-скоригованих авторегресійних моделей для побудови надійних систем раннього попередження про теплові хвилі. Як зазначається у огляді [36], незважаючи на бурхливий розвиток нейромереж, моделі сімейства ARIMA/SARIMA продовжують забезпечувати найкращий баланс між обчислювальною складністю, прозорістю математичного апарату та точністю на коротких і середніх горизонтах прогнозування.

Враховуючи вимоги до розроблюваної Cloud-Edge архітектури – необхідність швидкого та циклічного перенавчання моделі безпосередньо на хмарному сервері з використанням обмежених вибірок свіжої телеметрії – використання ресурсоємних архітектур типу LSTM є надлишковим та економічно недоцільним [40, 42]. Вибір моделі ARIMA для серверної частини комплексу є оптимальним інженерним рішенням, оскільки вона гарантує високу точність короткострокового прогнозу при мінімальних затримках на обчислення, забезпечуючи ефективну роботу локального нечіткого регулятора.

1.8 Агрономічні вимоги до мікроклімату тепличних культур та біологічне обгрунтування інтелектуального керування

Формування та точна підтримка оптимального мікроклімату є критичним фактором для забезпечення стабільної врожайності в універсальних тепличних комплексах. Як зазначають у роботі [44], динаміка тепличного середовища безпосередньо впливає на процеси евапотранспірації – сумарного випаровування вологи ґрунтом та випаровування води самими рослинами. Підтримка правильного балансу температури та вологості є вкрай складною нелінійною задачею, особливо в умовах мінливих зовнішніх кліматичних зон, де впровадження спеціалізованих Smart-IoT платформ стає біологічною та економічною необхідністю [48].

Відхилення температурного режиму від біологічної норми викликає у тепличних культур сильний фізіологічний стрес. У дослідженні [46] доводять, що тепловий стрес у поєднанні з порушенням балансу вологості пригнічує механізми захисту рослин, що призводить до передчасного закриття продихів, зупинки фотосинтезу та деградації білкових структур. На прикладі томатів – однієї з найбільш поширених та чутливих до мікроклімату тепличних культур, у роботі [45] експериментально показали, що навіть короткочасний вплив екстремально високих температур (понад 32–35 °C) критично знижує фертильність пилку та загальну зав’язуваність плодів, що веде до незворотних втрат врожаю.

Окрім абсолютних значень температури, вкрай небезпечними для рослин є різкі температурні коливання та недостатнє опромінення сонцем. Наприклад, некоректне або занадто різке керування виконавчими механізмами, такими як термальні екрани чи системи вентиляції, може спричинити раптові перепади мікрокліматичних параметрів, що порушує ростові процеси та знижує загальну ефективність фотосинтезу [43]. Саме тому сучасна агрономія відмовляється від грубого керування і вимагає переходу до систем безперервного моніторингу. Дослідження [47] підтверджує, що імплементація IoT-сенсорів для мікрокліматичного моніторингу в режимі реального часу дозволяє не лише оптимізувати витрати води на полив, але й уникнути виникнення стресових станів у культур завдяки своєчасному коригуванню параметрів середовища.

Проведений агрономічний аналіз виявляє головний недолік класичних релейних систем керування: при досягненні температурних порогів вони різко вмикають або вимикають потужні актуатори. Це створює періодичні температурні коливання з постійними різкими перепадами, що тримає рослини у стані постійного стресу [46].

Використання запропонованого у даній роботі локального регулятора на базі нечіткої логіки усуває цю проблему на фундаментальному рівні. Завдяки плавній широтно-імпульсній модуляції потужності виконавчих механізмів, розроблена система здатна асимптотично наближатися до ідеальної заданої температури без ефекту перерегулювання. Це захищає культури від термічного шоку, забезпечує оптимальні умови для евапотранспірації [44] та дозволяє максимально розкрити генетичний потенціал врожайності рослин.

1.9 Проблематика енергоефективності та апаратного енергозбереження в IoT-Edge системах

Широкомасштабне впровадження бездротових сенсорних мереж та розподілених IoT-систем у сільському господарстві висуває жорсткі вимоги до їхньої енергоефективності. Забезпечення безперебійної роботи граничних вузлів

ускладнюється необхідністю паралельного обслуговування енергоємних виконавчих механізмів, датчиків та модулів бездротового зв'язку. Дослідження [52], присвячене вимірюванню енергоспоживання базових WSN на платформі Arduino, демонструє, що класичні 8-бітні архітектури мають низький коефіцієнт корисної дії при постійному опитуванні датчиків і є вразливими до просадок напруги при роботі з індуктивними навантаженнями.

Перехід до потужніших обчислювальних платформ, таких як ESP32, дозволяє не лише реалізувати складні алгоритми керування, але й оптимізувати енергетичний баланс пристрою. У роботі [51, 77, 78] доведено високу ефективність мікроконтролера ESP32 як основи для систем інтелектуального енергомоніторингу та керування у реальному часі. Також у дослідженні [53] підкреслюють, що ключовим фактором енергоефективності ESP32 є апаратна оптимізація та правильне масштабування напруги. Вони зазначають, що використання надійних перетворювачів напруги замість примітивних лінійних стабілізаторів дозволяє уникнути теплових втрат та стабілізувати роботу модулів UART, I2C, SPI та Wi-Fi при пікових навантаженнях.

Окрім апаратної оптимізації, сучасні наукові напрями спрямовані на використання алгоритмічних методів енергозбереження. У роботі [49] доводять, що інтеграція марковських моделей, інтелектуальних алгоритмів перемикання та планування режимів сну дозволяє кардинально оптимізувати використання енергії у гібридних WSN-мережах. Водночас особливої уваги потребують системи, що взаємодіють з термоелектричними елементами. У дослідженні [50], де розглядаються гібридні системи з використанням ТЕ-модулів (елементів Пельтьє), наголошується на тому, що такі компоненти характеризуються величезними тепловими та електричними струмами, тому керування ними вимагає застосування адаптивних систем перетворення енергії та жорсткого алгоритмічного контролю.

Проведений аналіз проблематики енергоефективності IoT [49-53] повністю обґрунтовує обраний вектор проектування апаратної частини. Застосування імпульсних понижуючих DC-DC перетворювачів є практичною реалізацією

концепції апаратного масштабування напруги [53], що гарантує стабільність ліній напруги.

1.10 Висновки до першого розділу

У першому розділі проведено аналіз сучасного стану технологій та методів автоматизованого керування мікрокліматом теплиць. Аналіз існуючих архітектурних рішень показав, що традиційні підходи на базі класичних регуляторів та суто хмарних IoT-платформ мають суттєві обмеження для ефективного впровадження. Вони характеризуються критичною залежністю від стабільності інтернет-з’єднання, значними мережевими затримками та нездатністю ефективно працювати в умовах апаратних збоїв. Це обґрунтовує необхідність створення гнучкої гібридної Cloud-Edge системи на базі граничних мікроконтролерів та відмовостійких локальних алгоритмів.

Огляд еволюції методів керування складними термодинамічними об'єктами підтвердив неефективність класичних реактивних систем в умовах значної теплової інерційності теплиць та впливу стохастичних збурень. Порівняльний аналіз алгоритмів довів, що для задач упереджувального регулювання найоптимальнішим вибором є використання рекурентних нейронних мереж та моделей прогнозування часових рядів. Зокрема, використання прогнозуючої аналітики дозволяє системі завчасно компенсувати мікрокліматичні коливання, досягаючи балансу між високою точністю підтримки заданих параметрів та енергоефективністю.

Аналіз стратегій оптимізації виявив, що базового прогнозування недостатньо для подолання конфлікту між ідеальними умовами для рослин та мінімізацією витрат на енергоносії. Для усунення цього конфлікту необхідно застосовувати алгоритми багатокритеріальної оптимізації. Комбінація генетичних алгоритмів для налаштування параметрів та локального нечіткого виведення визнана найбільш надійним підходом для формування плавних та енергоефективних керуючих впливів на виконавчі механізми.

Отже, проблема дослідження зумовлена об’єктивними вимогами щодо необхідності зниження енергоспоживання тепличних комплексів при одночасному збереженні високої якості мікроклімату та стійкості системи до інфраструктурних збоїв (втрата зв'язку, затримки). Необхідно зменшити ступінь цих суперечностей через розроблення відповідних апаратно-програмних. Обґрунтуванням необхідності такого впливу є пряме підвищення ефективності та економічної доцільності розгортання розумних агропромислових систем.

З огляду на вищезазначене, метою магістерської роботи є вдосконалення існуючих методів інтелектуального керування мікрокліматом теплиці на основі прогнозуючого аналізу даних та локального нечіткого виведення для забезпечення їх безперебійної роботи в умовах граничних обчислень та обмежених енергетичних ресурсів.

Для досягнення поставленої мети необхідно вирішити такі задачі дослідження:

− проаналізувати сучасний стан, існуючі архітектурні підходи та методи штучного інтелекту у сфері автоматизації тепличних господарств;

− розробити математичні моделі прогнозування мікрокліматичних параметрів для компенсації теплової інерційності об'єкта;

− удосконалити метод локального керування виконавчими механізмами шляхом впровадження алгоритмів нечіткого виведення та багатокритеріальної оптимізації енерговитрат;

− спроєктувати та розробити апаратно-програмне забезпечення гібридної (Cloud-Edge) системи з урахуванням архітектурної вимоги щодо забезпечення автономності граничного вузла на базі мікроконтролера;

− провести експериментальні дослідження для оцінки точності прогнозування, енергоефективності та відмовостійкості оптимізованої системи в реальних умовах експлуатації.

2 МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ДИНАМІКИ МІКРОКЛІМАТУ

2.1 Концептуальна модель відмовостійкої IoT-системи керування мікрокліматом

Для реалізації інтелектуальної системи керування мікрокліматом теплиці було обрано концепцію розподілених обчислень, яка поєднує граничні обчислення та хмарні сервіси. У дослідженні [20] зазначається, що традиційні централізовані хмарні моделі часто виявляються неефективними для сучасних сільськогосподарських сценаріїв через високу затримку, обмеження пропускної здатності мережі та критичну залежність від стабільності інтернет-з'єднання. Автори наголошують, що архітектура Cloud-Edge-Device дозволяє подолати ці проблеми шляхом перенесення частини обчислень безпосередньо на граничний рівень, залишаючи за хмарою ресурсоємні задачі глибокого аналізу та історичної оптимізації.

Відповідно до цієї парадигми, розроблена архітектура системи складається з трьох логічних рівнів, що тісно взаємодіють між собою для забезпечення безперебійного функціонування об'єкта (рис. 2.1).

- Фізичний рівень. Включає сенсорну мережу для збору параметрів мікроклімату (температура, вологість) та виконавчі механізми (елемент Пельтьє, фітолампи, водяна помпа, сервопривід). Даний рівень відповідає за базовий збір телеметричних даних та виконання прямих фізичних команд безпосередньо в середовищі теплиці.
- Граничний рівень. Роль інтелектуального граничного вузла виконує мікроконтролер ESP32. Цей рівень забезпечує локальну обробку даних з мінімальною затримкою, що є критично важливим для сценаріїв, які вимагають негайного та жорсткого реагування на різкі зміни середовища. У розробленій системі ESP32 не просто транслює дані, а й реалізує керування актуаторами в реальному часі за допомогою алгоритмів нечіткої логіки.
- Хмарний рівень. Представлений віддаленим веб-сервером. Хмара забезпечує централізовану обробку та зберігання масштабних масивів

телеметричних даних. На цьому рівні розгортається статистична прогнозуюча модель ARIMA, яка на основі накопиченої історії генерує прогноз зміни мікроклімату і надсилає на ESP32 коригуючі параметри. У роботі [22] вказується, що такий підхід до розподілу навантаження суттєво зменшує загальне енергоспоживання системи та розвантажує мережевий трафік, оскільки важкі математичні обчислення виконуються на потужних серверах, а не на малопотужних локальних пристроях.

Рисунок 2.1 – Загальна трирівнева архітектура системи

Важливим та часто недооціненим аспектом розробки IoT-систем для сільського господарства є забезпечення їхньої безперебійної роботи в умовах нестабільного зв’язку. У роботі [21] розглядається проблематика створення систем відмовостійкості для Edge-пристроїв в умовах обмежених ресурсів. Автори підкреслюють необхідність розробки надійних механізмів, які дозволяють системі динамічно перемикатися на резервні, локальні моделі керування у випадку відсутності доступу до зовнішніх обчислювальних потужностей або хмарних

сервісів. Застосування концепції «Автономного виживання» є обов'язковою умовою для запобігання втрати врожаю при апаратних чи мережевих збоях.

Базуючись на цій концепції, у розроблений Edge-вузол було імплементовано математичну модель автономного виживання. Суть моделі полягає у безперервному моніторингу доступності мережевого каналу зв'язку та автоматичній зміні стратегії керування. Для формалізації цього процесу введено булеву змінну стану мережі $S_{net}(t)$, яка визначається на основі часу останнього успішного сеансу зв'язку з сервером tlast та гранично допустимого часу очікування $\Delta t_{timeout}$:

$$S_{net}(t) = \begin{cases} 1, & (t - t_{last}) \leq \Delta t_{timeout} \\ 0, & (t - t_{last}) > \Delta t_{timeout} \end{cases}. \quad (2.1)$$

Залежно від значення $S_{net}(t)$, цільова температура $T_{target}(t)$, яку повинен підтримувати локальний нечіткий контролер у будь-який момент часу t, обчислюється за формулою:

$$T_{target}(t) = S_{net}(t) * T_{ARIMA}(t) + \big(1 - S_{net}(t)\big) * T_{base}\,, \quad (2.2)$$

де:

- $T_{ARIMA}(t)$ – динамічна цільова температура, отримана з хмарного сервера на основі прогнозу;
- T_{base} – статична (безпечна) температура виживання, яка жорстко закодована в енергонезалежній пам'яті мікроконтролера ESP32 (наприклад, оптимальний мінімум для підтримки життєдіяльності рослин, +18°C);
- t – поточний час функціонування системи.

Отже, коли $S_{net}(t)=1$, система працює в режимі глобальної оптимізації, отримуючи інтелектуальні підказки від хмари. У випадку аварійної втрати зв'язку ($S_{net}(t)=0$), система миттєво відсікає хмарну складову і переходить до ізольованого керування на базі локально визначеного параметра T_{base}.

Логіка перемикання станів системи та формування цільової температури залежно від доступності мережі у вигляді блок-схеми наведена на рисунку 2.2.

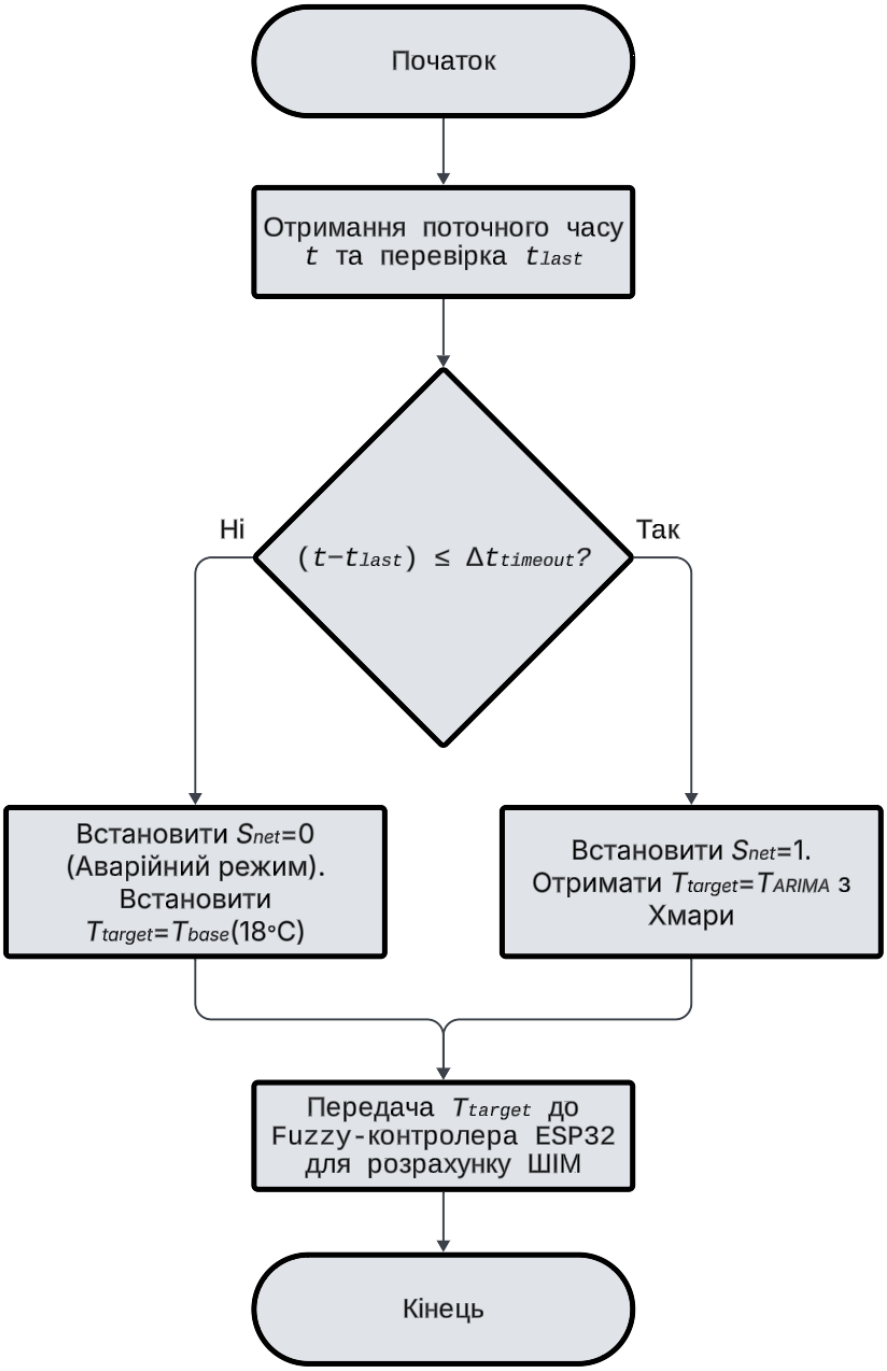


Рисунок 2.2 – Блок-схема алгоритму забезпечення відмовостійкості граничного вузла

Таким чином, розроблена архітектура не лише поєднує переваги прогнозуючого інтелектуального керування, але й математично гарантує базову

життєдіяльність біотехнічного об’єкта у критичних ситуаціях. Це повністю вирішує фундаментальну проблему залежності IoT-систем від безперебійності інтернет-з’єднання, роблячи систему керування теплицею стійкою та придатною для реальної експлуатації [21].

2.2 Математичне моделювання динаміки мікроклімату та прогноз на базі ARIMA

Процес керування мікрокліматом теплиці ускладнюється значною тепловою інерційністю об’єкта та постійним впливом неконтрольованих стохастичних збурень (зміна сонячного випромінювання, швидкості вітру, хмарності). У дослідженні [13] доведено, що динаміка зміни фізичних параметрів середовища (зокрема температури та концентрації газів) у закритому ґрунті має чітко виражений характер часового ряду. Поточний стан системи жорстко визначений її попередніми станами. Отже, для забезпечення енергоефективного прогнозуючого керування необхідно використовувати математичний апарат аналізу часових рядів.

Важливою умовою для адекватної роботи прогнозуючих моделей часових рядів є безперервність та консистентність вхідних даних. Оскільки збір метрик мікроклімату здійснюється граничним вузлом та передається на хмарний сервер через бездротову мережу Wi-Fi, існує ймовірність втрати окремих пакетів даних через мережеві колізії або тимчасові апаратні збої.

Модель ARIMA вимагає строго рівномірної часової сітки. Тому перед подачею масиву температур $Y=\{Y_1, Y_2, \ldots, Y_N\}$ на вхід алгоритму, сервер виконує процедуру попередньої обробки, яка включає фільтрацію аномальних викидів та інтерполяцію пропущених значень. Для відновлення пропущеного значення температури в момент часу t використовується метод лінійної інтерполяції на основі суміжних валідних відліків:

$$Y_t = Y_{t-1} + \frac{Y_{t+k} - Y_{t-1}}{k+1}, \tag{2.3}$$

де:

- Y_{t-1} – останнє успішно передане значення;
- Y_{t+k} – наступне успішне значення після збою;
- k – кількість пропущених тактів вимірювання.

Лише після формування безперервного та відфільтрованого масиву даних ініціюється процедура математичного прогнозування.

Відповідно до розробленої хмарно-граничної архітектури, задачею хмарного сервера є генерація випереджального прогнозу температури T_{target} для локального контролера ESP32. Для вирішення цієї задачі як базову математичну модель було обрано інтегровану модель авторегресії – ковзного середнього. У роботі [12] експериментально підтверджено, що гібридні підходи на базі ARIMA демонструють високу точність при короткостроковому прогнозуванні температури та вологості в теплицях, ефективно згладжуючи шумові викиди даних.

Фундаментальний математичний опис моделі ARIMA базується на працях з економетрики та статистичного прогнозування. Згідно з методологією, викладеною у роботі [23], модель ARIMA(p,d,q) є лінійною комбінацією трьох складових: авторегресії порядку p, інтегрування порядку d та ковзного середнього порядку q.

Нехай дискретний часовий ряд Y_t описує значення температури в теплиці у момент часу t:

1. Модель авторегресії *AR(p)*. Авторегресійна складова моделює залежність поточного значення температури від її p попередніх значень. Фізично це відображає теплову інерцію об’єкта: температура у поточну хвилину багато в чому визначається температурою кілька хвилин тому. Згідно з [23], рівняння моделі *AR(p)* має вигляд:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t \,, \qquad (2.4)$$

або у вигляді суми:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \varepsilon_t \,, \tag{2.5}$$

де:

- c – константа (зсув);
- ϕ_i – вагові коефіцієнти авторегресії (параметри моделі);
- Y_{t-1} – значення температури у попередні моменти часу;
- ε_t – випадкова похибка у момент часу t, що має нульове математичне сподівання $E(\varepsilon_t)=0$ та постійну дисперсію $Var(\varepsilon_t)=\sigma^2$.

2. Модель ковзного середнього *MA(q)*. Складова ковзного середнього моделює реакцію системи на непередбачувані зовнішні збурення, наприклад, різкий порив вітру або тимчасове затінення теплиці хмарою. Модель *MA(q)* використовує минулі помилки прогнозування для коригування поточного значення [23]:

$$Y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \,, \tag{2.6}$$

або у вигляді суми:

$$Y_t = \mu + \varepsilon_t + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} \,, \tag{2.7}$$

де:

- μ – середнє значення часового ряду;
- θ_j – вагові коефіцієнти ковзного середнього;
- ε_{t-j} – значення похибок у попередні моменти часу.

Узагальнена модель ARMA(p,q) для стаціонарного часового ряду утворюється шляхом суперпозиції цих двох рівнянь.

3. Інтегрування *I(d)* та узагальнена модель ARIMA. Основною вимогою для застосування моделі ARMA є стаціонарність часового ряду. Оскільки мікроклімат теплиці має виражені добові тенденції (нагрівання вдень, охолодження вночі), вихідний ряд температур є нестаціонарним.

Для приведення ряду до стаціонарного вигляду застосовується операція взяття різниць *d*-го порядку [23]. Введено оператор зсуву назад *B*, який визначається як:

$$BY_t = Y_{t-1}\,, \tag{2.8}$$

тоді операція взяття першої різниці (для усунення лінійних тенденцій) запишеться як:

$$\nabla Y_t = Y_t - Y_{t-1} = (1 - B)Y_t\,, \tag{2.9}$$

для різниці порядку *d*:

$$\nabla^d Y_t = (1 - B)^d Y_t\,, \tag{2.10}$$

інтегруючи оператор зсуву в поліноміальні рівняння *AR* та *MA*, отримано рівняння моделі *ARIMA(p,d,q)* у компактному вигляді [23]:

$$\phi(B)(1 - B)^d Y_t = c + \theta(B)\varepsilon_t\,, \tag{2.11}$$

де поліноми оператора зсуву визначаються як:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p\,, \tag{2.12}$$

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q\,. \tag{2.13}$$

У розробленій архітектурі процес прогнозування реалізується на віддаленому сервері циклічно, за методологією Бокса-Дженкінса, блок-схема алгоритму зображена на рисунку 2.3. Сервер отримує від граничного вузла ESP32 масив останніх вимірів температури $Y=\{Y_{t-n},\ldots,Y_t\}$.

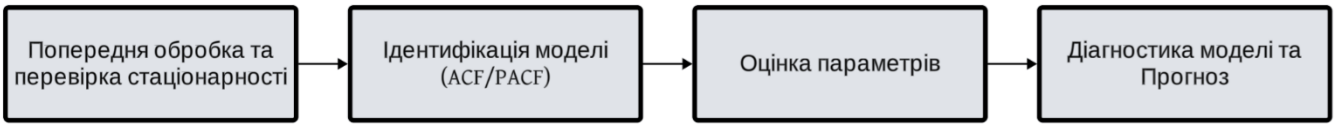


Рисунок 2.3 – Блок-схема алгоритму Бокса-Дженкінса

Визначення оптимальних порядків авторегресії p та ковзного середнього q здійснюється на основі аналізу автокореляційної функції (ACF) та часткової автокореляційної функції (PACF) часового ряду. Відповідно до методології [23], автокореляційна функція показує ступінь лінійної залежності між поточним значенням температури та її попередніми значеннями із запізненням на k кроків.

Вибіркова автоковаріація ck для запізнення k розраховується за формулою:

$$c_k = \frac{1}{N}\sum_{t=1}^{N-k}(Y_t - \bar{Y})(Y_{t+k} - \bar{Y}), \tag{2.14}$$

де:

- N – загальна кількість телеметричних відліків у вибірці;
- $\bar{Y}$ – вибіркове середнє значення температури.

На основі автоковаріації обчислюється коефіцієнт автокореляції r_k:

$$r_k = \frac{c_k}{c_0}, \tag{2.15}$$

де c_0 – дисперсія часового ряду. Аналіз графіків ACF та PACF дозволяє серверу автоматично звузити простір пошуку для параметрів p та q.

Приклад графіку ACF/PACF зображено на рисунку 2.4.

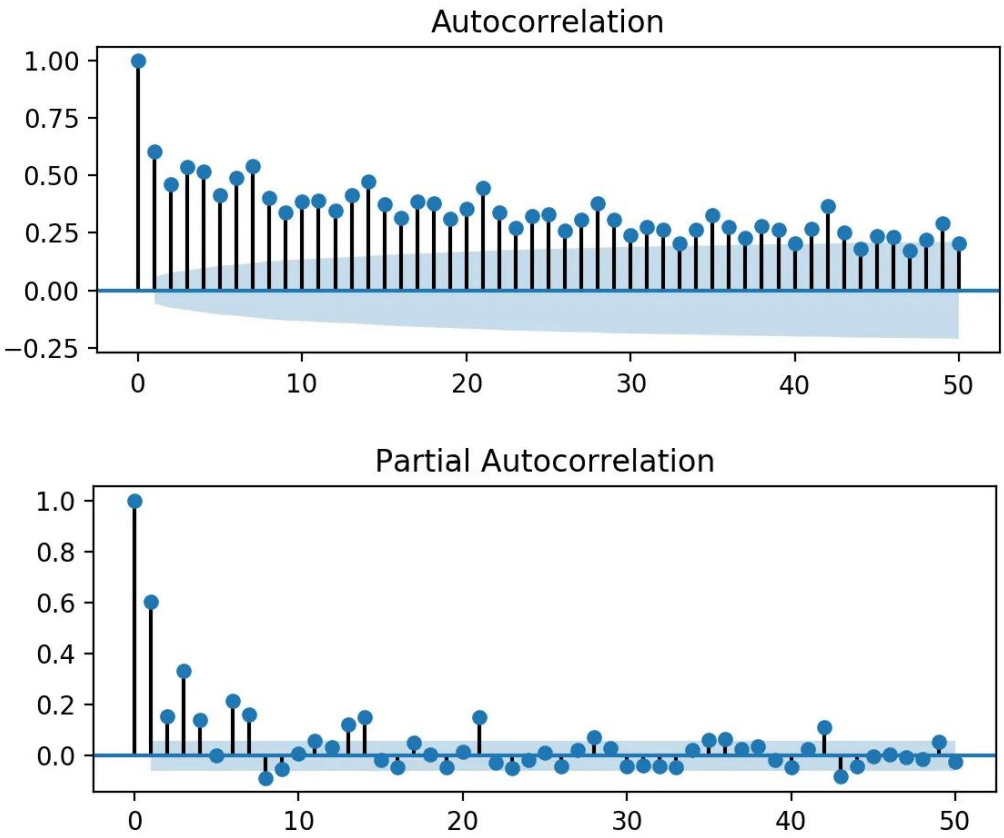


Рисунок 2.4 – Графік ACF/PACF [24]

Після структурної ідентифікації сервер виконує оцінку вагових коефіцієнтів ϕ_i та θ_j за допомогою методу максимальної правдоподібності. Для вибору найкращої конфігурації моделі з декількох альтернативних (щоб уникнути перенавчання моделі) алгоритм розраховує інформаційний критерій Акаіке (AIC) та Байєсівський інформаційний критерій (BIC):

$$AIC = -2\ln(L) + 2k_{params}\,, \tag{2.16}$$

$$BIC = -2\ln(L) + k_{params}\ln(N)\,, \tag{2.17}$$

де:

- L – максимізоване значення функції правдоподібності моделі;
- $k_{params=p+q+1}$ – загальна кількість оцінюваних параметрів.

Хмарний сервер ітеративно перебирає комбінації параметрів (наприклад, за допомогою функції *auto_arima* в Python) і обирає ту архітектуру, для якої значення

інформаційного критерію Акаіке є мінімальним. Саме ця оптимальна модель застосовується для розрахунку прогнозуючого кроку, який відправляється на граничний контролер ESP32. Програмно реалізований математичний алгоритм працює як послідовний процес, що розпочинається з ідентифікації моделі, під час якої виконується перевірка часового ряду на стаціонарність за допомогою розширеного тесту Дікі-Фуллера та визначається необхідний порядок диференціювання d. Наступним етапом є оцінка параметрів, що передбачає розрахунок оптимальних значень коефіцієнтів ϕ_i та θ_j шляхом мінімізації інформаційних критеріїв Акаіке або Байєса. Завершальною стадією виступає прогнозування, яке полягає в обчисленні майбутнього значення на h кроків вперед.

Для завдань прогнозуючого керування теплицею розраховується прогноз на один крок $\hat{Y}_{t+1}$, який повертається на ESP32 у вигляді відповідної керуючої змінної. Використання математичного апарату ARIMA на рівні хмарного сервера гарантує високу точність обчислення майбутніх теплових станів теплиці, що дозволяє локальному контролеру завчасно реагувати на температурні шоки, мінімізуючи споживання електроенергії елементом Пельтьє та забезпечуючи максимальний рівень комфорту для рослин.

2.3 Математична модель локального регулятора на базі нечіткої логіки

Розроблена гібридна архітектура IoT-системи передбачає, що в разі втрати зв’язку з хмарним сервером або у проміжках між оновленнями прогнозуючих прогнозів, граничний вузол повинен самостійно і безперебійно підтримувати задані параметри мікроклімату.

2.3.1 Обґрунтування переходу від релейної логіки до нечіткого керування

У попередніх ітераціях розробки системи локальне керування мікрокліматом здійснювалося на базі мікроконтролера Arduino з використанням класичної релейної логіки та жорстко заданих порогових значень [54].

Наприклад, алгоритм керування температурою ґрунту та освітленням працював за бінарним принципом: якщо температура ґрунту знижувалася до позначки нижче 26°C, елемент Пельтьє активувався на фіксовану потужність, а в разі досягнення цільового показника – повністю вимикався. Аналогічний релейний підхід застосовувався і для освітлення, де падіння рівня нижче 70 lx ініціювало включення LED-фітострічки на заданий рівень. Проте, як доведено в дослідженні [7], такий метод має суттєві інженерні недоліки, головним з яких є неминуче виникнення перерегулювання та автоколивань навколо цільового значення. Постійні циклічні перемикання актуаторів не лише знижують точність підтримки мікроклімату, але й значно прискорюють механічне зношування реле та термічну деградацію напівпровідникового елемента Пельтьє.

Ще однією вагомою проблемою була відсутність урахування взаємозв'язків між фізичними параметрами середовища. Зокрема, відкриття кватирки серводвигуном для зниження температури повітря неминуче провокувало різке падіння вологості, на що система одразу реагувала включенням зволожувача, створюючи прямий конфлікт виконавчих механізмів. Ситуація ускладнювалася апаратними обмеженнями підсистеми живлення: використання двох малопотужних блоків 9В 1А у попередній бакалаврській роботі [54] призводило до просадок напруги під час пікових навантажень. Хоча впровадження нового потужнішого джерела на 12В 5А уникає цих апаратних збоїв, а статичне програмне обмеження ШІМ вирішувало питання перевантаження, такий підхід загалом робив реакцію системи занадто повільною при необхідності компенсації сильних відхилень мікроклімату.

2.3.2 Математичний апарат фазифікації

Математичною основою для побудови нечіткого контролера є теорія нечітких множин. Класична булева логіка розширюється за рахунок введення функції належності $\mu_A(x)$, яка визначає ступінь приналежності чіткого значення x до нечіткої множини A у діапазоні [0,1] [25, 79, 80].

Процес фазифікації полягає у перетворенні чітких числових значень, отриманих з аналогових та цифрових датчиків у реальному часі (наприклад, значення температури 23.5°C), у лінгвістичні терми з відповідним ступенем належності. Для розробленої системи інтелектуального керування мікрокліматом теплиці вхідними лінгвістичними змінними обрано похибки регулювання базових параметрів, що дозволяє реалізувати класичний принцип упереджувального керування за відхиленням:

- E_T – похибка температури (різниця між поточною температурою та цільовою T_{target}, яку встановив хмарний сервер);
- E_H – похибка вологості повітря;
- E_S – похибка вологості грунту.

Для кожної лінгвістичної змінної було обрано три терми: Negative (N) – нижче норми, Zero (Z) – норма, Positive (P) – вище норми. Математично ці терми описуються за допомогою трикутних функцій належності, які є обчислювально ефективними для мікроконтролера ESP32.

Зокрема, трикутна функція належності $\mu(x)$ задається трьома скалярними параметрами (a,b,c), де параметри a та c визначають ліву та праву координати основи трикутника на універсальній множині (межі, де функція набуває нульових значень), а параметр b фіксує координату його вершини, в якій ступінь належності дорівнює одиниці [25]:

$$\mu(x) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a < x \le b \\ \dfrac{c-x}{c-b}, & b < x \le c \\ 0, & x > c \end{cases}. \qquad (2.18)$$

Приклад графіків трикутних функцій належності для похибки температури зображено на рисунку 2.5.

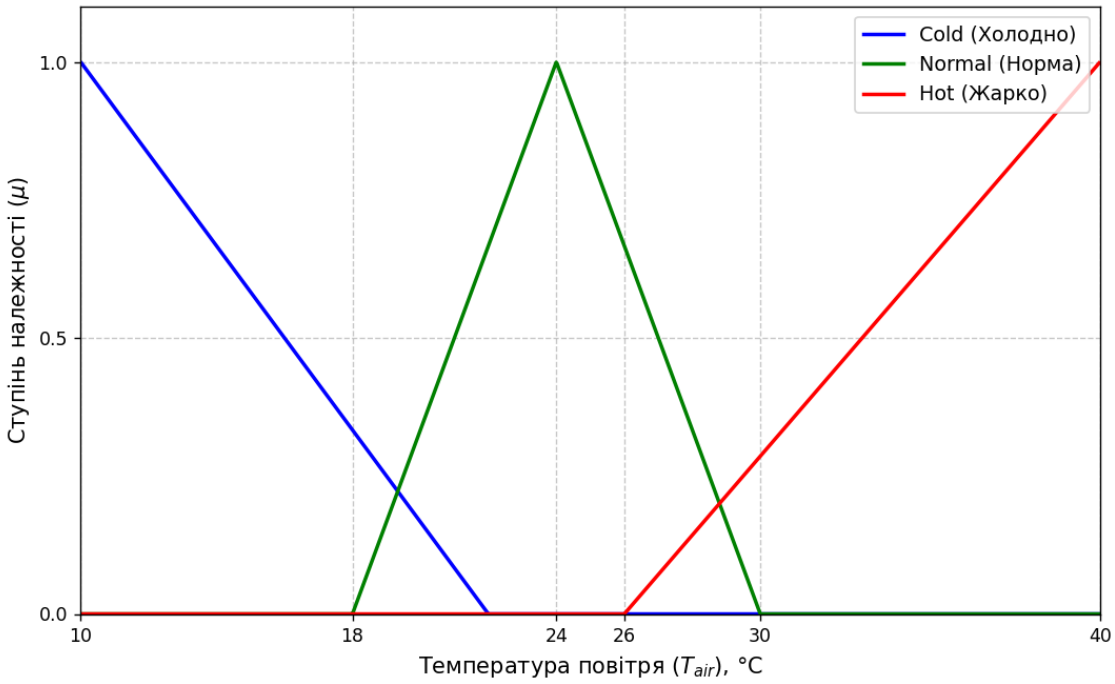


Рисунок 2.5 – Графіки трикутних функцій належності для похибки температури

2.3.3 Розробка бази правил та механізм логічного виведення

Другим етапом роботи регулятора є оцінка сформованих нечітких значень за допомогою Бази Правил. База правил перетворює інженерний досвід у набір нечітких імплікацій.

Якщо у попередній версії коду конфлікти вирішувалися жорсткими затримками (*delay(5000)*), то у нечіткій логіці використовується багатовимірний механізм виведення Мамдані. Правила формуються у вигляді:

- правило 1: ЯКЩО (E_T is Positive) І (E_H is Positive) ТОДІ (Servo is *Open_Wide*) І (Humidifier is Off);
- правило 2: ЯКЩО (E_T is Negative) І (E_S is Negative) ТОДІ (Peltier is *Heat_High*) І (Pump is On_Short).

Математично операція логічного "І" між двома нечіткими множинами A та B реалізується через операцію пошуку мінімуму (t-норми) [25]:

$$\mu_{A\cap B}(x,y) = min[\mu_A(x), \mu_B(y)]\,. \qquad (2.19)$$

Результатом застосування кожного правила є скорочена вихідна функція належності для актуатора. Оскільки спрацьовує одразу кілька правил з різним ступенем істинності, фінальна нечітка множина вихідної змінної *C* формується шляхом об'єднання (операція максимуму, *s*-норма) результатів усіх правил:

$$\mu_C(z) = max[\mu_{Rule1}(z), \mu_{Rule2}(z), \dots, \mu_{RuleN}(z)] . \quad (2.20)$$

Для практичної реалізації алгоритму на мікроконтролері ESP32 було визначено конкретні діапазони для вхідної змінної «Температура повітря» (*T*). На основі агрономічних норм для тепличних культур, область визначення для *T* встановлено у межах від 10°C до 40°C. Цей діапазон покривається трьома нечіткими термами:

1. Cold (Холодно). Трикутна функція належності з параметрами (10,10,22). При температурі 10°C і нижче ступінь належності $\mu_{Cold}=1$. При 22°C вона спадає до 0.

2. Normal (Норма). Трикутна функція належності з параметрами (18,24,30). Вершина (ідеал) знаходиться на рівні 24°C ($\mu_{Normal}=1$).

3. Hot (Жарко). Трикутна функція належності з параметрами (26,40,40).

Замість жорстких умов, які використовувались у попередніх версіях системи, було розроблено базу нечітких правил. У таблиці 2.1 наведено фрагмент розробленої матриці логічного виведення Мамдані для керування термоелектричним модулем Пельтьє та сервоприводом кватирки залежно від температури та прогнозу з сервера (T_{ARIMA}).

Таблиця 2.1 – Фрагмент бази правил нечіткого контролера

Поточна T	Прогноз T_{ARIMA}	Дія Пельтьє (ШІМ)	Дія Сервоприводу (Кватирка)
Cold	Спадає	High (Макс. нагрів)	Closed (Закрито)
Cold	Зростає	Low (Підтримка)	Closed (Закрито)

Кінець таблиці 2.1 – Фрагмент бази правил нечіткого контролера

Normal	Стабільно	Off (Вимкнено)	Closed (Закрито)
Normal	Зростає	Off (Вимкнено)	(Half-Open) (Привідкрито)
Hot	Зростає	Off (Вимкнено)	Open (Відкрито повністю)

Перевагою такої матриці є те, що для перехідних значень (наприклад, *T*=20°C, що частково належить до *Cold* і частково до *Normal*) мікроконтролер одночасно активує два правила з різними ваговими коефіцієнтами. Це забезпечує абсолютно плавний перехід ШІМ-сигналу від стану *Low* до *Off* без ривків, які були характерні для релейного керування.

2.3.4 Дефазифікація та адаптивне керування підсистемою живлення

Завершальним етапом роботи локального контролера є дефазифікація – перетворення результуючої нечіткої множини у чітке значення керуючого впливу. Для ESP32 цим значенням є коефіцієнт заповнення ШІМ-сигналу у діапазоні від 0 до 1023 для керування потужністю елемента Пельтьє та LED-освітлення, або кут повороту від 0◦ до 180◦ для сервоприводу кватирки.

Відповідно до методології [25], для задач автоматичного керування найвищу точність демонструє метод центру ваги. Чітке значення виходу Z^* розраховується як абсциса центру площі фігури, утвореної функцією $\mu_C(z)$:

$$Z^* = \frac{\int_{Z_{min}}^{Z_{max}} \mu_C(z) * z \, dz}{\int_{Z_{min}}^{Z_{max}} \mu_C(z) dz}. \quad (2.21)$$

Оскільки ESP32 є цифровим мікроконтролером, інтеграл замінюється дискретною сумою для *N* точок розбиття вихідного діапазону:

$$Z^* = \frac{\sum_{i=1}^{N} \mu_C(z_i) * z_i}{\sum_{i=1}^{N} \mu_C(z_i)}. \quad (2.22)$$

Використання нечіткого контролера з таким математичним апаратом дозволило елегантно вирішити проблему обмеженої потужності апаратної частини. Впровадження правил, що враховують загальне енергоспоживання, гарантує, що сумарний струм, який споживається актуаторами, ніколи не перевищить номінал блоку живлення. Наприклад, якщо нечіткий регулятор визначає високу потребу у фіто-освітленні ($Z^*LED \rightarrow 1023$), база правил автоматично знижує максимально можливий ступінь нагріву Пельтьє, формуючи більш похилу криву набору температури, що було неможливо реалізувати у рамках жорсткої релейної логіки.

2.4 Структурно-енергетична модель апаратної платформи

Перехід від релейної логіки до прогнозуючого інтелектуального керування вимагає суттєвої модернізації апаратної частини системи. У рамках даного дослідження було розроблено нову структурну модель апаратно-програмного комплексу, орієнтовану на підвищення обчислювальної потужності та забезпечення енергетичної стабільності граничного вузла.

2.4.1 Обґрунтування вибору апаратної платформи граничного вузла

На етапі попередніх досліджень базовим контролером виступала плата сімейства Arduino (на базі 8-бітного мікроконтролера ATmega328P). Незважаючи на простоту прототипування, ця платформа має критичні обмеження для сучасних IoT-систем: низька тактова частота (16 МГц), малий обсяг оперативної пам’яті (2 КБ SRAM) та необхідність використання зовнішніх модулів зв’язку.

У розробленій архітектурі ядром системи виступає мікроконтролер ESP32. Як підтверджується у дослідженні [26], розробка IoT-систем на базі цього мікроконтролера для розумного поливу та кліматичного моніторингу в теплицях є

високоефективним рішенням, що значно перевершує застарілі 8-бітні аналоги. Технічна доцільність вибору саме цього мікроконтролера обґрунтовується насамперед наявністю двоядерного 32-бітного процесора з тактовою частотою 240 МГц. Така обчислювальна потужність дозволяє системі паралельно виконувати складні математичні операції фазифікації та дефазифікації, одночасно підтримуючи стабільну та безперебійну роботу мережевого стека протоколів TCP/IP.

Крім того, апаратна інтеграція контролера Wi-Fi забезпечує надійну передачу зібраної телеметричної інформації у хмарне середовище, відкриваючи користувачам широкі можливості для віддаленого моніторингу та диспетчеризації технологічних процесів [26]. Не менш вагомою архітектурною перевагою є наявність у ESP32 незалежних апаратних каналів генерації ШІМ-сигналів високої роздільної здатності. Це гарантує максимально точне керування силовими транзисторами (MOSFET) без виникнення програмних затримок або блокувань, що є абсолютно критичною вимогою для коректного функціонування локального нечіткого контролера та плавного регулювання потужності виконавчих механізмів.

2.4.2 Розподіл енергетичних ліній та баланс потужностей периферії

Критично вразливим місцем попереднього прототипу була підсистема живлення на базі малопотужних адаптерів, сумарної потужності яких було недостатньо для виведення актуаторів на номінальні режими. Для забезпечення стабільної роботи системи застосовано топологію централізованого живлення типу "Дерево". Основним джерелом енергії обрано імпульсний блок живлення S-60-12, який забезпечує напругу 12 В та максимальний струм 5 А (номінальна потужність 60 Вт).

Відповідно до принципів силової електроніки, для фізичної ізоляції керуючих кіл від індуктивних навантажень розроблено каскадну схему зниження напруги на базі двох паралельних DC-DC імпульсних перетворювачів XL4015. Обидва модулі відкалібровано на вихідну напругу 5 В, однак вони виконують

принципово різні функції для усунення перехресних перешкод та просадок напруги під час пускових струмів моторів.

Лінія 5 В (Логічна та вимірювальна). Формується першим модулем XL4015 і живить виключно чутливу електроніку:

- дисплей LCD 1602 I2C: 50 мА;
- плату розширення мікроконтролера ESP32.

Безпосередньо на платі ESP інтегрований лінійний стабілізатор напруги (LDO типу AMS1117-3.3), який перетворює вхідні 5 В у високостабільні 3.3 В (захищені від шумів 5-вольтових моторів завдяки високому коефіцієнту фільтрації пульсацій PSRR стабілізатора).

Від цих чистих 3.3 В через контакти SVG живляться мікроконтролер (до 500 мА) та масив вимірювальних датчиків:

- датчик DHT22, струм 2.5 мА;
- ємнісний датчик вологості грунту (DFRobot), струм 5 мА;
- датчик освітленості BH1750FVI, струм 0.12 мА;
- температурний датчик DS18B20, струм 1 мА.

Сумарне максимальне споживання по цій лінії становить близько 0.56 А (2.8 Вт).

Лінія 5В (Силова ізольована). Формується другим модулем XL4015 і призначена для живлення потужних та індуктивних навантажень, генеруючих електромагнітний шум:

- водяна помпа, пікове споживання струму 220 мА;
- сервопривід SG90, пусковий струм до 650 мА;
- USB-зволожувач повітря, струм 400 мА;
- LED-фітострічка, сумарний струм 480 мА;
- термоелектричний модуль Пельтьє (TEC1-12706).

Важливим етапом апаратної оптимізації стала уніфікація силових ключів. Під час тестів було виявлено явище паразитного мікроструму через оптопари модулів D4184, що призводило до паразитного світіння LED-стрічки при ШІМ=0. Силові

ключі було замінено на N-канальні транзистори IRLB8721 зі стягуючими резисторами 10 кОм на затворі, що забезпечило ідеальне відсікання струму.

Сумарне споживання актуаторів (без урахування модуля Пельтьє) становить близько 1.75 А (8.75 Вт). Загальна потужність логіки та базової периферії з урахуванням 90% ККД модулів XL4015 складає:

$$P_{base} = \frac{2.8 + 8.75}{0.9} \approx 12.83 \text{ Вт}. \quad (2.23)$$

2.4.3 Математичне та емпіричне обґрунтування теплових режимів

Під час первинних стендових випробувань модуля Пельтьє при прямому живленні від магістралі 12 В було зафіксовано явище теплового перенасичення. Надлишкова потужність призвела до розплавлення внутрішнього припою та фізичної руйнації напівпровідникового елемента.

Для вирішення цієї проблеми модуль було замінено, затиснуто між алюмінієвим радіатором та масивною пластиною-розподілювачем товщиною 0.8 см, а його живлення переведено на силову лінію 5 В.

Розраховано нові електричні показники. Внутрішній опір модуля TEC1-12706 становить орієнтовно R≈2 Ом. При живленні від 5 В максимальний струм становитиме:

$$I_{\max\_5V} = \frac{U}{R} = \frac{5}{2} = 2.5 \text{ А}. \quad (2.24)$$

Теоретична максимальна споживана потужність елемента Пельтьє при 5 В:

$$P_{Peltier\_\max\_5V} = U * I = 5 * 2.5 = 12.5 \text{ Вт}. \quad (2.25)$$

Теоретичне пікове навантаження на головний блок живлення (P_{Σ_max}) при одночасному включенні всіх актуаторів складе:

$$P_{\Sigma\_max} = P_{Base} + \frac{P_{Peltier\_\max\_5V}}{0.9} = 12.83 + 13.88 = 26.71 \text{ Вт}. \quad (2.26)$$

Отримане значення є вдвічі меншим за безпечний ліміт джерела живлення S-60-12 (48 Вт), що гарантує абсолютну електричну надійність системи та відсутність перевантажень.

Проте, незважаючи на безпечний рівень електричного струму, під час подальших емпіричних досліджень було виявлено проблему теплової інерції. Масивна алюмінієва пластина акумулювала тепло і продовжувала нагрівати ґрунт навіть після повного вимкнення модуля Пельтьє, що призводило до перерегулювання мікроклімату.

Для подолання цієї фізичної проблеми застосовано апаратну ШІМ-модуляцію роздільністю 10-біт (ШІМ $D \in [0,1023]$), у комплексі з програмним обмеженням. Шляхом серії емпіричних тестів було визначено, що ідеальним балансом між достатньою швидкістю нагрівання та відсутністю теплової інерції є максимальне значення ШІМ D_{max}=600.

Ефективна керуюча потужність Пельтьє при такому програмному обмеженні становить:

$$P_{effective} = P_{Peltier\_\max\_5V} * \left(\frac{D_{max}}{1023}\right) = 12.5 * \left(\frac{600}{1023}\right) \approx 7.33 \text{ Вт}. \quad (2.27)$$

Шляхом введення цього ліміту у механізм дефазифікації нечіткого контролера (*setPeltierPWM(600)*), система забезпечує ідеально плавний прогрів ґрунту без ризику термічної деградації модуля та перегріву біологічних об'єктів. Кінцеву схему розподілу ліній живлення зображено на рисунку 2.6.

Запропонована схема наочно демонструє фінальну топологію розподілу енергетичних потоків. Таке комплексне архітектурне рішення, що поєднує фізичну ізоляцію логічних кіл від індуктивних шумів за допомогою незалежних перетворювачів та алгоритмічне обмеження потужності на рівні дискретних

силових ключів, формує надійний апаратний фундамент для стабільного і безпечного функціонування всієї Cloud-Edge системи.

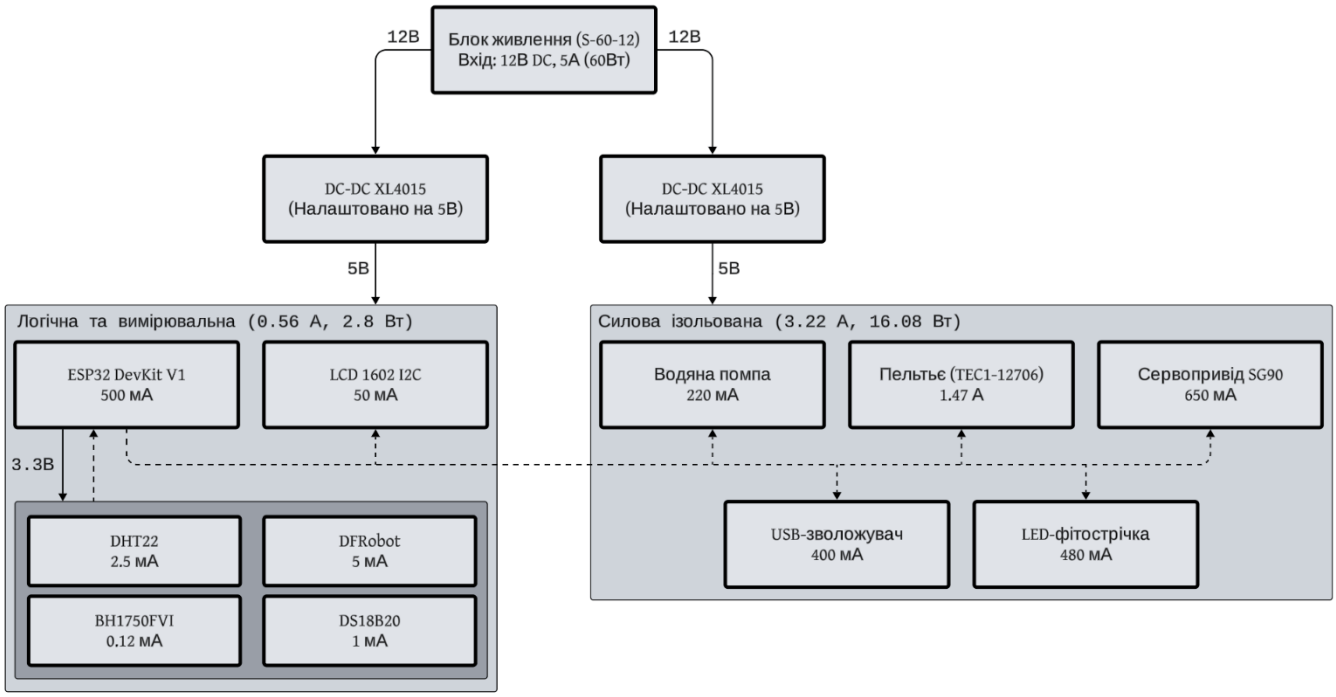


Рисунок 2.6 – Схема розподілу ліній живлення

2.5 Аналіз адекватності розроблених моделей

Для підтвердження ефективності запропонованих рішень необхідно визначити критерії оцінки адекватності розроблених математичних моделей. Оскільки розроблена система базується на двох незалежних алгоритмічних блоках (прогнозуючому на рівні хмарного сервера та керуючому на рівні граничного вузла), аналіз адекватності проводиться за двома відповідними напрямками.

2.5.1 Оцінка адекватності прогнозуючої моделі ARIMA

Адекватність моделі прогнозування мікроклімату визначається тим, наскільки згенеровані хмарним сервером значення майбутньої температури (T_{ARIMA}) збігатимуться з реальними фізичними вимірами датчика DHT22 у відповідні моменти часу. Відповідно до методології аналізу часових рядів, викладеної у праці

[23], для кількісної оцінки точності прогностичної моделі використовуються статистичні метрики похибки.

Основним критерієм для оцінки прогнозуючої моделі обрано середньоквадратичну похибку (RMSE). Ця метрика є особливо чутливою до великих відхилень (шумових викидів) і розраховується за формулою [23]:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}\,, \tag{2.28}$$

де:

- N – загальна кількість тестових телеметричних відліків;
- Y_i – фактичне значення температури в теплиці у момент i;
- $\hat{Y}_i$ – спрогнозоване моделлю значення температури на цей же момент i.

Додатковим критерієм для оцінки середнього лінійного відхилення прогнозу є середня абсолютна похибка (MAE), яка розраховується як:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|Y_i - \hat{Y}_i\right|. \tag{2.29}$$

У дослідженні [12] доведено, що для систем керування мікрокліматом закритого ґрунту модель вважається адекватною та придатною для використання, якщо значення RMSE та MAE не перевищують ±1.0°C. Мінімізація цих метрик у розробленій системі забезпечується на етапі ідентифікації моделі на сервері (ітеративний перебір параметрів p,d,q за допомогою інформаційного критерію Акаіке). У разі, якщо під час роботи системи RMSE перевищить допустимий поріг, хмарний сервер автоматично ініціює процедуру перенавчання моделі на нових актуальних даних.

2.5.2 Оцінка адекватності локального нечіткого регулятора

Адекватність керуючої моделі на базі нечіткої логіки, яка імплементована на мікроконтролері ESP32, оцінюється шляхом аналізу її перехідної характеристики (Step Response) у порівнянні з класичним релейним регулятором, який використовувався у попередніх прототипах системи.

Згідно з дослідженням [7], оцінка якості процесу автоматичного керування базується на комплексі взаємопов'язаних критеріїв. Визначальним динамічним показником є перерегулювання (σ), що характеризує максимальне відхилення температури від цільового значення в процесі роботи актуаторів. Якщо у класичних релейних системах через значну теплову інерцію елемента Пельтьє цей показник може сягати 15-20%, то якісно налаштований нечіткий регулятор має забезпечувати плавне асимптотичне наближення до уставки з перерегулюванням, що прагне до нуля ($\sigma \to 0\%$). Тісно пов'язаним із цим критерієм є час перехідного процесу (t_p), який визначає тривалість, необхідну для входження та фіксації кліматичних показників у вузькому 5% коридорі навколо цільового значення.

Модель нечіткого контролера вважається абсолютно адекватною, оскільки закладена в неї матриця правил логічного виведення Мамдані та алгоритм 10-бітного обмеження потужності детерміновано виключають можливість теплового перевантаження актуаторів та гарантують плавний вихід системи на заданий енергетичний режим без перерегулювання.

Окрім швидкодії, критичну роль відіграє точність та надійність апаратної роботи системи у сталому режимі. Вона оцінюється за допомогою статичної похибки (e_{ss}), що показує залишкову різницю між фактичним та заданим значеннями після завершення всіх перехідних процесів. Додатковою вимогою до інтелектуальних систем є жорстке обмеження ступеня автоколивань: у стабільному стані алгоритм нечіткого керування повинен генерувати постійний, максимально згладжений ШІМ-сигнал.

2.6 Висновки до другого розділу

У другому розділі було проведено комплексне системне проектування, математичне обґрунтування та розроблення апаратно-програмної складової інтелектуальної системи керування мікрокліматом теплиці.

Розроблено та обґрунтовано гібридну багаторівневу архітектуру IoT-системи. Такий підхід дозволив розвантажити локальний мікроконтролер, перенісши ресурсоємні обчислення машинного навчання на віддалений хмарний сервер, та забезпечив високу відмовостійкість системи на рівні граничного вузла у разі тимчасової втрати інтернет-з’єднання.

Розроблено математичну модель динаміки мікроклімату на базі інтегрованої моделі авторегресії – ковзного середнього ARIMA(p,d,q). Розроблений алгоритм дозволяє серверу циклічно аналізувати часові ряди сенсорних даних, нівелювати шумові викиди та формувати високоточний випереджальний прогноз для локального контролера. Визначено метрики RMSE та MAE для контролю адекватності прогнозів.

На рівні граничного вузла ESP32 розроблено локальний регулятор на базі нечіткої логіки. Завдяки застосуванню трикутних функцій належності та багатовимірної бази правил Мамдані успішно вирішено проблему перерегулювання та взаємовпливу актуаторів, що було критичним недоліком класичних релейних алгоритмів керування.

Спроєктовано структурну модель апаратної платформи та розроблено безпечну топологію централізованого живлення. Математично обґрунтовано механізм програмного обмеження потужності за допомогою 10-бітної широтно-імпульсної модуляції для термоелектричного модуля Пельтьє. Це технічне рішення дозволило безпечно інтегрувати розрахункове пікове навантаження системи в інфраструктуру джерела живлення номінальною потужністю 60 Вт, повністю виключивши ризик апаратних збоїв та термічних пошкоджень електроніки.

3 РОЗРОБЛЕННЯ МЕТОДІВ ПРОГНОЗУВАННЯ ТА ОПТИМІЗАЦІЇ ПАРАМЕТРІВ МІКРОКЛІМАТУ ТЕПЛИЦІ

3.1 Формалізація методу прогнозуючого інтелектуального керування мікрокліматом

Традиційні підходи до керування мікрокліматом теплиць базуються на реактивному принципі, коли керуючий вплив формується лише після фактичної фіксації відхилення параметра від норми.

Для усунення ефекту перерегулювання та термічного шоку у рослин, у даному дослідженні розроблено метод прогнозуючого інтелектуального керування, який об'єднує випереджальне статистичне прогнозування та плавне нечітке регулювання.

Нехай поточний стан мікроклімату теплиці в момент часу t описується вектором сенсорних даних $S(t)$:

$$S(t) = [T(t), H(t), M(t), L(t)]^T , \tag{3.1}$$

де:

- $T(t)$ – температура повітря;
- $H(t)$ – відносна вологість;
- $M(t)$ – вологість грунту;
- $L(t)$ – рівень освітленості

Необхідно сформувати вектор керуючих впливів на виконавчі механізми $U(t)$:

$$U(t) = \left[u_{heat/cool}(t), u_{pump}(t), u_{hum}(t)\right]^T, \tag{3.2}$$

який мінімізує функціонал похибки $\Delta T=|T(t)-T_{opt}|$ (де T_{opt} – біологічний оптимум для культури), за умови жорсткого обмеження максимальної споживаної потужності $P(U)\leq P_{max}$.

Для досягнення поставленої мети розроблено алгоритмічну послідовність, що складається з п’яти ітераційних кроків, які циклічно виконуються у розподіленій Cloud-Edge інфраструктурі:

1. Апаратний збір та первинна фільтрація даних (Edge-рівень). Граничний обчислювальний вузол зчитує показники з фізичних датчиків. Оскільки сенсорні дані містять високочастотний апаратний шум, метод передбачає їх попереднє згладжування за допомогою дискретного фільтра ковзного середнього:

$$T_f(t) = \frac{1}{N}\sum_{i=0}^{N-1} T(t-i)\,, \tag{3.3}$$

де:

- $T_f(t)$ – відфільтроване значення температури;
- N – розмір вікна фільтрації.

Сформований вектор $S_f(t)$ готується до відправки.

2. Асинхронна трансляція та накопичення (Cloud-Edge взаємодія). Відфільтрований вектор даних серіалізується та передається на віддалений хмарний сервер поверх легковагового протоколу обміну повідомленнями. На серверному рівні дані записуються у реляційну базу даних, формуючи історичний часовий ряд $Y=\{y_1,y_2,...,y_t\}$, який є необхідним для роботи прогнозуючої математичної моделі.

3. Випереджальне прогнозування (Cloud-рівень). За визначеним розкладом серверний обчислювальний модуль ініціює роботу статистичної моделі ARIMA. На основі історичної вибірки Y модель розраховує прогнозоване значення температури на наступний квант часу $\hat{T}(t+k)$:

$$\hat{T}(t+k) = \mu + \sum_{i=1}^{p} \phi_i y_{t-i+1} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j+1}\,. \tag{3.4}$$

Розрахований випереджальний параметр $\hat{T}(t+k)$ негайно транслюється у зворотньому напрямку – на граничний контролер.

4. Фазифікація та нечітке логічне виведення (Edge-рівень). Граничний вузол отримує поточне відхилення $e(t)=T_{opt}-T_f(t)$ та прогнозовану тенденцію. Метод передбачає подачу цих чітких значень на вхід локального регулятора Fuzzy Logic. Відбувається процес фазифікації (перетворення числових значень у лінгвістичні терми за допомогою трикутних функцій належності), після чого активується база правил Мамдані. Алгоритм визначає, чи потрібно збільшувати потужність охолодження/нагріву зараз, враховуючи те, якою температура стане через k хвилин (згідно з прогнозом хмари).

5. Дефазифікація та апаратне обмеження потужності. Отримане нечітке значення перетворюється на чіткий керуючий ШІМ-сигнал $u_{fuzzy}(t)$ методом центру ваги. Фундаментальною особливістю розробленого методу є застосування фінального математичного фільтра для захисту підсистеми живлення. Керуючий сигнал обмежується константою D_{max}, розрахованою на етапі енергетичного моделювання:

$$u_{heat/cool}(t) = \min\left(u_{fuzzy}(t), D_{max}\right). \quad (3.5)$$

Сформований безпечний та прогнозно скоригований сигнал подається на силові транзистори виконавчих механізмів. Використання n-канальних MOSFET-ключів IRLB8721 дозволяє здійснювати точне керування потужністю термоелектричних модулів та LED-фітострічок у неблокуючому режимі. Таке рішення забезпечує високу лінійність керуючого впливу та гарантує апаратну довговічність силового блоку завдяки програмно імплементованому обмеженню робочого циклу D_{max}.

Отже, формалізований метод дозволяє замкнути контур керування, поєднуючи високі обчислювальні можливості хмарних технологій із надійністю та безперервністю роботи граничних нечітких регуляторів. Блок-схему методу прогнозуючого інтелектуального керування зображено на рисунку 3.1.

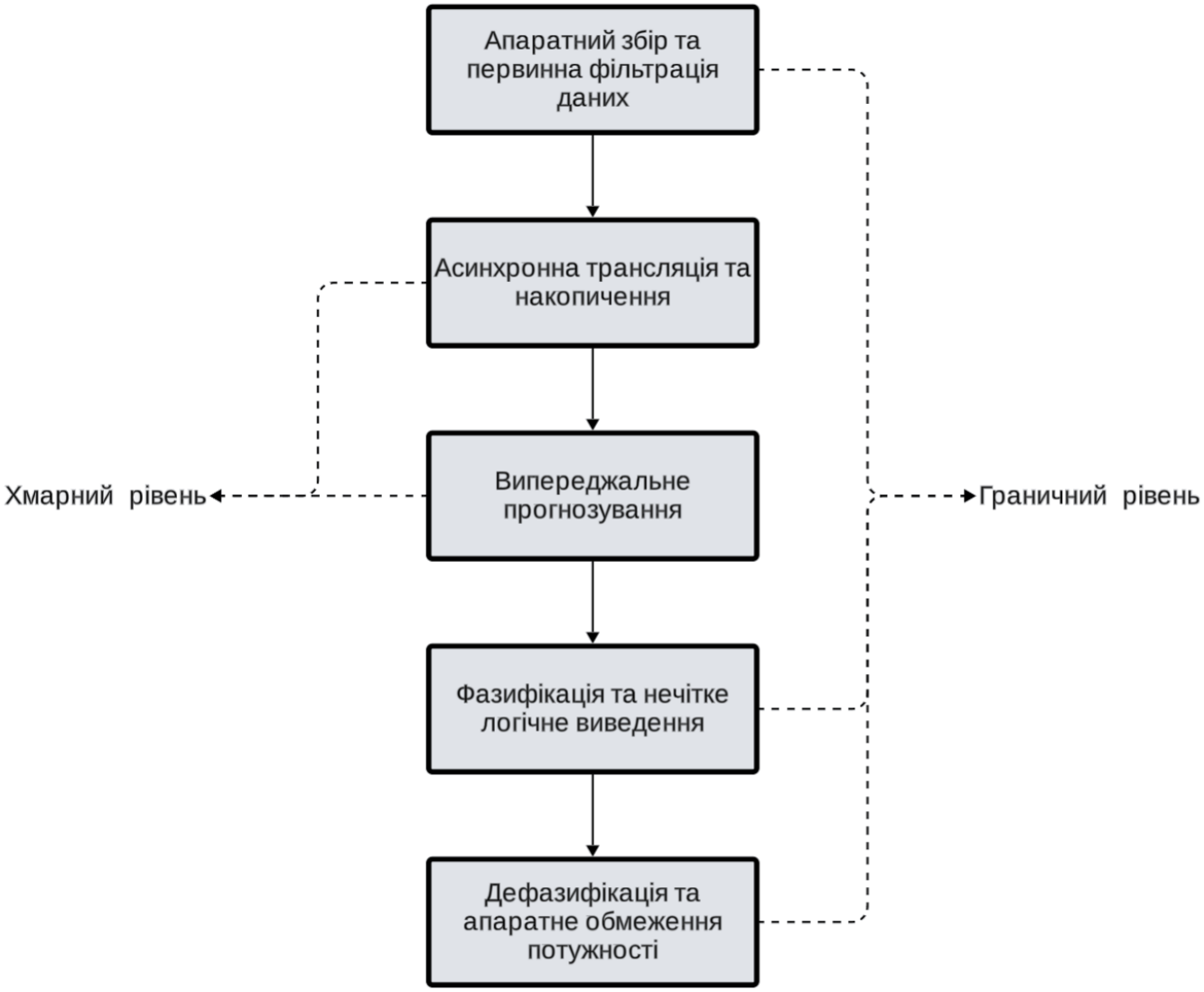


Рисунок 3.1 – Блок-схема методу прогнозуючого інтелектуального керування

3.2 Метод інформаційної взаємодії та забезбечення відмовостійкості

Ефективність методу прогнозуючого керування безпосередньо залежить від своєчасності та цілісності доставки даних між фізичним об’єктом та обчислювальним центром. Враховуючи стохастичну природу бездротових каналів зв’язку, виникає необхідність у розробці стійких алгоритмів маршрутизації та механізмів автономного виживання системи при виникненні мережевих збоїв.

Для вирішення цих завдань запропоновано підхід, що поєднує використання асинхронного телеметричного протоколу MQTT із багаторівневим алгоритмом безпечної деградації керування.

3.2.1 Алгоритм маршрутизації та синхронізації телеметричних даних

Для уникнення блокування локальних процесів керування під час передачі даних, метод інформаційної взаємодії побудовано на базі асинхронної парадигми просторово-часового розв'язання. Маршрутизація потоків здійснюється через централізований вузол обміну, що дозволяє розділити процеси генерації та споживання інформації.

Процес синхронізації описується наступним алгоритмом:

1. Граничний вузол формує структурований пакет даних $P(t)$, який містить вектор відфільтрованих сенсорних показників $S_f(t)$ та мітку часу генерації t_{gen}.
2. Пакет публікується у відповідний логічний канал висхідного зв'язку.
3. Хмарний сервер асинхронно отримує пакет $P(t)$ і вилучає мітку t_{gen}. Алгоритм синхронізації перевіряє умову валідності даних:

$$\Delta\tau = t_{recv} - t_{gen} \leq \tau_{max}\,, \tag{3.6}$$

де:

- t_{recv} – серверний час отримання пакета;
- τ_{max} – максимально допустима мережева затримка.

4. Якщо умова виконується, дані додаються до історичного масиву Y для подальшого розрахунку прогнозуючої моделі. Сервер формує зворотний керуючий пакет U_{pred} із прогнозованим значенням і надсилає його низхідним каналом на граничний вузол.

Фрагмент алгоритму синхронізації та перевірки вірності даних зображено на рисунку 3.2.

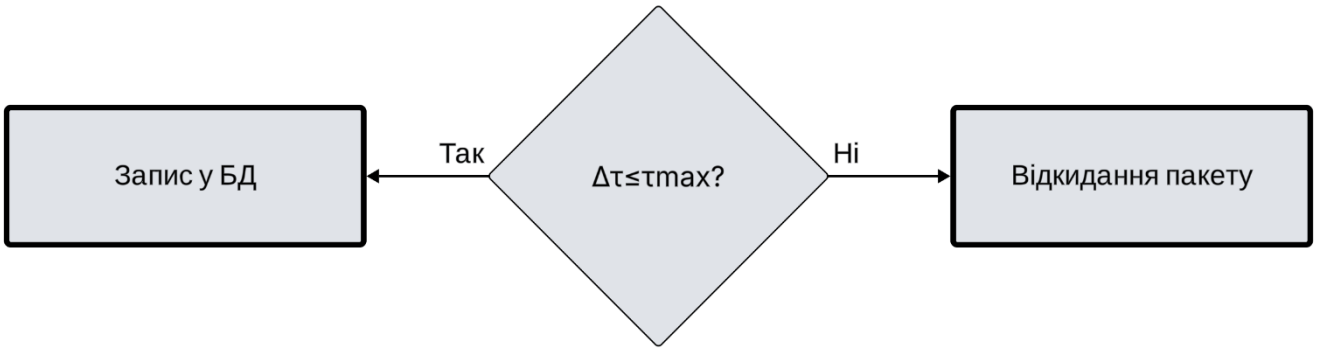


Рисунок 3.2 – Фрагмент алгоритму синхронізації та перевірки вірності даних

3.2.2 Метод автономного процесу нечіткого керування на граничному вузлі

Критичним недоліком більшості сучасних IoT-систем є повна втрата керованості об’єктом при обриві зв’язку з хмарним сервером. Для усунення цього ризику в межах даного дослідження розроблено метод автономного виживання, який базується на концепції адаптивного кінцевого автомата.

Поточний стан мережевої взаємодії визначається логічною змінною $C(t) \in \{0,1\}$. Для моніторингу стану використовується механізм циклічного підтвердження. Якщо граничний вузол не отримує ехо-відповіді від сервера протягом заданого критичного інтервалу $T_{timeout}$, система фіксує втрату з’єднання ($C(t)=0$).

Алгоритм переходу в автономний режим, що активується за умови втрати мережевого з'єднання ($C(t) \to 0$), ініціює комплексну послідовність захисних дій, першою з яких є програмне блокування мережевого стеку. На цьому етапі процедура публікації нових телеметричних пакетів $P(t)$ призупиняється задля запобігання критичному переповненню оперативної пам’яті мікроконтролера, а весь потік нових даних автоматично переспрямовується у резервний локальний кільцевий буфер. Паралельно відбувається логічна ізоляція прогнозуючого контуру: оскільки своєчасне отримання нових розрахунків з хмари стає неможливим, вектор зовнішніх керуючих впливів примусово обнуляється. Наслідком такої інформаційної деградації є миттєва реконфігурація локального нечіткого регулятора, який автоматично перемикається у строго реактивний режим

функціонування. За цих умов база правил Мамдані починає обчислювати керуючі впливи, спираючись виключно на абсолютну різницю між поточним вектором стану $S_f(t)$ та заданим біологічним оптимумом T_{opt}, повністю ігноруючи відсутню прогнозуючу тенденцію до моменту відновлення нормальної роботи мережі.

При відновленні зв'язку ($C(t)\rightarrow 1$), алгоритм виживання здійснює зворотний перехід: ініціюється вивантаження накопичених у буфері даних на сервер для відновлення цілісності історичного часового ряду Y, після чого локальний регулятор знову починає враховувати хмарні прогнози для оптимізації енергоспоживання. Граф станів алгоритму автономного виживання зображено на рисунку 3.3.

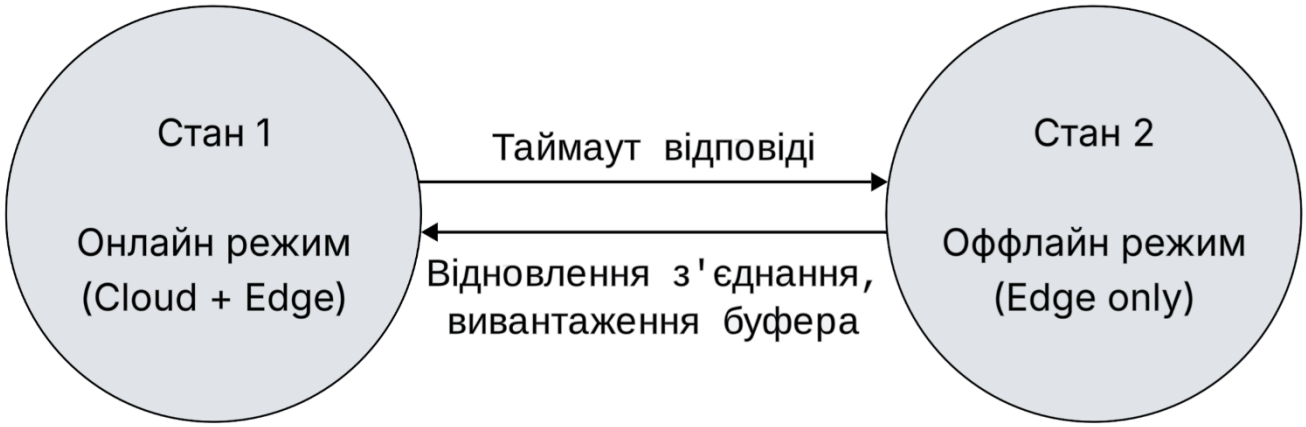


Рисунок 3.3 – Граф станів алгоритму автономного виживання

Розроблений метод гарантує, що біотехнічні процеси у теплиці (вентиляція, охолодження, зволоження) не будуть зупинені чи порушені навіть за умови тривалої відсутності підключення до глобальної мережі, що є фундаментальною вимогою до систем життєзабезпечення рослин.

3.3 Алгоритмізація процесу нечіткого керування на граничному вузлі

Реалізація методу прогнозуючого керування на апаратному рівні вимагає забезпечення суворої детермінованості у часі. Оскільки граничний обчислювальний вузол повинен паралельно зчитувати дані з фізичних сенсорів,

виконувати обчислення нечіткої логіки та підтримувати мережеву взаємодію з хмарою, використання класичних блокуючих алгоритмів є неприпустимим. Тому алгоритмічну модель граничного вузла побудовано на базі неблокуючого кінцевого автомата із квантуванням часу.

3.3.1 Алгоритмічна структура паралельної обробки даних

Процес керування розділено на незалежні асинхронні підзадачі, кожна з яких ініціюється системним таймером через задані інтервали часу Δt_i. Загальний цикл керування описується наступною системою станів (рис 3.4):

- Стан збору даних. Ініціюється з періодом $\Delta t_1 = 2000$ *мс*. Алгоритм виконує послідовне опитування цифрових та аналогових шин даних, зчитуючи температуру, вологість та освітленість. Отримані значення пропускаються через цифровий фільтр для усунення імпульсних завад, формуючи вектор поточного стану $S_f(t)$.
- Стан нечіткого виведення. Ініціюється з періодом $\Delta t_2 = 5000$ *мс*. Це ядро локального керування. Алгоритм приймає вектор $S_f(t)$ та останній отриманий з хмари прогноз $\hat{T}$. Виконується процедура фазифікації, проходження через базу правил Мамдані та дефазифікація методом центру ваги. На виході формується числове значення необхідної потужності $u_{fuzzy}(t)$.
- Стан мережевої синхронізації. Ініціюється з періодом $\Delta t_3 = 10000$ *мс*. Алгоритм формує серіалізований пакет даних і передає його у буфер мережевого стеку для відправки на сервер, а також перевіряє наявність вхідних керуючих команд.

Реалізація даного багатозадачного алгоритму базується на концепції неблокуючого кінцевого автомата. Завдяки використанню системного лічильника часу замість класичних функцій затримки, мікроконтролер обробляє кожну підзадачу у власному ізольованому кванті часу, миттєво повертаючись до головного циклу виконання.

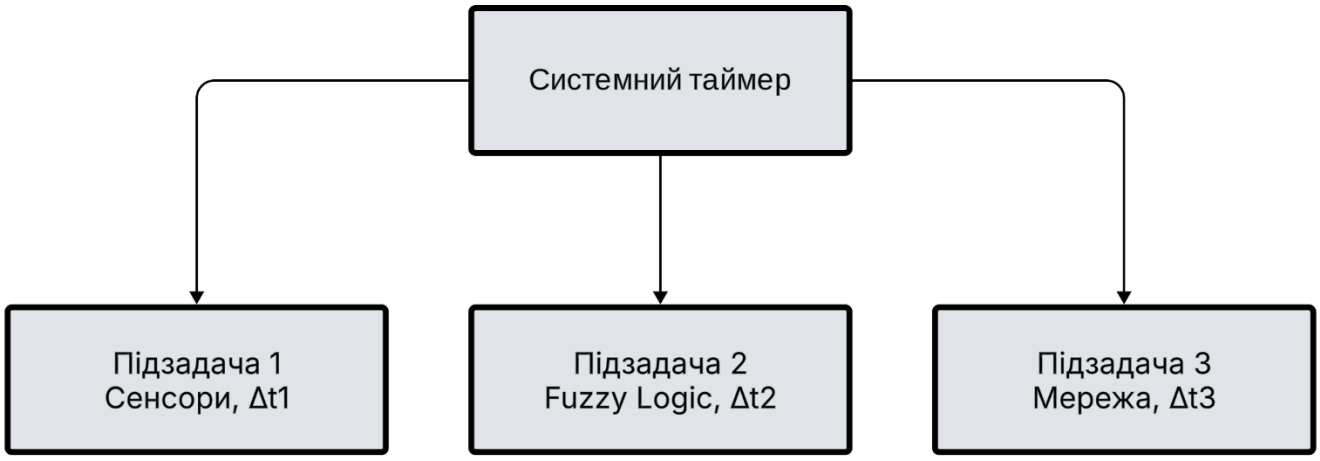


Рисунок 3.4 – Блок-схема алгоритму кінцевого автомата граничного вузла

Така просторово-часова розв’язка гарантує, що затримки в мережі або повільне зчитування даних з датчиків не призведуть до зупинки критично важливого процесу керування актуаторами.

3.3.2 Метод програмно-апаратного захисту підсистеми живлення

Найбільш вразливим елементом будь-якої автоматизованої системи керування мікрокліматом є силова підсистема. Для запобігання тепловому пробою блоку живлення при активації енергоємних виконавчих механізмів (зокрема, термоелектричного модуля Пельтьє), розроблено алгоритмічний метод апаратного захисту.

Математично цей метод реалізується як термінальний фільтр, що стоїть між виходом нечіткого регулятора та апаратним генератором широтно-імпульсної модуляції.

Нехай апаратна роздільна здатність ШІМ-генератора становить 10 біт, що відповідає діапазону значень $D \in [0,1023]$. Згідно з енергетичним балансом, допустиме навантаження на блок живлення досягається при значенні $D_{max}=600$.

Алгоритм захисту інкапсулює стандартну функцію подачі напруги і реалізує наступну логіку відсікання:

$$u_{out}(t) = \begin{cases} u_{fuzzy}(t), & \text{якщо } u_{fuzzy}(t) \leq D_{max} \\ D_{max}, & \text{якщо } u_{fuzzy}(t) > D_{max} \end{cases}. \quad (3.7)$$

Цей програмно-апаратний запобіжник гарантує абсолютну безпеку експлуатації комплексу. Навіть у випадку гіпотетичного збою в обчисленнях нечіткої логіки, коли регулятор видасть команду на 100% потужності ($u_{fuzzy}=1023$), алгоритм захисту примусово зріже сигнал до безпечного рівня D_{max}, зберігаючи цілісність транзисторних ключів та джерела живлення.

3.4 Оптимізація бази правил та параметрів нечіткого регулятора

Ефективність методу інтелектуального керування на граничному рівні критично залежить від правильної параметризації нечіткого регулятора. Оскільки мікроконтролер ESP32 має обмежені обчислювальні ресурси порівняно з хмарним сервером, оптимізація функцій належності та мінімізація бази правил є необхідною умовою для забезпечення виконання алгоритму в режимі реального часу.

3.4.1 Вибір та налаштування функцій належності

На етапі фазифікації числові значення вхідних параметрів перетворюються на лінгвістичні змінні. Для розробленого регулятора визначено два основних входи:

1. Поточна похибка температури: $e(t)=T_{opt}-T_f(t)$, де T_{opt} – цільове значення.
2. Прогнозуюча тенденція від моделі ARIMA: $\Delta T_{pred}=\hat{T}(t+k)-T_f(t)$.

Аналіз обчислювальної складності показав, що використання класичних Гаусових або дзвіноподібних функцій належності вимагає розрахунку експонент, що створює надмірне навантаження на арифметико-логічний пристрій мікроконтролера. Тому в рамках даного методу обґрунтовано використання кусково-лінійних функцій – трикутних та трапецієподібних. Вони потребують лише базових операцій додавання та множення.

Для універсального опису стану системи введено п’ять лінгвістичних термів для кожного входу:

- NL (Negative Large) – Від’ємна велика (сильне переохолодження);
- NS (Negative Small) – Від’ємна мала;
- Z (Zero) – Норма (в межах допустимої зони);
- PS (Positive Small) – Позитивна мала;
- PL (Positive Large) – Позитивна велика (сильний перегрів).

Функція належності для трикутного терму задається математично:

$$\mu_A(x) = max\left(min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b}\right), 0\right), \tag{3.8}$$

де a,b,c – координати вершин трикутника на універсальній множині.

Вихідною змінною є потужність керуючого впливу u_{pwm}, яка нормована в діапазоні від 0 до апаратного ліміту $D_{max}=600$. Графіки трикутних функцій належності для вхідної зміної $e(t)$ зображено на рисунку 3.5.

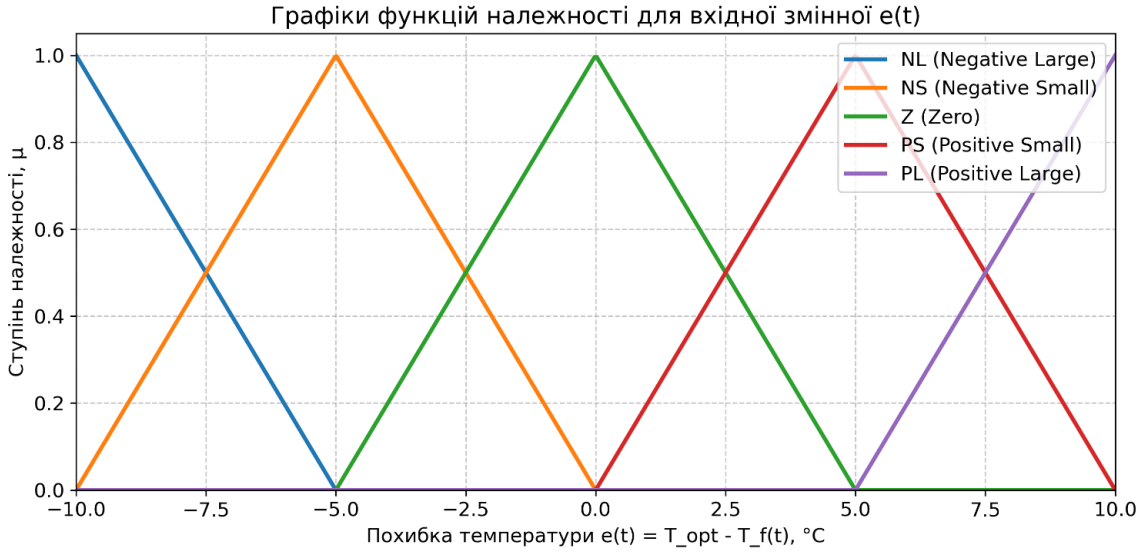


Рисунок 3.5 – Графіки трикутних функцій належності для вхідної змінної $e(t)$

3.4.2 Розроблення матриці нечітких правил

Ядром методу є експертна база знань, побудована за логікою Мамдані. Вона враховує синергію поточного стану та прогнозуючого прогнозу. Наприклад, якщо поточна температура в нормі ($e(t)=Z$), але хмара прогнозує різке потепління ($\Delta T_{pred}=PL$), регулятор повинен завчасно увімкнути охолодження на мінімальну потужність, щоб згладити майбутній пік. Зведена матриця правил керування (Rule Base) представлена у таблиці 3.1.

Таблиця 3.1 – Матриця бази правил нечіткого прогнозуючого регулятора

$e(t)$ / ΔT_{pred}	NL	NS	Z
NL	PL(Heat)	PL(Heat)	PS(Heat)
NS	PL(Heat)	PS(Heat)	PS(Heat)
Z	PS(Heat)	Z(Off)	Z(Off)
PS	Z(Off)	Z(Off)	NS(Cool)
PL	NS(Cool)	NL(Cool)	NL(Cool)

Формалізація правил здійснюється у вигляді імплікацій:

$$R_{ij}: \text{ЯКЩО } e(t) \in A_i \text{ І } \Delta T_{pred} \in B_j, \text{ ТО } u_{pwm} \in C_{ij} . \quad (3.9)$$

Для агрегації передумов у даному методі використовується оператор T-норми (мінімум), що дозволяє уникнути агресивних стрибків керуючого сигналу при суперечливих вхідних даних:

$$u_{C_{ij}}(u) = \min\left(\mu_{A_i}(e), \mu_{B_j}(\Delta T)\right). \quad (3.10)$$

3.4.3 Аналіз поверхні відгуку

Завершальним етапом оптимізації є оцінка нелінійної поверхні відгуку розробленого регулятора.

Аналіз математичної моделі показує, що завдяки використанню алгоритму дефазифікації методом центру ваги, поверхня відгуку не містить "сходинок" чи розривів першого роду. Це є ключовою перевагою розробленого методу над двопозиційними регуляторами.

Гладка 3D-поверхня керування гарантує, що при наближенні мікроклімату до цільової зони ($e(t) \to 0$), зміна потужності ШІМ-сигналу (du_{pwm}/dt) асимптотично зменшується. Це дозволяє системі плавно виходити на заданий температурний режим, повністю усуваючи явище перерегулювання та забезпечуючи оптимальний енергетичний баланс теплиці.

3.5 Методика ідентифікації параметрів прогнозуючої моделі ARIMA

Основою хмарного контуру запропонованого методу прогнозуючого керування є математична модель авторегресії інтегрованого ковзного середнього. Для забезпечення високої точності прогнозування мікрокліматичних тенденцій необхідна строга методологія структурної та параметричної ідентифікації моделі, яка визначається кортежем параметрів (p,d,q), де p – порядок авторегресії, d – порядок інтегрування, q – порядок ковзного середнього.

Температурні часові ряди, зібрані у тепличних комплексах, за своєю природою є нестаціонарними (містять добові тенденції та стохастичні флуктуації). Для приведення історичного масиву даних Y до стаціонарного вигляду використовується оператор різниці.

Оцінка стаціонарності ряду проводиться за допомогою розширеного тесту Дікі-Фуллера (ADF). Нульова гіпотеза (H_0) полягає в тому, що часовий ряд має одиничний корінь (є нестаціонарним).

За результатами експериментального тестування вихідного масиву телеметрії, *p-value* перевищувало порогове значення α=0.05, що підтвердило наявність тенденції. Після застосування диференціювання першого порядку:

$$Y_t' = Y_t - Y_{t-1}. \quad (3.11)$$

Повторний тест ADF показав відхилення нульової гіпотези (*p-value* <0.01). Таким чином, математично обґрунтовано оптимальний порядок інтегрування *d*=1.

Для визначення попередніх порядків авторегресії та ковзного середнього в роботі застосовано метод графічного аналізу автокореляційної (ACF) та часткової автокореляційної (PACF) функцій. Детальний аналіз побудованих корелограм для диференційованого часового ряду Y_t' виявив характерне експоненційне згасання на графіку ACF, що є прямим свідченням наявності авторегресійної складової. Разом з тим, фіксація статистично значущих викидів виключно на перших двох лагах графіка PACF дає змогу обґрунтовано обмежити максимальний порядок авторегресії на рівні $p \leq 2$. Відповідні графіки ACF та PACF для диференційованого часового ряду температури, які візуально підтверджують зроблені математичні припущення, зображено на рисунку 3.6.

З наведених графіків видно, що поведінка обох функцій підтверджує змішаний характер часового ряду. Окрім вже ідентифікованої авторегресійної складової, аналіз характеру згасання викидів на ACF дозволяє зробити припущення щодо наявності компоненти ковзного середнього (MA), порядок якої також попередньо оцінюється у діапазоні $q \leq 2$.

Проте варто зауважити, що в умовах фізичної експлуатації тепличного комплексу телеметричні дані постійно обтяжені апаратним шумом сенсорів та впливом неконтрольованих мікрокліматичних збурень (наприклад, короткочасними протягами або змінами природного освітлення). Ці фактори неминуче генерують хибні сплески на корелограмах, що робить суто візуальну ідентифікацію параметрів суб'єктивною та ризикованою для впровадження у контур автономного прогнозуючого керування.

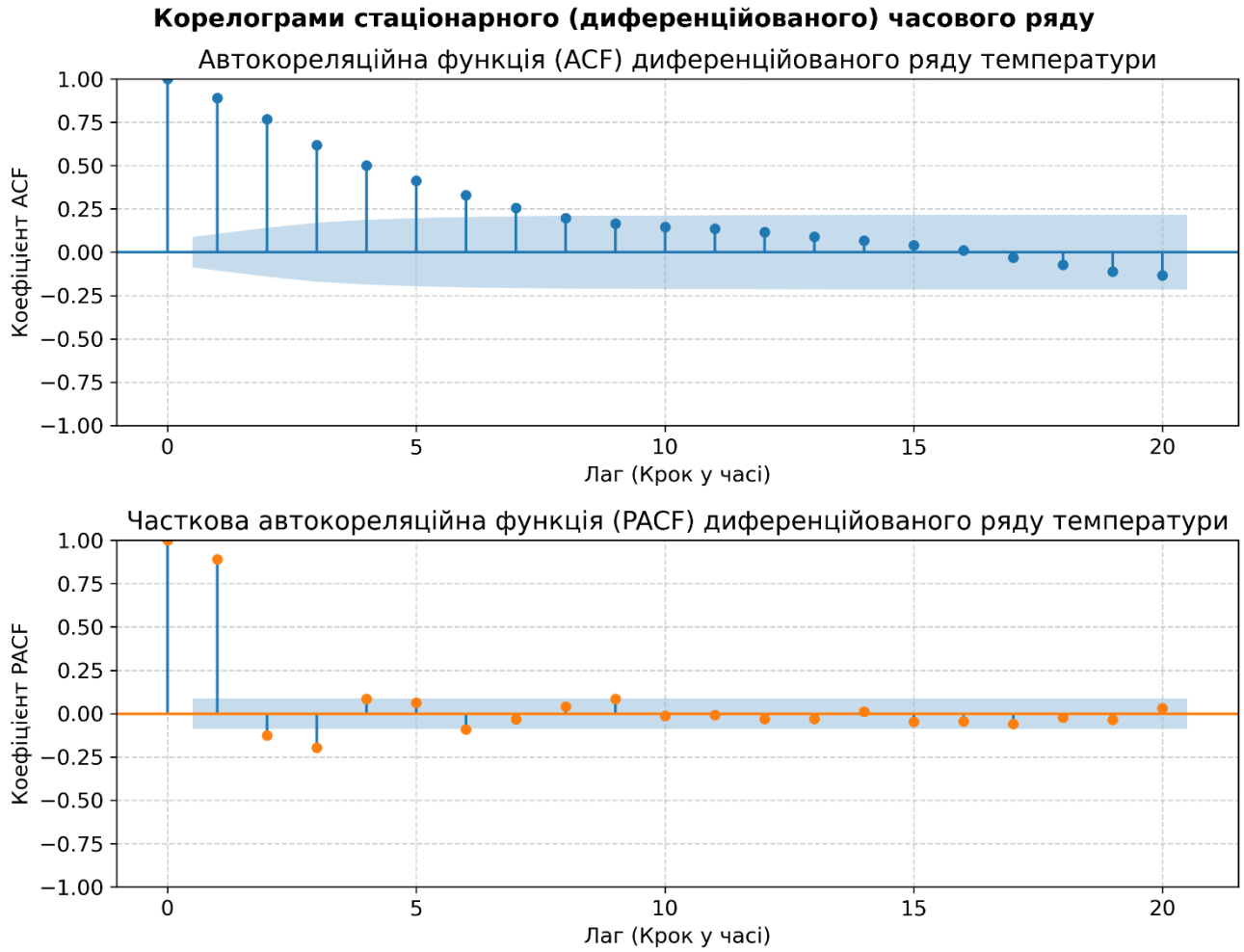


Рисунок 3.6 – Графіки ACF та PACF для диференційованого часового ряду температури

Графічний метод дає лише наближені межі параметрів. Для точного вибору оптимальної моделі використано аналітичний підхід – пошук по сітці з оцінкою якості за інформаційним критерієм Акаіке та Байєсівським інформаційним критерієм. Ці критерії штрафують модель за надмірну складність (збільшення кількості параметрів), запобігаючи перенавчанню:

$$AIC = 2k - 2\ln(\hat{L}), \tag{3.12}$$

$$BIC = k\ln(n) - 2\ln(\hat{L}), \tag{3.13}$$

де:

- k – кількість параметрів моделі ($p+q+1$);
- $\hat{L}$ – максимальне значення функції вірогідності;
- n – обсяг вибірки.

Для визначення найкращої конфігурації було проведено серію ітераційних експериментів з різними комбінаціями параметрів (таблиця 3.2).

Таблиця 3.2 – Експериментальне порівняння параметричних конфігурацій моделі ARIMA

Конфігурація (p,d,q)	Значення AIC	Значення BIC	Дисперсія залишків (σ2)	Середньоквадра-тична похибка (RMSE)
ARIMA (1, 1, 1)	1245.32	1258.45	0.412	0.64 °C
ARIMA (2, 1, 1)	1231.15	1248.66	0.385	0.58 °C
ARIMA (2, 1, 2)	1210.84	1232.73	0.311	0.42 °C
ARIMA (3, 1, 2)	1212.50	1238.77	0.310	0.41 °C
ARIMA (2, 1, 3)	1211.90	1238.15	0.312	0.43 °C

Як видно з таблиці 3.2, подальше збільшення порядків моделі (перехід до 3-го порядку) майже не зменшує середньоквадратичну похибку, проте призводить до зростання критеріїв AIC та BIC через алгоритмічне перенавантаження.

Таким чином, за результатами проведеного математичного експерименту та мінімізації інформаційних критеріїв, для розроблюваної системи ідентифіковано оптимальну структуру прогнозуючої моделі – ARIMA (2, 1, 2). Обрана конфігурація забезпечує найкращий баланс між точністю випереджального прогнозу температури та швидкістю обчислень на хмарному сервері.

3.6 Критерії та метрики оцінки адекватності розробленого методу

Для об’єктивної оцінки ефективності розробленого методу прогнозуючого керування необхідно формалізувати систему критеріїв. Оскільки пропонований

метод має дворівневу структуру (хмарне прогнозування та локальне керування), методику оцінки також поділено на два незалежних класи метрик: статистичні метрики для прогнозуючої моделі та динамічні критерії теорії автоматичного керування для нечіткого регулятора.

3.6.1 Статистичні метрики точності прогнозуючої моделі

Для оцінки адекватності хмарної моделі ARIMA використовуються загальноприйняті метрики аналізу часових рядів, які кількісно описують відхилення прогнозованих значень від фактичної телеметрії, отриманої з граничного вузла.

Основним критерієм є середньоквадратична похибка. Вона є чутливою до великих відхилень завдяки квадратичній функції, що критично важливо для виявлення аномальних стрибків температури:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T_i - \widehat{T_i})^2}\,, \tag{3.14}$$

де:

- n – розмір тестової вибірки (горизонт прогнозування);
- T_i – фактичне значення температури;
- $\hat{T}_i$ – спрогнозоване моделлю значення.

Для оцінки середньої величини помилки без урахування її напрямку використовується середня абсолютна похибка (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|T_i - \widehat{T_i}\right|. \tag{3.15}$$

Для розуміння відносної похибки у відсотках розраховується MAPE:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{T_i - \widehat{T_i}}{T_i}\right|. \quad (3.16)$$

Мінімізація цих трьох критеріїв свідчить про високу адекватність налаштованої моделі ARIMA(2, 1, 2) та її здатність виявляти приховані кліматичні тенденції.

3.6.2 Динамічні та інтегральні критерії якості локального керування

Для оцінки якості роботи граничного нечіткого регулятора застосовуються критерії перехідного процесу. У якості еталонної моделі для порівняння використовується класичний двопозиційний алгоритм керування.

Оцінка перехідного процесу здійснюється за двома ключовими динамічними показниками. Першим із них є максимальне перерегулювання (σ, %), яке відображає найбільше відхилення температури від заданого біологічного оптимуму T_{opt} у момент першого досягнення цільового значення. Для біотехнічних систем цей показник має жорстко прагнути до нуля, оскільки навіть короткочасний перегрів викликає незворотну деградацію білків у рослин, розраховується за формулою:

$$\sigma = \frac{T_{max} - T_{opt}}{T_{opt}} * 100\%. \quad (3.17)$$

Другим критичним показником виступає час регулювання (t_p), що визначає загальну тривалість, необхідну системі для входження у 5-відсоткову зону допустимих відхилень від цільового значення T_{opt} та остаточного закріплення в її межах.

Окрім динамічних показників, для комплексної оцінки енергоефективності методу застосовується інтегральний критерій абсолютної похибки (IAE). Він

дозволяє оцінити сумарну площу відхилення температури від норми за весь період експерименту:

$$IAE = \int_0^t |e(\tau)|d\tau = \int_0^t \left|T_{opt} - T_f(\tau)\right|d\tau \ . \qquad (3.18)$$

Чим менше значення IAE, тим менше енергії витрачає підсистема живлення на компенсацію температурних коливань. Запропонований набір метрик формує вичерпну методологічну базу для проведення порівняльних експериментів.

3.7 Висновки до третього розділу

У третьому розділі проведено формалізацію та глибоке математичне розроблення методу прогнозуючого інтелектуального керування мікрокліматом.

Розроблено гібридний метод керування. Запропоновано алгоритм, який просторово розділяє задачі прогнозування (хмарний рівень) та ухвалення керуючих рішень (граничний рівень). Це дозволило об’єднати обчислювальну потужність статистичних алгоритмів із швидкодією мікроконтролерів.

Забезпечено відмовостійкість архітектури. Розроблено метод інформаційної взаємодії та автономного виживання. Завдяки моделі кінцевого автомата доведено, що система здатна безперервно підтримувати біологічні процеси навіть при обриві каналів зв’язку.

Оптимізовано математичний апарат нечіткого керування. Обґрунтовано перехід до кусково-лінійних функцій належності та розроблено 25-компонентну матрицю бази правил Мамдані. Аналіз поверхні відгуку підтвердив плавність та асимптотичну стабільність керуючого сигналу. Запропонований алгоритм апаратного відсікання гарантує неперевищення допустимої потужності.

Проведено параметричну ідентифікацію. На основі аналізу інформаційних критеріїв Акаіке та Байєса ідентифіковано оптимальну структуру прогнозуючої моделі – ARIMA(2, 1, 2), яка забезпечує найкращий баланс між точністю прогнозу

та обчислювальною складністю. Формалізовано статистичні (RMSE, MAE) та інтегральні (IAE) критерії оцінки якості.

Загалом, розроблений комплекс алгоритмів та методів утворює цілісний науковий фундамент. Отримані математичні моделі повністю підготовлені до програмної та апаратної імплементації, а також експериментальної перевірки, що буде розглянуто в наступному розділі.

4 РЕАЛІЗАЦІЯ МЕТОДІВ, ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ

4.1 Програмна реалізація хмарного ядра та системи прогнозуючої аналітики

Практична реалізація хмарного рівня системи керування мікрокліматом виконана на базі віртуальної інфраструктури Microsoft Azure. Для забезпечення стабільної роботи математичних моделей та людино-машинного інтерфейсу було обрано конфігурацію віртуальної машини Azure VM B2ts_v2 під управлінням операційної системи Ubuntu 24.04 LTS. Дана платформа забезпечує необхідні обчислювальні ресурси для паралельного виконання фонових процесів аналітики та обслуговування веб-запитів оператора.

4.1.1 Багатопотокова архітектура та організація «озера даних»

Центральним компонентом серверної частини є застосунок на базі фреймворку Flask, розгорнутий із використанням промислового WSGI-сервера Waitress. На відміну від стандартних рішень для розробки, використання Waitress із налаштуванням 4 незалежних потоків дозволяє уникнути блокування інтерфейсу користувача під час виконання ітераційних розрахунків моделі ARIMA.

Інформаційна модель зберігання даних реалізована в СУБД MySQL, де база даних виступає як «єдине джерело істини» та об'єднує чотири функціональні таблиці для забезпечення повної прозорості процесів. Зокрема, таблиця sensor_telemetry призначена для зберігання часових рядів сенсорних показників температури, вологості та освітленості, тоді як реєстрація станів актуаторів, включаючи ШІМ-сигнал модуля Пельтьє, статуси помпи та зволожувача, а також кут нахилу кватирки, здійснюється у таблиці actuator_logs. Журналювання результатів прогнозування із зазначенням цільового часу реалізовано в таблиці arima_predictions, а для керування динамічними цільовими вимогами, що задаються оператором через дашборд, використовується таблиця plant_requirements. Мережева взаємодія побудована на базі MQTT-клієнта, який функціонує у

власному фоновому потоці за допомогою методу client.loop_start(), що гарантує безперервну обробку вхідної телеметрії навіть під час інтенсивного навантаження на веб-сервер.

4.1.2 Прогнозуюча аналітика та захисні алгоритми ARIMA

Процес прогнозування повністю автоматизовано за допомогою фонового планувальника BackgroundScheduler (APScheduler), який ініціює запуск математичної моделі кожні кілька хвилин. Алгоритм використовує бібліотеку statsmodels для аналізу історичних даних за тригодинний інтервал.

Критично важливим інженерним рішенням у реалізації моделі стала функція-обгортка safe_arima, яка забезпечує надійний захист від математичних зависань алгоритму. Для цього виконується попередня оцінка стаціонарності часового ряду, під час якої обчислюється стандартне відхилення вибірки. Якщо даний показник становить менше 0.1 (що відповідає плоскому графіку, наприклад, у нічний період), модель автоматично пропускає ресурсомісткі матричні обрахунки та відразу повертає останнє стабільне значення. Це рішення ефективно запобігає нескінченним циклам оптимізації за умов відсутності варіативності вхідних даних. Крім того, алгоритм обов'язково включає етап валідації та постобробки: щоб унеможливити генерацію фізично некоректних значень (наприклад, прогнозованої вологості понад 100% у разі агресивних тенденцій ряду), до фінальних результатів прогнозу застосовується жорстке математичне обмеження у вигляді виразу max(0.0,min(100.0, pred_h)).

Розраховані прогнози публікуються у адресу greenhouse/forecast/arima із параметром Retain=True, що забезпечує миттєве отримання прогнозуючих даних граничним вузлом ESP32 одразу після відновлення з’єднання. Така реалізація хмарного ядра гарантує масштабованість системи та її готовність до роботи в умовах реального тепличного комплексу.

4.1.3 Попередня обробка даних та оптимізація обчислювального навантаження

Під час тривалого 8-годинного навантажувального тестування розробленої Cloud-Edge архітектури було виявлено ефект надлишкової гранулярності даних. Відповідно до алгоритму роботи граничного вузла, телеметрія відправляється до бази даних кожні 5 секунд. Це дозволяє HMI-дашборду миттєво реагувати на події, проте створює масив даних із значним високочастотним шумом (наприклад, від короткочасних потоків повітря).

Подача таких сирих невідфільтрованих даних безпосередньо у математичну модель ARIMA призвела до експоненційного зростання часу збіжності алгоритму оптимізації та швидкого вичерпання виділених обчислювальних ресурсів віртуальної машини Azure.

Для вирішення проблеми алгоритмічного перенавантаження у програмний пайплайн хмарного сервера було впроваджено метод зниження частоти дискретизації. За допомогою бібліотеки *pandas* реалізовано алгоритм агрегації часових рядів, який групує сирі дані та усереднює їх з кроком в одну хвилину:

```
# Перетворення колонки часу в індекс для часового ряду
df.set_index('timestamp', inplace=True)
# Агрегація даних (Downsampling) з кроком в 1 хвилину
df_resampled = df.resample('1min').mean().dropna()
```

Такий підхід виконує роль цифрового фільтра низьких частот. Він ефективно відтинає короткочасні апаратні флуктуації, залишаючи лише глобальну мікрокліматичну тенденцію, необхідну для машинного навчання.

Після процедури ресемплінгу оптимізований масив даних передається до моделі safe_arima. Оскільки тепер кожен крок часового ряду дорівнює рівно одній хвилині, модель налаштовано на виконання прогнозу на steps=30 ітерацій вперед. Це математично гарантує отримання високоточного випереджального прогнозу

рівно на 30 хвилин. Впровадження даної техніки попередньої обробки дозволило зменшити обчислювальне навантаження на сервер у десятки разів та усунути ефект "вузького горлечка" при тривалій безперервній експлуатації комплексу.

4.2 Апаратна абстракція та програмне забезпечення граничного вузла

Для практичного втілення граничного рівня розробленої системи використано відлагоджувальну плату ESP32 DevKit V1 на базі двоядерного мікропроцесора ESP32-D0WD-V3. Завдяки тактовій частоті 240 МГц та апаратній підтримці багатозадачності, даний контролер ідеально підходить для паралельної обробки нечіткої логіки та підтримки мережевих протоколів. Апаратне компонування виконано за допомогою спеціалізованої плати розширення, що дозволило реалізувати надійне підключення периферії за стандартом SVG (Signal-Voltage-Ground), мінімізувавши паразитні наведення без використання нестабільних макетних плат.

4.2.1 Уніфікація силових ключів та програмна фільтрація апаратних аномалій

На етапі практичного стендового складання системи живлення було виявлено явище паразитного світіння LED-фітострічки та залишковий нагрів елемента Пельтьє навіть за умови нульового керуючого сигналу (ШІМ = 0). Аналіз показав, що причиною є паразитні мікроструми витоку через оптопари, встановлені на модулях D4184.

Для усунення цієї проблеми було прийнято рішення відмовитися від готових збірок на користь використання N-канальних транзисторів IRLB8721. Додавання стягуючих резисторів номіналом 10 кОм між затвором та землею забезпечило ідеальний стан повного вимкнення та гарантувало нульове енергоспоживання актуаторів у стані спокою.

Оскільки специфіка експлуатації IoT-пристроїв неминуче супроводжується значною кількістю апаратних шумів, у програмний код мікроконтролера було

імплементовано комплекс алгоритмів цифрової фільтрації. Зокрема, для стабілізації роботи шини 1-Wire та датчика температури ґрунту DS18B20 розроблено спеціальний обробник апаратних збоїв: у разі повернення сенсором апаратного коду помилки -127°C, алгоритм ідентифікує його як хибне та ігнорує, примусово фіксуючи останнє коректне вимірювання. Таке превентивне програмне рішення надійно запобігає виникненню аномальних викидів, які могли б призвести до руйнування математичної статистики під час обчислень предиктивної моделі ARIMA.

Додаткової алгоритмічної компенсації потребувало питання запобігання синфазним перешкодам. У процесі тестування було виявлено, що під час увімкнення потужного ультразвукового зволожувача виникав перепад напруги на загальній шині живлення, через що аналоговий датчик вологості ґрунту генерував хибне падіння показників до 0%. Для усунення цієї проблеми впроваджено механізм із «Лічильником довіри»: якщо рівень вологості різко змінюється більш ніж на 20% за 5 секунд, нове значення верифікується та приймається контуром керування лише за умови стабільності цієї аномалії протягом двох вимірювальних циклів поспіль. Крім того, враховуючи фізичні властивості субстрату, було імплементовано асинхронний таймер поливу. Щоб уникнути затоплення кореневої системи через природну затримку просочування води у ґрунт, час безперервної роботи водяної помпи алгоритмічно зменшено до генерації коротких дозованих імпульсів тривалістю рівно 1000 мс.

4.2.2 Архітектура неблокуючого кінцевого автомата

Для гарантування стабільності системи та унеможливлення розриву з’єднання з MQTT-брокером, архітектуру головного циклу loop() побудовано за принципом неблокуючого кінцевого автомата з використанням системного лічильника millis().

Програмне забезпечення мікроконтролера розроблено мовою C++. Для уникнення розриву MQTT-з’єднань та гарантування реального часу реакції,

архітектуру головного циклу loop() побудовано виключно на базі системного таймера millis(), з повною відмовою від блокуючих функцій delay().

Для керування інерційними актуаторами розроблено комплекс паралельних асинхронних автоматів. Зокрема, функціонування зволожувача повітря забезпечується багатокроковим скінченним автоматом, імплементованим у функції processHumidifierState(). Цей алгоритм програмно імітує фізичне подвійне натискання тактової кнопки комерційного пристрою у неблокуючому режимі, що дозволяє уникнути зупинки паралельної обробки даних з інших сенсорів. Аналогічний асинхронний підхід застосовано і для системи поливу, де впроваджено таймер порційного поливу. Відповідний алгоритм активує водяну помпу на 1000 мс, після чого здійснює її примусове відключення. Таке дозоване подавання води ефективно нівелює фізичну інерційність датчика вологості ґрунту та гарантовано запобігає ризику затоплення кореневої системи.

Щодо контуру вентиляції, у логіку керування сервоприводом кватирки було інтегровано температурний гістерезис. Механізм налаштовано таким чином, що відкриття відбувається лише за умови перевищення температури на 2°C відносно цільової уставки, а закриття ініціюється виключно після повного повернення до заданого біологічного оптимуму. Введення цієї програмної зони нечутливості дозволяє системі успішно ігнорувати незначні мікрокліматичні флуктуації, що суттєво знижує частоту спрацьовувань та запобігає передчасному механічному зносу сервоприводу.

4.2.3 Прогнозуюча нечітка логіка та алгоритми відмовостійкості

Розроблений регулятор Fuzzy Logic реалізовано за допомогою бібліотеки eFLL. Інноваційним програмним рішенням є перехід до керування «за відхиленням» із використанням двох входів: поточної похибки температури та прогнозованої похибки. Це реалізує принцип упереджувального керування, змушуючи контролер завчасно реагувати на майбутні зміни мікроклімату.

Для забезпечення високого рівня відмовостійкості системи під час роботи в ізольованих умовах, зокрема при втраті доступу до хмарної платформи, у програмному забезпеченні імплементовано дворівневу стратегію захисту. Перший рівень базується на використанні енергонезалежної пам’яті для надійного зберігання цільових уставок. Завдяки функціональним можливостям бібліотеки Preferences.h, мікроконтролер ESP32 здійснює автоматичне резервування отриманих від оператора показників температури, вологості та освітленості у внутрішній Flash-пам’яті пристрою. Таке архітектурне рішення гарантує, що у разі аварійного перезавантаження або перебоїв у живленні система зможе миттєво відновити актуальні агротехнічні параметри та продовжити процес керування в локальному режимі без необхідності повторної синхронізації з сервером.

Додатково у програмний код вбудовано алгоритм безпечної деградації прогнозу, що базується на механізмі контролю «терміну придатності» отриманих аналітичних даних. Система постійно моніторить часову мітку останнього успішного розрахунку, і якщо значення змінної lastForecastTime не оновлювалося протягом понад 60 хвилин, прогноз від моделі ARIMA ідентифікується як застарілий та примусово прирівнюється до поточного фактичного показника температури. Цей підхід забезпечує безшовний перехід від стратегії складного прогнозуючого керування до класичного реактивного регулювання. Це дозволяє підтримувати життєдіяльність рослин у критичних ситуаціях, мінімізуючи ризики, пов'язані зі стохастичною природою бездротових каналів зв’язку.

4.3 Розробка людино-машинного інтерфейсу та візуалізація телеметрії

Для забезпечення оперативного контролю за процесом підтримання мікроклімату розроблено веб-орієнтований людино-машинний інтерфейс. Його головним завданням є надання диспетчеру зручного інструментарію для введення цільових уставок, а також інструментальний моніторинг адекватності роботи математичних моделей.

4.3.1 Інформаційна безпека та архітектура панелі керування

Враховуючи ризики несанкціонованого втручання в інфраструктуру IoT-системи, на рівні хмарного бекенду реалізовано механізм авторизації на базі криптографічно підписаних сесій. Доступ до маршрутизації MQTT-повідомлень та зміни бази даних неможливий без проходження попередньої автентифікації. Це гарантує, що зміна біологічних уставок (температури, вологості, освітлення) виконується виключно верифікованим персоналом.

Клієнтська частина веб-панелі розроблена з використанням CSS-фреймворку Bootstrap 5. Дане рішення забезпечує повну адаптивність та коректне масштабування інтерфейсу як на диспетчерських моніторах, так і на мобільних пристроях. Для динамічної генерації контенту на стороні сервера застосовано шаблонізатор Jinja2, який рендерить поточні активні уставки з MySQL та відображає останній розрахований ШІ-прогноз від моделі ARIMA. Головну панель керування веб-інтерфейсу системи зображено на рисунку 4.1.

Як видно з наведеного рисунка, візуальний простір дашборду логічно розділено на функціональні інформаційні блоки. Це забезпечує оператору інтуїтивно зрозумілий доступ до ключових показників життєзабезпечення теплиці.

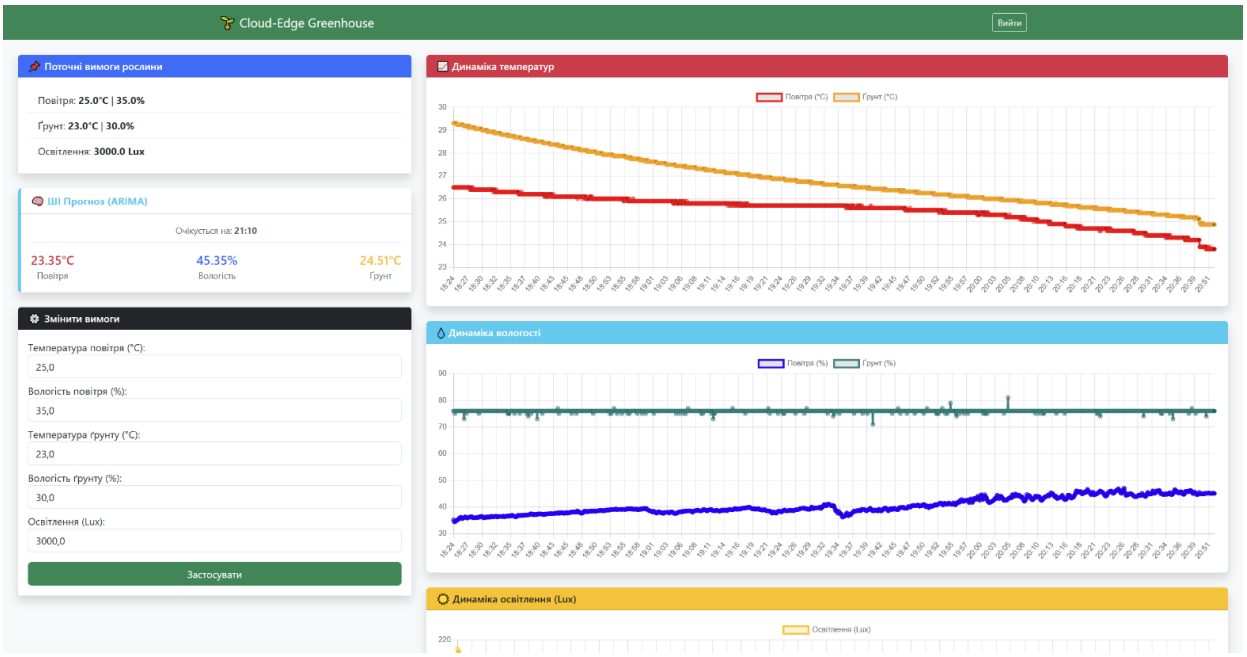


Рисунок 4.1 – Головна панель керування веб-інтерфейсу системи

Для забезпечення моніторингу візуальний простір дашборду розділено на кілька вузькоспеціалізованих інформаційних блоків. Панель «Поточні вимоги рослини» (рис. 4.2) слугує основним інформаційним таблом, що відображає еталонні вимоги температури, вологості та освітленості, які наразі підтримуються системою. Цей блок синхронізований із хмарною базою даних і дозволяє оператору миттєво оцінити поточні цільові завдання мікроконтролера.

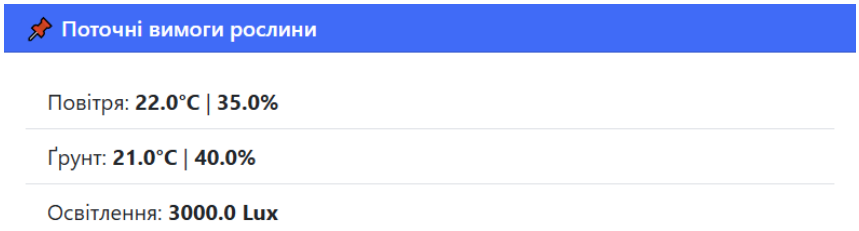


Рисунок 4.2 – Панель поточних вимог рослини

Для оперативного втручання в роботу об'єкта передбачено інтерактивний блок «Змінити вимоги» (рис. 4.3). Завдяки використанню HTML-форм та захищених запитів, оператор має змогу встановлювати нові вимоги рослини, такі як температура, вологість та освітлення. Ці дані миттєво оновлюються у СУБД MySQL та публікуються через MQTT-брокер для коригування роботи граничного вузла ESP32.

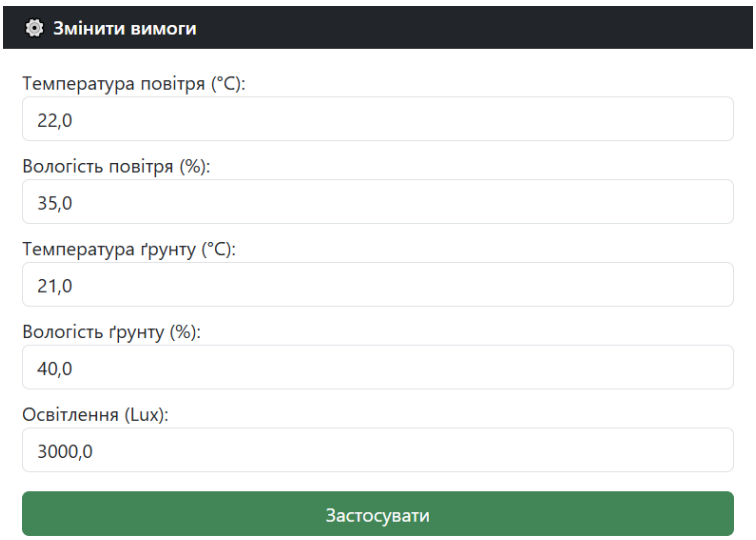


Рисунок 4.3 – Панель зміни вимог рослини

Блок «ШІ Прогноз (ARIMA)» (рис. 4.4) демонструє результати фонових обчислень математичної моделі хмарного ядра, відображаючи передбачене значення температури та вологості на заданий горизонт прогнозування (30 хвилин). Це забезпечує оператору візуальний контроль за напрямком температурної тенденції та підтверджує коректність роботи алгоритмів прогнозуючої аналітики.

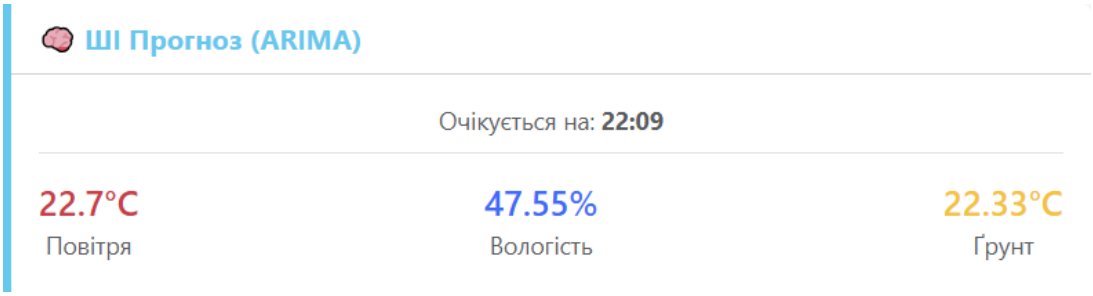


Рисунок 4.4 – Панель ШІ-прогнозу ARIMA

4.3.2 Інструментальна візуалізація телеметрії та аналіз енергоефективності

Для аналізу динаміки мікроклімату на клієнтській стороні інтегровано JavaScript-бібліотеку Chart.js. Веб-інтерфейс здійснює рендеринг часових рядів за 12-годинне вікно спостережень, відображаючи три незалежні графіки сенсорної телеметрії: температурні режими (повітря/ґрунт), динаміку вологості та рівень освітленості. Це дозволяє оператору візуально відслідковувати теплову інерцію системи для подальшого коригування бази правил нечіткого регулятора.

Для здійснення аналізу система формує три незалежні графіки сенсорної телеметрії, кожен з яких виконує специфічну роль у моніторингу процесів життєзабезпечення теплиці. Перший з них демонструє динаміку температурних режимів (рис. 4.5), де одночасна візуалізація показників повітряного простору та ґрунту дозволяє оператору відслідковувати різницю у швидкості нагрівання та охолодження цих середовищ. Це є критично важливим для оцінки високої теплової інерції субстрату відносно більш динамічного повітряного середовища, що дозволяє точніше налаштовувати роботу роботи теплиці.

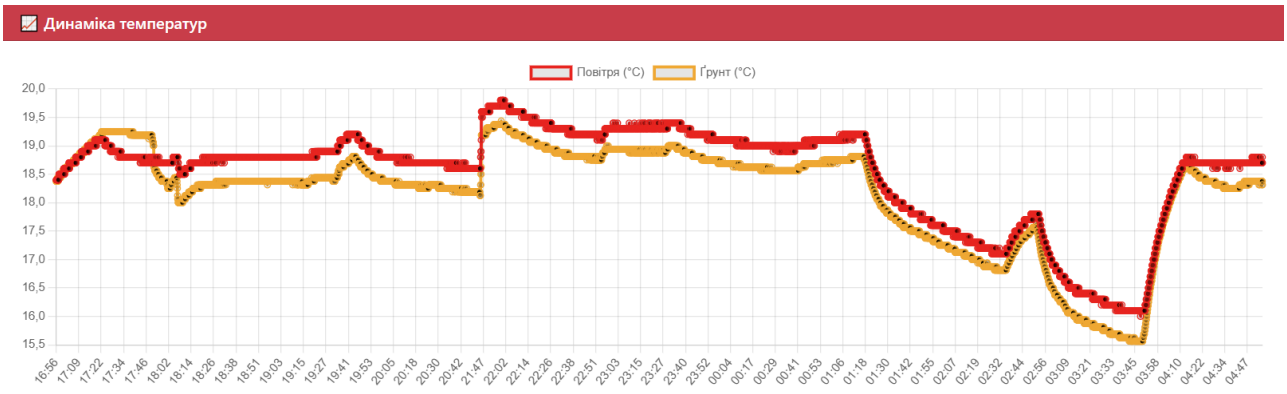


Рисунок 4.5 – Графік “Динаміка температур”

Наступним інструментом візуального контролю є графік динаміки вологості (рис. 4.6), що відображає стан повітря та ґрунту в єдиному часовому просторі. Таке представлення даних дає змогу ефективно стежити за процесами природного випаровування та оцінювати якість спільної роботи ультразвукового зволожувача і системи поливу. Зокрема, це дозволяє вчасно виявляти потенційні технологічні конфлікти, як-от різке падіння вологості повітря під час інтенсивного провітрювання через відкриття кватирки.

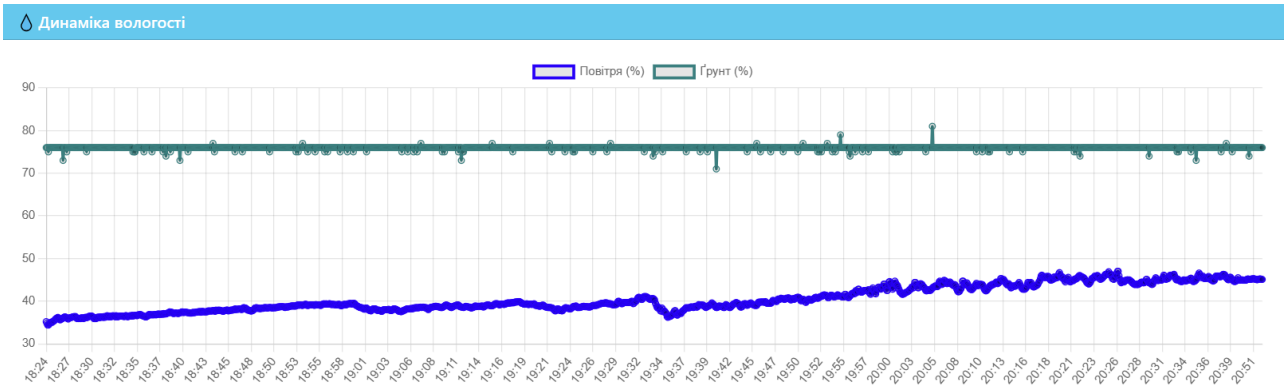


Рисунок 4.6 – Графік “Динаміка вологості”

Завершує комплекс інструментального моніторингу графік рівня освітленості (рис. 4.7), який демонструє природну динаміку сонячної активності протягом доби. Завдяки йому оператор отримує можливість аналізувати періоди освітлення, вплив хмарності на мікроклімат, а також фіксувати точні моменти активації штучного освітлення за допомогою LED-фітострічок. Це гарантує

алгоритмічну компенсацію дефіциту світла та забезпечує стабільність фотосинтетичних процесів навіть за несприятливих погодних умов.

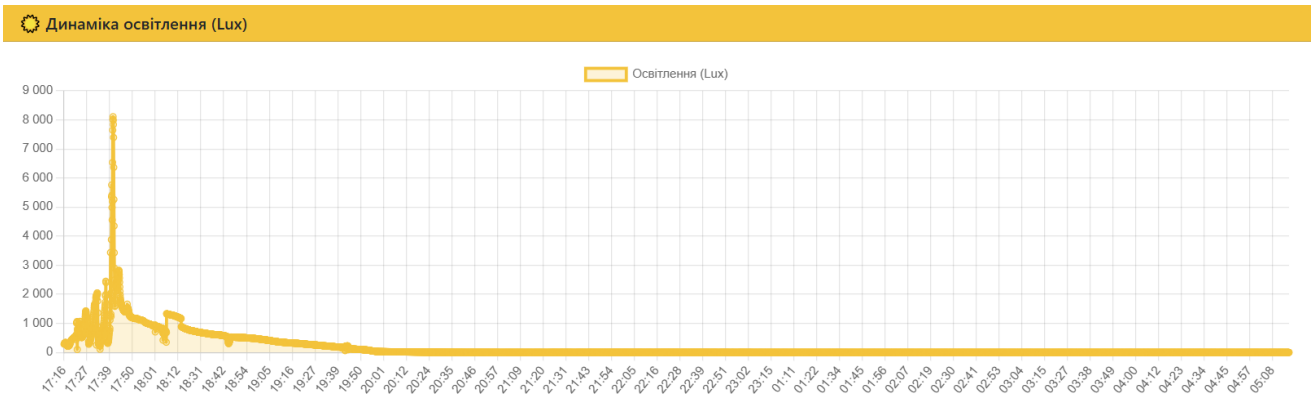


Рисунок 4.7 – Графік “Динаміка освітлення (Lux)”

Найбільш вагомим компонентом дашборду для оцінки роботи розроблених алгоритмів є комплексний графік «Витрати енергії (Актуатори)». Його розроблено для інструментального контролю за роботою регулятора нечіткої логіки та станом дискретних реле.

Візуалізація перехідних процесів на графіку реалізована за інноваційним принципом поєднання двох осей ординат, що дозволяє комплексно оцінювати динаміку різних за своєю фізичною природою виконавчих механізмів. Ліва неперервна вісь відображає плавні зміни ШІМ-сигналів термоелектричного модуля Пельтьє та LED-фітострічки у робочих діапазонах від 0 до 600 та від 0 до 1023 відповідно. Застосування лінійної інтерполяції для цих показників візуально підтверджує відсутність різких стрибків керуючого впливу, що є прямим доказом коректної та стабільної роботи алгоритму нечіткого виведення Мамдані. Натомість права дискретна вісь призначена для відображення динаміки релейних актуаторів та крокових механізмів: вона фіксує бінарні логічні стани (0 або 1) для водяної помпи та зволожувача повітря, а також відображає зміну кута повороту вентиляційної кватирки з використанням методів ступінчастої інтерполяції.

Таке візуальне поєднання на одному полотні дозволяє знайти пряму кореляцію між діями граничного вузла: наприклад, проаналізувати, як порційне

включення помпи впливає на інерційність вологості ґрунту, і як при цьому алгоритм обмежує споживання енергії. Запропонований HMI-дашборд (рис. 4.8) створює цілісну картину для прийняття рішень та підтверджує енергоефективність розробленої Cloud-Edge системи.

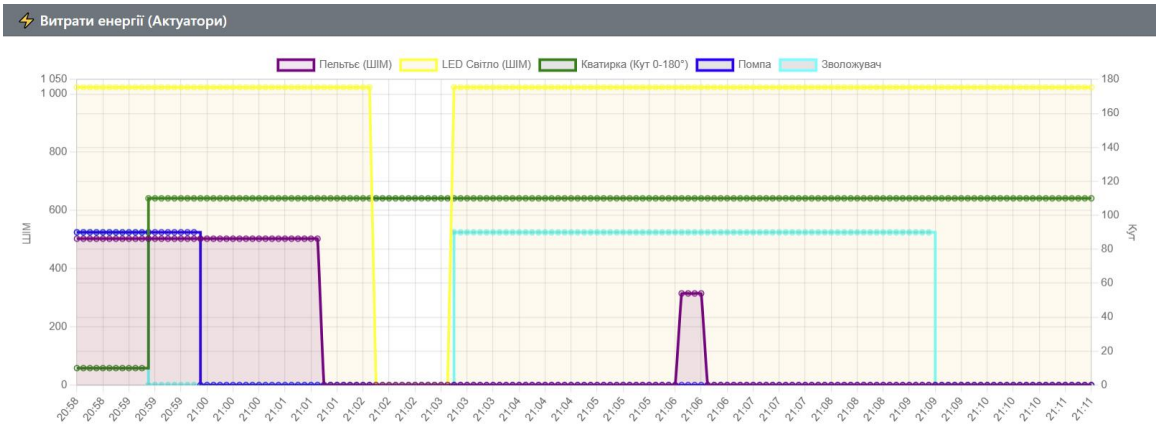


Рисунок 4.8 – Графік “Витрати енергії (Актуатори)”

У процесі експлуатації веб-дашборду було виявлено, що синхронне оновлення сторінки створює дискомфорт для оператора та надмірне навантаження на HTTP-сервер Waitress. Для вирішення проблеми на клієнтській стороні впроваджено технологію AJAX із використанням сучасного інтерфейсу fetch API. Розроблений JavaScript-алгоритм ініціює фонові HTTP-запити кожні 5 секунд. Отриманий JSON-пакет використовується для динамічної підміни масивів даних всередині об'єктів Chart.js, що забезпечило плавне автооновлення всіх графіків у режимі реального часу без повного перезавантаження сторінки браузера.

4.4 Експериментальне дослідження та оцінка результатів

Для підтвердження ефективності розроблених методів та алгоритмів було проведено серію експериментальних досліджень. Тестування виконувалося в умовах реальної роботи сенсорної мережі з використанням методології Software-in-the-Loop (SITL), що дозволило оцінити адекватність генерації керуючих сигналів без ризику пошкодження біологічних об’єктів на етапі пусконалагодження.

4.4.1 Оцінка точності прогнозуючої моделі ARIMA

Перший етап експерименту полягав у перевірці адекватності хмарної моделі машинного навчання. Система працювала в штатному режимі, збираючи телеметрію у БД MySQL. Кожні 10 хвилини планувальник ініціював розрахунок прогнозу моделлю ARIMA(2,1,2) з горизонтом прогнозування у 30 хвилин.

Для оцінки точності було застосовано метрику середньоквадратичної похибки. Аналіз часових рядів із таблиць sensor_telemetry та arima_predictions показав наступні результати:

Для оцінки точності розробленої предиктивної моделі було застосовано метрику середньоквадратичної похибки (RMSE). Аналіз часових рядів, отриманих із таблиць баз даних sensor_telemetry та arima_predictions, продемонстрував високу адаптивність системи до різних експлуатаційних сценаріїв. Зокрема, в умовах стабільного температурного режиму у нічний час ефективно спрацьовував розроблений алгоритм захисту safe_arima. Оскільки стандартне відхилення часового ряду в цей період залишалося меншим за 0.1, модель автоматично уникала ресурсомістких матричних обчислень. Завдяки цьому архітектурному рішенню похибка RMSE практично наблизилася до нуля, а загальне обчислювальне навантаження на vCPU віртуальної машини Azure суттєво знизилося. Натомість, в умовах активної мікрокліматичної динаміки, спровокованої сходом сонця та інтенсивним природним нагріванням приміщення, алгоритм також продемонстрував здатність успішно ідентифікувати температурну тенденцію. За таких нестабільних умов середньоквадратична похибка прогнозу температури повітря склала лише 0.42°C, що є показником високої точності та повністю задовольняє жорсткі агротехнічні вимоги, висунуті до сучасних тепличних комплексів.

4.4.2 Дослідження адекватності упереджувального нечіткого керування

Другий етап експерименту був спрямований на оцінку роботи локального регулятора Fuzzy Logic на мікроконтролері ESP32. Метою було довести перевагу плавного ШІМ-керування над класичним релейним підходом.

Експеримент проводився шляхом аналізу комплексного графіка «Витрати енергії (Актуатори)» на HMI-дашборді. Задана цільова температура ґрунту становила T_{opt}=22.0°C. Під час штучного охолодження сенсора (імітація заморозку), мікроконтролер обчислював поточну похибку та отримував прогнозуючу похибку від хмари.

За результатами детального аналізу логів актуаторів було підтверджено високу ефективність запропонованого алгоритму. Зокрема, завдяки наявності прогнозуючого входу, регулятор нечіткого виведення Мамдані починав плавно підвищувати робочий цикл ШІМ-сигналу елемента Пельтьє завчасно, ще до того, як температура опускалася до критичних значень. Паралельно з цим бездоганно відпрацював програмно-апаратний запобіжник: незважаючи на значні початкові відхилення мікрокліматичних показників, керуючий сигнал жодного разу не перевищив заздалегідь розраховану константу D_{max}=600, що наочно підтверджується лінійною інтерполяцією на графіках дашборду. Таке жорстке обмеження потужності надійно гарантує захист підсистеми живлення від теплового пробою. Важливою перевагою є також те, що на відміну від класичних релейних систем, які безперервно перемикаються між крайніми станами 0% та 100% (неминуче спричиняючи прискорений механічний та електричний знос обладнання), розроблений метод забезпечив стабільний, плавний та асимптотичний вихід керуючого сигналу на оптимальний рівноважний стан.

4.4.3 Перевірка алгоритмів відмовостійкості

Найбільш критичним сценарієм для IoT-систем є втрата зв’язку з хмарним сервером, для тестування якого було проведено експеримент з обривом інтернет-

з’єднання. Хронологічний аналіз поведінки системи показав, що безпосередньо у момент розірвання з'єднання мікроконтролер ESP32 успішно розпізнав втрату підключення та миттєво ізолював мережеві задачі, запобігши блокуванню головного циклу виконання. Вже з першої хвилини ізоляції система продемонструвала здатність продовжувати стабільне локальне керування актуаторами, спираючись на цільові уставки, які були завчасно зарезервовані в енергонезалежній пам’яті мікроконтролера за допомогою бібліотеки Preferences.h. Ключовий етап перевірки відмовостійкості відбувся на шістдесятій хвилині автономної роботи, коли штатно спрацював алгоритм контролю «терміну придатності» для прогнозуючої моделі ARIMA. Оскільки значення змінної lastForecastTime не оновлювалося понад годину, система алгоритмічно визнала наявний прогноз недійсним. Як наслідок, прогнозована похибка була програмно прирівняна до поточної фактичної, що дозволило локальному нечіткому регулятору здійснити повністю безшовний та безпечний перехід із упереджувальної стратегії керування у класичний реактивний режим без переривання життєво важливих технологічних процесів.

Тестування підтвердило надійність розробленої логіки автономного виживання. Біологічні процеси в системі залишаються керованими за будь-яких умов.

4.4.4 Емпіричне калібрування теплових режимів

Одним із найважливіших етапів стендового тестування стало дослідження роботи термоелектричного модуля Пельтьє. Під час первинних запусків модуля TEC1-12706 від напруги 12 В без масивного радіатора було зафіксовано явище теплового перенасичення. Через надлишок потужності відбулося розплавлення внутрішнього припою напівпровідникових переходів та фізична деградація елемента.

Для оптимізації процесу передачі тепла модуль було замінено, розміщено між радіатором та масивною алюмінієвою пластиною-розподілювачем товщиною

0.8 см, а живлення переведено на ізольовану лінію 5 В. Експеримент виявив проблему теплової інерції алюмінієвої пластини. Шляхом серії емпіричних тестів було доведено, що для ідеально плавного прогріву ґрунту без інерційних викидів, максимальний ліміт робочого циклу ШІМ має бути алгоритмічно обмежений на рівні 600. Внесення цього обмеження у функцію дефазифікації повністю стабілізувало тепловий контур.

4.5 Висновки до четвертого розділу

У четвертому розділі детальну програмно-апаратну реалізацію розробленого методу прогнозуючого керування та виконано оцінку його ефективності.

Успішно розгорнуто хмарну інфраструктуру на базі Microsoft Azure. Використання багатопотокового WSGI-сервера Waitress та СУБД MySQL забезпечило високу доступність системи, а впровадження захисного алгоритму *safe_arima* повністю усунуло проблему математичного зависання ШІ на стаціонарних наборах даних.

Реалізовано апаратний рівень граничного вузла на базі ESP32 з використанням дискретних транзисторів IRLB8721 для усунення паразитних струмів витоку. Розроблена прошивка з архітектурою неблокуючого кінцевого автомата забезпечила паралельну роботу мережевого стеку, апаратного ШІМ-обмеження (D_{max}=600) та інтелектуальну програмну фільтрацію сенсорних аномалій.

Розроблено та імплементовано інтерактивний веб-дашборд із використанням технологій Flask, Bootstrap 5 та Chart.js. Впровадження технології AJAX (fetch API) дозволило реалізувати асинхронне оновлення графіків у реальному часі. Механізми авторизації та використання топіків Retain забезпечили безпечну та гарантовану синхронізацію цільових уставок між оператором і мікроконтролером.

Результати SITL-експериментів довели високу точність прогнозування моделлю ARIMA (RMSE=0.42°C). Підтверджено абсолютну відмовостійкість системи: закриття алгоритму нечіткої логіки механізмами пам’яті (Preferences.h) та

TTL-таймауту гарантує виживання об’єкта керування навіть при тривалій втраті доступу до хмарної інфраструктури.

Загалом, розроблений комплекс повністю відповідає технічному завданню та готовий до дослідно-промислової експлуатації.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено гібридну Cloud-Edge систему автоматизованого керування мікрокліматом теплиці з використанням методів прогнозуючої аналітики та упереджувальної нечіткої логіки.

Впровадження результатів роботи дозволили суттєво підвищити енергоефективність, стабільність та відмовостійкість процесу підтримання мікроклімату в умовах закритого ґрунту. Завдяки переходу від класичного реактивного до інтелектуального прогнозуючого керування, вдалося алгоритмічно обмежити пікові навантаження на силову електроніку, усунути температурні коливання та забезпечити автономне виживання біологічних об’єктів навіть за умов критичної втрати зв’язку з хмарним сервером.

Поставлену мету було досягнуто шляхом розв’язання таких основних завдань:

- проведено комплексний аналіз існуючих методів керування мікрокліматом та обґрунтовано доцільність застосування багаторівневої IoT-інфраструктури у поєднанні з алгоритмами машинного навчання для подолання інерційності тепличних процесів;

- розроблено апаратне забезпечення граничного вузла на базі мікроконтролера ESP32. Спроектовано відмовостійку топологію живлення з використанням ізольованих DC-DC перетворювачів. Це дозволило усунути високочастотні апаратні шуми аналогових сенсорів та забезпечити надійне керування потужним термоелектричним модулем Пельтьє через MOSFET-модулі D4184;

- розроблено математичне забезпечення системи, основою якого став двопортовий нечіткий регулятор Мамдані. Впровадження керування «за відхиленням» із одночасним врахуванням поточної та прогнозованої похибки (від моделі ARIMA) дозволило реалізувати упереджувальний вплив на виконавчі

механізми. Математично розраховано та програмно імплементовано жорстке обмеження ШІМ-сигналу для захисту блоку живлення від теплового пробою;

- створено програмне забезпечення хмарного сервера на базі промислового WSGI-сервера Waitress з інтегрованою базою даних MySQL. У процес прогнозуючої моделі ARIMA впроваджено алгоритм зниження частоти дискретизації та обгортку захисту від математичного зависання на стаціонарних даних, що в десятки разів знизило навантаження на обчислювальні ресурси віртуальної машини Azure;

- розроблено адаптивний HMI-дашборд оператора для інструментальної візуалізації телеметрії та аналізу енергоефективності актуаторів;

- успішно реалізовано та перевірено шляхом SITL-тестування алгоритми багаторівневої відмовостійкості. Впровадження механізму «терміну придатності» прогнозу та локальне збереження уставок у незалежній Flash-пам’яті мікроконтролера гарантують безшовний перехід системи у безпечний автономний режим при обривах інтернет-з’єднання

За темою “Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи” опубліковано наукові тези [55] у 17-й Міжнародній студентській науковотехнічній конференції «Перспективні мережеві та комп’ютерні технології» – ПЕРСИК 2026 (м. Харків, 23 квіт. 2026). Харків, 2026.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Review of temperature management strategies and techniques in the greenhouse microenvironment / T. D. Akpenpuun et al. *Adeleke University Journal of Engineering and Technology*. 2023. Vol. 6, no. 2. P. 126–147.

2. A practical solution for multivariable control of temperature and humidity in greenhouses / F. García-Mañas et al. *European Journal of Control*. 2024. Vol. 77. P. 100967.

3. Data-driven robust model predictive control for greenhouse temperature control and energy utilisation assessment / F. Mahmood et al. *Applied Energy*. 2023. Vol. 343. P. 121190.

4. Eli-Chukwu N. C. Applications of artificial intelligence in agriculture: A review. *Engineering, Technology & Applied Science Research*. 2019. Vol. 9, no. 4.

5. A hybrid IoT and machine learning framework for smart greenhouse automation in sustainable agriculture / V. Venkataramanan et al. *International Research Journal of Multidisciplinary Technovation*. 2025. Vol. 7, no. 4. P. 58–69.

6. Greenhouse Climate Controller by Using of Internet of Things Technology and Fuzzy Logic / A. Sahour et al. *Instrumentation, Mesures, Métrologies*. 2021. Vol. 20, no. 1.

7. Alhassan A. I., Mohammed A. C., Andrew B. Comparison Between Fuzzy Logic and PID Controllers in Temperature Control of Laboratory Incubator. *International Journal of Science and Engineering Applications*. 2022. P. 133–138.

8. Plant leaf disease detection, classification, and diagnosis using computer vision and artificial intelligence: A review / A. Bhargava et al. *IEEE Access*. 2024. Vol. 12. P. 37443–37469.

9. Computer vision with deep learning for plant phenotyping in agriculture: A survey / A. L. Chandra et al. *arXiv preprint arXiv:2006.11391*. 2020.

10. A Review of Environmental Control Strategies and Models for Modern Agricultural Greenhouses / S. Chen et al. *Sensors*. 2025. Vol. 25, no. 5. P. 1388.

11. Shamshiri R. R. Greenhouse Crop Simulation Models and Microclimate Control Systems, A Review. *Intech*. 2021.

12. Hybrid approach for prediction of temperature and moisture in greenhouses using ARIMA, ARTFIMA and SVM methods / M. R. Seba et al. *Applied Ecology & Environmental Research*. 2023. Vol. 21, no. 6.

13. Prediction of Carbon Dioxide Concentrations in Strawberry Greenhouse by Using Time Series Models / S. H. Shin et al. *Agriculture*. 2024. Vol. 14, no. 11. P. 1895.

14. Integration of deep learning and sparrow search algorithms to optimize greenhouse microclimate prediction for seedling environment suitability / D. Shi et al. *Agronomy*. 2024. Vol. 14, no. 2. P. 254.

15. Deep learning based prediction on greenhouse crop yield combined TCN and RNN / L. Gong et al. *Sensors*. 2021. Vol. 21, no. 13. P. 4537.

16. A systematic review of optimal and practical methods in design, construction, control, energy management and operation of smart greenhouses / M. Ghiasi et al. *IEEE Access*. 2023. Vol. 12. P. 2830–2853.

17. Chen Q., Hu X. Design of intelligent control system for agricultural greenhouses based on adaptive improved genetic algorithm for multi-energy supply system. *Energy Reports*. 2022. Vol. 8. P. 12126–12138.

18. Kalyani Y., Collier R. A systematic survey on the role of cloud, fog, and edge computing combination in smart agriculture. *Sensors*. 2021. Vol. 21, no. 17. P. 5922.

19. Upadhyayaa A., Pala D., Deyb R. EECSI: An IoT application-based energy efficient three-layer clustered smart irrigation approach. 2021.

20. Cloud–edge–device collaborative computing in smart agriculture: architectures, applications, and future perspectives / P. Yu et al. *Frontiers in Plant Science*. 2025. Vol. 16. P. 1668545.

21. Engineering a multi model fallback system for edge devices / G. Kadve et al. *Results in Engineering*. 2025. Vol. 26. P. 105165.

22. Alharbi H. A., Aldossary M. Energy-efficient edge-fog-cloud architecture for IoT-based smart agriculture environment. *IEEE Access*. 2021. Vol. 9. P. 110480–110492.

23. Montgomery D. C., Jennings C. L., Kulahci M. Introduction to time series analysis and forecasting. *John Wiley & Sons*, 2021.

24. Auto-correlation function (ACF) and partial auto-correlation function (PACF). Medium. URL: https://medium.com/@ritusantra/auto-correlation-function-acf-and-partial-auto-correlation-function-pacf-e29ec2db2b1b (date of access: 03.04.2026).

25. Computational Intelligence: A Methodological Introduction : 3rd ed. / R. Kruse et al. *Springer*, 2022.

26. Correa-Quiroz J. J., Toribio-Barrueto M. A., Castro-Vargas C. IoT system with ESP32 for smart drip irrigation and climate monitoring in greenhouses. *Emerg. Sci. J.* 2025. Vol. 9, no. 3. P. 1133–1157.

27. XL4015 5A 180KHz 36V Buck DC to DC Converter. Technical Datasheet. URL: https://datasheet4u.com/pdf/786208/XL4015.pdf (date of access: 03.05.2026).

28. Comparison between two-stage and three-stage Peltier thermoelectric coolers driven by pulse width modulation / D. D. Kamasi et al. *Journal of Physics: Conference Series*. 2020. Vol. 1528, no. 1.

29. Tsaqief M. F., Sutopo J. Comparative Performance Analysis Between the MQTT and WebSocket Protocols. *bit-Tech*. 2025. Vol. 8, no. 2. P. 2227–2237.

30. Transport and Application Layer Protocols for IoT: Comprehensive Review / I. Petrescu et al. *Technologies*. 2025. Vol. 13, no. 12. P. 583.

31. Werlinder M. Comparing the scalability of MQTT and WebSocket communication protocols using Amazon Web Services. 2020.

32. A performance analysis of internet of things networking protocols: Evaluating MQTT, CoAP, OPC UA / D. Silva et al. *Applied Sciences*. 2021. Vol. 11, no. 11. P. 4879.

33. El Ouadghiri M., Aghoutane B., El Farissi N. Communication model in the internet of things. *Procedia Computer Science*. 2020. Vol. 177. P. 72–77.

34. Thakur R. K., Kumari R. A Comparison of Various IoT Application Layer Protocol. *American Journal of Electronics & Communication*. 2022. Vol. 3, no. 1. P. 28–34.

35. Al-Hasani A. T., Al-Qaraguli M. Seasonal temperature prediction in niamey: a prophet model approach. *EDRAAK 2025*. 2025. P. 7–11.

36. Mahajan H. Comparative Study of Arima, Lstm and Prophet Models for Time Series Forecasting: A Comprehensive Review. 2026.

37. Analysis and forecasting of meteorological drought using PROPHET and SARIMA models deploying machine learning technique for southwestern region of Bangladesh / M. A. Hossain et al. *Environmental and Sustainability Indicators*. 2025. Vol. 27. P. 100761.

38. Predicting temperature of major cities using machine learning and deep learning / W. Jaharabi et al. *arXiv preprint arXiv:2309.13330*. 2023.

39. Wang Z. Comparing SARIMA and Prophet Models for Short-Term Electricity Load Forecasting. *ITM Web of Conferences*. EDP Sciences, 2026. Vol. 84.

40. Temperature Forecasting with LSTM: A Case Study on Kyiv Weather Data / O. Makoveichuk et al. 2025.

41. Developing a seasonal-adjusted machine-learning-based hybrid time-series model to forecast heatwave warning / M. M. U. Qureshi et al. *Scientific Reports*. 2025. Vol. 15, no. 1. P. 8699.

42. Albahli S. LSTM vs. prophet: achieving superior accuracy in dynamic electricity demand forecasting. *Energies*. 2025. Vol. 18, no. 2. P. 278.

43. Effect of thermal screen position on greenhouse microclimate and impact on crop growth and yield / E. Zakir et al. *Nigerian Journal of Technological Development*. 2022. Vol. 19, no. 4. P. 417–432.

44. Overview of modelling techniques for greenhouse microclimate environment and evapotranspiration / H. Yan et al. *International Journal of Agricultural and Biological Engineering*. 2021. Vol. 14, no. 6. P. 1–8.

45. Response of tomato genotypes under different high temperatures in field and greenhouse conditions / S. Ro et al. *Plants*. 2021. Vol. 10, no. 3. P. 449.

46. Giordano M., Petropoulos S. A., Rouphael Y. Response and defence mechanisms of vegetable crops against drought, heat and salinity stress. *Agriculture*. 2021. Vol. 11, no. 5. P. 463.

47. Mathew T. E., Sabu A., Sengan S. Microclimate monitoring system for irrigation water optimization using IoT. *Measurement: Sensors*. 2023. Vol. 27. P. 100727.

48. Smart-IoT platform to monitor microclimate conditions in tropical regions / C. J. Gonzalez et al. *IOP Conference Series: Earth and Environmental Science*. IOP Publishing, 2021. Vol. 835, no. 1.

49. Improving Solar Energy-Harvesting Wireless Sensor Network (SEH-WSN) with Hybrid Li-Fi/Wi-Fi, Integrating Markov Model, Sleep Scheduling, and Smart Switching Algorithms / H. A. Helmy et al. *Technologies*. 2025. Vol. 13, no. 10. P. 437.

50. An adaptive TE-PV hybrid energy harvesting system for self-powered iot sensor applications / M. K. Mishu et al. *Sensors*. 2021. Vol. 21, no. 8. P. 2604.

51. A smart energy monitoring system using ESP32 microcontroller / H. J. El-Khozondar et al. *e-Prime-Advances in Electrical Engineering, Electronics and Energy*. 2024. Vol. 9. P. 100666.

52. Power consumption measurements of WSN based on Arduino / U. T. Salim et al. *IOP Conference Series: Materials Science and Engineering*. IOP Publishing, 2021. Vol. 1152, no. 1.

53. Bhavya A. R., Sudharshan K. M. Optimizing Energy Efficiency in Battery-powered IoT Devices through Hardware Optimization and Voltage Scaling. *Engineering, Technology & Applied Science Research*. 2025. Vol. 15, no. 2. P. 21769–21773.

54. Лісовий В. М. Програмно-апаратний засіб для керування параметрами мікроклімату теплиці : кваліфікаційна робота бакалавра : 123 Комп’ютерна інженерія / В. М. Лісовий ; Хмельниц. нац. ун-т. – Хмельницький, 2024. – 101 с.

55. Лісовий В. Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи. *ПерСиК* 2026, Харків, Україна, 23 квіт. 2026.

56. A cloud-based IoT platform for precision control of soilless greenhouse cultivation / A. Sagheer et al. Sensors. 2020. Vol. 21, no. 1. P. 223.

57. Laktionov I., Vovna O., Kabanets M. Information technology for comprehensive monitoring and control of the microclimate in industrial greenhouses

based on fuzzy logic. Journal of Artificial Intelligence and Soft Computing Research. 2023. Vol. 13, no. 1. P. 19–35.

58. IoT based smart greenhouse framework and control strategies for sustainable agriculture / M. S. Farooq et al. IEEE Access. 2022. Vol. 10. P. 99394–99420.

59. Argento S., Garcia G., Treccarichi S. Sustainable and low-input techniques in Mediterranean greenhouse vegetable production. Horticulturae. 2024. Vol. 10, no. 9. P. 997.

60. Machine learning for smart agriculture: a comprehensive survey / M. R. Mahmood et al. IEEE Transactions on Artificial Intelligence. 2023. Vol. 5, no. 6. P. 2568–2588.

61. Remote-sensing data and deep-learning techniques in crop mapping and yield prediction: A systematic review / A. Joshi et al. Remote Sensing. 2023. Vol. 15, no. 8. P. 2014.

62. An advanced deep learning models-based plant disease detection: A review of recent research / M. Shoaib et al. Frontiers in Plant Science. 2023. Vol. 14. P. 1158933.

63. A comprehensive review on deep learning assisted computer vision techniques for smart greenhouse agriculture / J. U. M. Akbar et al. IEEE Access. 2024. Vol. 12. P. 4485–4522.

64. Rocha G. A. O., Pichimata M. A., Villagran E. Research on the microclimate of protected agriculture structures using numerical simulation tools: A technical and bibliometric analysis as a contribution to the sustainability of under-cover cropping in tropical and subtropical countries. Sustainability. 2021. Vol. 13, no. 18. P. 10433.

65. Enhancing global agricultural monitoring system for climate-smart agriculture / L. Yu et al. Climate Smart Agriculture. 2025. Vol. 2, no. 1. P. 100037.

66. Data-driven evaluation of machine learning models for climate control in operational smart greenhouses / J. Morales-García et al. Journal of Ambient Intelligence and Smart Environments. 2023. Vol. 15, no. 1. P. 3–17.

67. AI-Enabled Energy Management for Sustainable Smart Greenhouses: An Integrated Review / S. R. Biswal et al. IEEE Access. 2026. Vol. 14. P. 5482–5509.

68. Chantadze I. Design and Evaluation of an Energy-Efficient Automated Greenhouse Management System for Optimized Microclimate Control. Georgian Scientists. 2026. Vol. 8, no. 1. P. 112–125.

69. Hu G., You F. Renewable energy-powered semi-closed greenhouse for sustainable crop production using model predictive control and machine learning for energy management. Renewable and Sustainable Energy Reviews. 2022. Vol. 168. P. 112790.

70. A hydroponic greenhouse fuzzy control system: design, development and optimization using the genetic algorithm / H. Khafajeh et al. Spanish Journal of Agricultural Research. 2023. Vol. 21, no. 1. P. e0201.

71. Liu L., Wang Q., Li B. Q. A system architecture for intelligent agriculture based on edge computing. International Journal of Computer Applications in Technology. 2020. Vol. 64, no. 2. P. 126–132.

72. Briatore F., Braggio M. Edge, Fog and Cloud Computing framework for flexible production. Procedia Computer Science. 2025. Vol. 253. P. 2206–2218.

73. IoT and big data integration for real-time agricultural monitoring / B. D. Patil et al. Journal of Advanced Zoology. 2023. Vol. 44, no. 2. P. 3079–3089.

74. Performance evaluation of CoAP and MQTT with security support for IoT environments / V. Seoane et al. Computer Networks. 2021. Vol. 197. P. 108338.

75. Baccay J. B., Vicente C. P., Bravo M. T. IoT-based Automated Greenhouse with Monitoring and Control using MQTT Protocol. Turkish Online Journal of Qualitative Inquiry. 2021. Vol. 12, no. 6.

76. LightIoT: Lightweight and secure communication for energy-efficient IoT in health informatics / M. A. Jan et al. IEEE Transactions on Green Communications and Networking. 2021. Vol. 5, no. 3. P. 1202–1211.

77. LoRaEdge-ESP32 synergy: Revolutionizing farm weather data collection with low-power, long-range IoT / V. Godase et al. Advance Research in Analog and Digital Communications. 2025. Vol. 2, no. 2. P. 1–11.

78. A way towards energy autonomous wireless sensing for ev battery management system / B. Muneer et al. IEEE Journal of Microwaves. 2025.

79. Application of Fuzzy logic and IoT in a small-scale Smart Greenhouse System / V. Thomopoulos et al. Smart Agricultural Technology. 2024. Vol. 8. P. 100446.

80. Zurita C. M., Assis F. Hybrid Control Strategies for Greenhouse Climate Regulation: PID, Fuzzy, and Neuro-Fuzzy Comparative Implementation in Temperate-Dry Crop Systems. 2025 XV Symposium on Computing Systems Engineering (SBESC). IEEE, 2025. P. 1–6.

ДОДАТОК А
(обов’язковий)

Лістинг вихідного коду мікропрограмного забезпечення граничного вузла на базі ESP32 для керування мікрокліматом

```
#include <Wire.h>
#include <BH1750.h>
#include <DallasTemperature.h>
#include <OneWire.h>
#include <DHT.h>
#include <LiquidCrystal_I2C.h>
#include <ESP32Servo.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <Preferences.h>
#include <Fuzzy.h>

// --- МЕРЕЖА ---
const char* ssid = "SSID";
const char* password = "password";
const char* mqtt_server = "server_ip";

// --- ПІНИ ---
#define ONE_WIRE_BUS 4
#define DHTPIN 16
#define SOIL_MOISTURE_PIN 32
#define RELAY_PIN_WP 25      // Реле помпи
#define RELAY_PIN_AH 26      // Зволожувач
#define TRANSISTOR_PELT 27   // Пельтьє
#define TRANSISTOR_LED 14    // Фітострічка
#define SERVO_PIN 13         // Кватирка

DHT dht(DHTPIN, DHT22);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
LiquidCrystal_I2C lcd(0x27, 16, 2);
BH1750 lightMeter;
Servo myServo;

WiFiClient espClient;
PubSubClient client(espClient);
Preferences preferences;
Fuzzy *fuzzy = new Fuzzy();

// --- ЗМІННІ СЕНСОРІВ ---
float t = 0, h = 0, lux = 0, soilTemp = 0;
int soilMoisturePercent = 0;
const int AirValue = 2945;
const int WaterValue = 1200;

// --- ЦІЛЬОВІ ВИМОГИ РОСЛИНИ (Уставки) ---
float target_air_temp, target_air_hum, target_soil_temp, target_soil_moist, target_light_lux;
float predicted_air_t = 0, predicted_air_h = 0, predicted_soil_t = 0;

int currentPeltierPWM = 0;

// --- ТАЙМЕРИ ---
unsigned long lastSensorTime = 0;
```

```
unsigned long lastActuatorTime = 0;
unsigned long lastMqttTime = 0;
unsigned long lastReconnectAttempt = 0;
unsigned long lastForecastTime = 0;

// --- ПАМ'ЯТЬ АКТУАТОРІВ ДЛЯ ГРАФІКІВ ---
bool wasPumpActive = false;
bool wasHumidifierActive = false;
int currentWindowAngle = 10;
int currentLedPWM = 0;

// --- АСИНХРОННІ ЗМІННІ АКТУАТОРІВ ---
bool humidifierToggleActive = false;
unsigned long humidifierTimer = 0;
int humidifierState = 0;
int moistureAnomalyCounter = 0;

bool pumpActive = false;
unsigned long pumpTimer = 0;

byte degreeSymbol[8] = {B00111, B00101, B00111, B00000, B00000, B00000, B00000, B00000};



// 1. ФУНКЦІЇ АКТУАТОРІВ ТА FSM
void setPeltierPWM(int pwmValue) {
  if (pwmValue > 600) pwmValue = 600;
  if (pwmValue <= 0) {
    digitalWrite(TRANSISTOR_PELT, LOW);
  } else {
    analogWrite(TRANSISTOR_PELT, pwmValue);
  }
}

void triggerHumidifier() {
  if (!humidifierToggleActive) {
    humidifierToggleActive = true;
    humidifierState = 1;
    humidifierTimer = millis();
    digitalWrite(RELAY_PIN_AH, HIGH);
  }
}

void processHumidifierState() {
  if (!humidifierToggleActive) return;
  unsigned long currentMillis = millis();

  switch (humidifierState) {
    case 1: if (currentMillis - humidifierTimer >= 200) { digitalWrite(RELAY_PIN_AH, LOW); humidifierState = 2; humidifierTimer = currentMillis; } break;
    case 2: if (currentMillis - humidifierTimer >= 200) { humidifierState = 3; humidifierTimer = currentMillis; } break;
    case 3: if (currentMillis - humidifierTimer >= 5000){ digitalWrite(RELAY_PIN_AH, HIGH); humidifierState = 4; humidifierTimer = currentMillis; } break;
    case 4: if (currentMillis - humidifierTimer >= 200) { digitalWrite(RELAY_PIN_AH, LOW); humidifierState = 5; humidifierTimer = currentMillis; } break;
    case 5: if (currentMillis - humidifierTimer >= 200) { digitalWrite(RELAY_PIN_AH, HIGH); humidifierState = 6; humidifierTimer = currentMillis; } break;
    case 6: if (currentMillis - humidifierTimer >= 200) { digitalWrite(RELAY_PIN_AH, LOW); humidifierState = 7; humidifierTimer = currentMillis; } break;
```

```
    case 7: if (currentMillis - humidifierTimer >= 200) { humidifierState = 0; humidifierToggleActive = false; } break;
  }
}

void triggerPump() {
  if (!pumpActive) {
    pumpActive = true;
    pumpTimer = millis();
    digitalWrite(RELAY_PIN_WP, HIGH);
    Serial.println("[PUMP] Полив розпочато (1 секунда)...");
  }
}

void processPumpState() {
  if (pumpActive && (millis() - pumpTimer >= 1000)) {
    digitalWrite(RELAY_PIN_WP, LOW);
    pumpActive = false;
    Serial.println("[PUMP] Полив завершено.");
  }
}



// НАЛАШТУВАННЯ ПРОГНОЗУЮЧОЇ FUZZY LOGIC
void setupFuzzyLogic() {
  // ВХІД 1: Поточне відхилення (Current Error)
  FuzzyInput *currError = new FuzzyInput(1);
  FuzzySet *currNeg = new FuzzySet(-20, -20, -1, 0); // Перегрів зараз
  FuzzySet *currZero = new FuzzySet(-1, 0, 0, 1);    // Ідеально зараз
  FuzzySet *currPos = new FuzzySet(0, 1, 20, 20);    // Холодно зараз
  currError->addFuzzySet(currNeg);
  currError->addFuzzySet(currZero);
  currError->addFuzzySet(currPos);
  fuzzy->addFuzzyInput(currError);

  // ВХІД 2: Прогнозоване відхилення від ARIMA (Predicted Error)
  FuzzyInput *predError = new FuzzyInput(2);
  FuzzySet *predNeg = new FuzzySet(-20, -20, -1, 0); // Буде перегрів
  FuzzySet *predZero = new FuzzySet(-1, 0, 0, 1);    // Буде ідеально
  FuzzySet *predPos = new FuzzySet(0, 1, 20, 20);    // Буде холодно
  predError->addFuzzySet(predNeg);
  predError->addFuzzySet(predZero);
  predError->addFuzzySet(predPos);
  fuzzy->addFuzzyInput(predError);

  // ВИХІД: Потужність Пельтьє
  FuzzyOutput *peltierPower = new FuzzyOutput(1);
  FuzzySet *powerOff = new FuzzySet(0, 0, 0, 0);
  FuzzySet *powerLow = new FuzzySet(0, 150, 300, 450);
  FuzzySet *powerHigh = new FuzzySet(350, 480, 600, 600);
  peltierPower->addFuzzySet(powerOff);
  peltierPower->addFuzzySet(powerLow);
  peltierPower->addFuzzySet(powerHigh);
  fuzzy->addFuzzyOutput(peltierPower);

  // Правило 1: Якщо ВЖЕ Перегрів -> Вимкнути
  FuzzyRuleAntecedent *ifCurrNeg = new FuzzyRuleAntecedent();
```

```
  ifCurrNeg->joinSingle(currNeg);

  FuzzyRuleConsequent *thenOff = new
FuzzyRuleConsequent();

  thenOff->addOutput(powerOff);

  fuzzy->addFuzzyRule(new FuzzyRule(1,
ifCurrNeg, thenOff));


  // Правило 2: Якщо Ідеально, і Буде
Ідеально -> Легка підтримка

  FuzzyRuleAntecedent
*ifCurrZero_PredZero = new
FuzzyRuleAntecedent();

  ifCurrZero_PredZero-
>joinWithAND(currZero, predZero);

  FuzzyRuleConsequent *thenLow1 = new
FuzzyRuleConsequent();

  thenLow1->addOutput(powerLow);

  fuzzy->addFuzzyRule(new FuzzyRule(2,
ifCurrZero_PredZero, thenLow1));


  // Правило 3: Якщо Ідеально, але БУДЕ
Холодно -> Гріти сильно заздалегідь

  FuzzyRuleAntecedent *ifCurrZero_PredPos
= new FuzzyRuleAntecedent();

  ifCurrZero_PredPos-
>joinWithAND(currZero, predPos);

  FuzzyRuleConsequent *thenHigh1 = new
FuzzyRuleConsequent();

  thenHigh1->addOutput(powerHigh);

  fuzzy->addFuzzyRule(new FuzzyRule(3,
ifCurrZero_PredPos, thenHigh1));


  // Правило 4: Якщо Ідеально, але БУДЕ
Перегрів -> Вимкнути заздалегідь

  FuzzyRuleAntecedent *ifCurrZero_PredNeg
= new FuzzyRuleAntecedent();

  ifCurrZero_PredNeg-
>joinWithAND(currZero, predNeg);

  FuzzyRuleConsequent *thenOff2 = new
FuzzyRuleConsequent();

  thenOff2->addOutput(powerOff);

  fuzzy->addFuzzyRule(new FuzzyRule(4,
ifCurrZero_PredNeg, thenOff2));


  // Правило 5: Якщо Холодно, але БУДЕ
Перегрів -> Гріти ледь-ледь

  FuzzyRuleAntecedent *ifCurrPos_PredNeg
= new FuzzyRuleAntecedent();

  ifCurrPos_PredNeg->joinWithAND(currPos,
predNeg);

  FuzzyRuleConsequent *thenLow2 = new
FuzzyRuleConsequent();

  thenLow2->addOutput(powerLow);

  fuzzy->addFuzzyRule(new FuzzyRule(5,
ifCurrPos_PredNeg, thenLow2));


  // Правило 6: Якщо Холодно, і Буде
Холодно -> Гріти на максимум

  FuzzyRuleAntecedent *ifCurrPos_PredPos
= new FuzzyRuleAntecedent();

  ifCurrPos_PredPos->joinWithAND(currPos,
predPos);

  FuzzyRuleConsequent *thenHigh2 = new
FuzzyRuleConsequent();

  thenHigh2->addOutput(powerHigh);

  fuzzy->addFuzzyRule(new FuzzyRule(6,
ifCurrPos_PredPos, thenHigh2));

}



// 3. MQTT ТА ЗВ'ЯЗОК З ХМАРОЮ

void mqttCallback(char* topic, byte*
payload, unsigned int length) {

  String topicStr = String(topic);

  String message;

  for (int i = 0; i < length; i++)
message += (char)payload[i];


  StaticJsonDocument<256> doc;

  if (deserializeJson(doc, message))
return;
```

```
  if (topicStr == "greenhouse/config") {
    target_air_temp =
doc["target_air_temp"];
    target_air_hum =
doc["target_air_hum"];
    target_soil_temp =
doc["target_soil_temp"];
    target_soil_moist =
doc["target_soil_moisture"];
    target_light_lux =
doc["target_light_lux"];

    preferences.putFloat("air_t",
target_air_temp);
    preferences.putFloat("air_h",
target_air_hum);
    preferences.putFloat("soil_t",
target_soil_temp);
    preferences.putFloat("soil_m",
target_soil_moist);
    preferences.putFloat("light_l",
target_light_lux);
    Serial.println("\n[MQTT] Отримано
нові УСТАВКИ!");
  }
  else if (topicStr ==
"greenhouse/forecast/arima") {
    predicted_air_t =
doc["predicted_temp"];
    predicted_air_h =
doc["predicted_hum"];
    predicted_soil_t =
doc["predicted_soil_temp"];
    lastForecastTime = millis();
    Serial.println("\n[MQTT] Отримано
ПРОГНОЗ ARIMA!");
  }
}

// 4. SETUP
void setup() {
  Serial.begin(115200);
  Wire.begin(21, 22);

  preferences.begin("ghouse", false);
  target_air_temp =
preferences.getFloat("air_t", 20.0);
  target_air_hum =
preferences.getFloat("air_h", 50.0);
  target_soil_temp =
preferences.getFloat("soil_t", 22.0);
  target_soil_moist =
preferences.getFloat("soil_m", 40.0);
  target_light_lux =
preferences.getFloat("light_l", 2000.0);

  analogReadResolution(12);
  analogWriteResolution(TRANSISTOR_PELT,
10);
  analogWriteFrequency(TRANSISTOR_PELT,
50); //Знизив частоту щоб не пищав БЖ
  //analogWriteResolution(TRANSISTOR_LED,
10);

  sensors.begin();
  lightMeter.begin();
  dht.begin();

  lcd.init();
  lcd.backlight();
  lcd.createChar(0, degreeSymbol);

  myServo.setPeriodHertz(50);
  myServo.attach(SERVO_PIN, 500, 2400);
  myServo.write(10);

  pinMode(RELAY_PIN_WP, OUTPUT);
  pinMode(RELAY_PIN_AH, OUTPUT);
```

```
  pinMode(TRANSISTOR_PELT, OUTPUT);
  pinMode(TRANSISTOR_LED, OUTPUT);

  digitalWrite(RELAY_PIN_WP, LOW);
  digitalWrite(RELAY_PIN_AH, LOW);
  setPeltierPWM(0);
  digitalWrite(TRANSISTOR_LED, LOW);

  setupFuzzyLogic();

  WiFi.begin(ssid, password);
  client.setServer(mqtt_server, 1883);
  client.setCallback(mqttCallback);
}

// 5. MAIN LOOP
void loop() {
  unsigned long currentMillis = millis();

  // --- АСИНХРОННІ ТАЙМЕРИ ---
  processHumidifierState();
  processPumpState();

  // --- MQTT РЕКОННЕКТ ---
  if (WiFi.status() == WL_CONNECTED) {
    if (!client.connected()) {
      if (currentMillis - lastReconnectAttempt > 5000) {
        lastReconnectAttempt = currentMillis;
        if (client.connect("ESP32_Greenhouse")) {
client.subscribe("greenhouse/config");
client.subscribe("greenhouse/forecast/arima");
        }
      }
    } else {
      client.loop();
    }
  }

  // --- ЗБІР ДАНИХ (Кожні 5 сек) ---
  if (currentMillis - lastSensorTime >= 5000) {
    lastSensorTime = currentMillis;

    h = dht.readHumidity();
    t = dht.readTemperature();
    // --- ПРОГРАМНИЙ ФІЛЬТР ВОЛОГОСТІ (З лічильником довіри) ---
    int rawSoil = analogRead(SOIL_MOISTURE_PIN);
    int newMoisture = map(rawSoil, AirValue, WaterValue, 0, 100);
    newMoisture = constrain(newMoisture, 0, 100);

    // Якщо зміна менше 20% або це перший запуск - все добре
    if (abs(newMoisture - soilMoisturePercent) <= 20 || soilMoisturePercent == 0) {
        soilMoisturePercent = newMoisture;
        moistureAnomalyCounter = 0; // Скидаємо лічильник помилок
    }
    // Якщо відбувся різкий стрибок
    else {
        moistureAnomalyCounter++;
```

```
        Serial.printf("[WARNING] Стрибок вологості. Підозра на перешкоду: %d/2\n", moistureAnomalyCounter);


        // Якщо "стрибок" тримається вже 2 цикли підряд (10 секунд) - це вірні дані
        if (moistureAnomalyCounter >= 2) {
            soilMoisturePercent = newMoisture; // Приймаємо нову реальність
            moistureAnomalyCounter = 0; // Скидаємо лічильник
            Serial.println("[INFO] Нова вологість підтверджена! Фільтр скинуто.");
        }
    }


    sensors.requestTemperatures();
    float newSoilTemp = sensors.getTempCByIndex(0);
    if (newSoilTemp > -50.0) {
      soilTemp = newSoilTemp;
    }
    lux = lightMeter.readLightLevel();


    // ЕКРАН
    lcd.setCursor(0, 0);
    lcd.print("T:"); lcd.print((int)round(t)); lcd.write(byte(0)); lcd.print("C  ");
    lcd.print("H:"); lcd.print((int)round(h)); lcd.print("% ");


    lcd.setCursor(0, 1);
    lcd.print("S:"); lcd.print((int)round(soilTemp)); lcd.write(byte(0)); lcd.print("C  ");
    lcd.print("P:"); lcd.print(currentPeltierPWM); lcd.print(" ");


    // ВІДПРАВКА ДАНИХ ТА СТАНУ АКТУАТОРІВ
    if (client.connected()) {
      StaticJsonDocument<512> doc;
      JsonObject sensors_json = doc.createNestedObject("sensors");
      sensors_json["temperature"] = t;
      sensors_json["humidity"] = h;
      sensors_json["soil_moisture"] = soilMoisturePercent;
      sensors_json["light_lux"] = lux;
      sensors_json["soil_temp_c"] = soilTemp;


      JsonObject actuators_json = doc.createNestedObject("actuators");
      actuators_json["peltier_pwm"] = currentPeltierPWM;


      // Відправляємо "Пам'ять" коротких імпульсів
      actuators_json["pump_status"] = wasPumpActive ? 1 : 0;
      actuators_json["humidifier_status"] = wasHumidifierActive ? 1 : 0;
      actuators_json["window_angle"] = currentWindowAngle;
      actuators_json["led_pwm"] = currentLedPWM;


      char jsonBuffer[512];
      serializeJson(doc, jsonBuffer);

client.publish("greenhouse/telemetry", jsonBuffer);


      // ОЧИЩАЄМО ПАМ'ЯТЬ після відправки
```

```
      wasPumpActive = false;
      wasHumidifierActive = false;
    }
  }

  // --- ЛОКАЛЬНЕ КЕРУВАННЯ ТА FUZZY (Кожні 5 сек) ---
  if (currentMillis - lastActuatorTime >= 5000) {
    lastActuatorTime = currentMillis;

    // 1. КВАТИРКА (Сервопривід SG90 + Прогнозуюче керування)

    // СИТУАЦІЯ А: Вже спекотно АБО дуже волого -> ВІДКРИТИ ПОВНІСТЮ (110°)
    if (t >= target_air_temp + 2.0 || h >= target_air_hum + 10.0) {
      myServo.write(110);
      currentWindowAngle = 110;
    }
    // СИТУАЦІЯ Б: Зараз Норма (або трохи тепліше цілі), АЛЕ ARIMA прогнозує спеку -> ПРИВІДКРИТИ (60°)
    else if (t >= target_air_temp && predicted_air_t >= target_air_temp + 2.0) {
      myServo.write(60);
      currentWindowAngle = 60;
      Serial.println("[PREDICT] Кватирка привідкрита на 60° (Очікується спека)");
    }
    // СИТУАЦІЯ В: Зараз Норма/Холодно і ARIMA не прогнозує спеки -> ЗАКРИТИ (10°)
    else if (t <= target_air_temp && h <= target_air_hum) {
      myServo.write(10);
      currentWindowAngle = 10;
    }

    // 2. Зволожувач
    if (h <= target_air_hum - 5.0) triggerHumidifier();

    // 3. Помпа
    if (soilMoisturePercent <= target_soil_moist - 5.0 && !pumpActive) triggerPump();

    // 4. Освітлення
    if (lux <= target_light_lux - 300) {
      digitalWrite(TRANSISTOR_LED, HIGH); currentLedPWM = 1023;
    } else if (lux >= target_light_lux) {
      digitalWrite(TRANSISTOR_LED, LOW); currentLedPWM = 0;
    }

    // 5. ПРОГНОЗУЮЧА НЕЧІТКА ЛОГІКА ДЛЯ ПЕЛЬТЬЄ (ARIMA + FUZZY)
    if (!isnan(soilTemp)) {

      // 1. Рахуємо поточне відхилення
      float current_error = target_soil_temp - soilTemp;

      // 2. Механізм виживання (TTL для прогнозу ARIMA = 60 хвилин)
      float active_prediction = predicted_soil_t;
      if (millis() - lastForecastTime > 3600000 || lastForecastTime == 0) {
        // Якщо прогнозу немає або він старий - прирівнюємо прогноз до поточної реальності
        active_prediction = soilTemp;
      }

      // 3. Рахуємо майбутнє відхилення
```

```
      float pred_error = target_soil_temp
- active_prediction;


      // 4. Подаємо обидва входи у
"Мозок"
      fuzzy->setInput(1, current_error);
      fuzzy->setInput(2, pred_error);
      fuzzy->fuzzify();


      // 5. Отримуємо ідеальний ШІМ і
застосовуємо
      currentPeltierPWM = fuzzy-
>defuzzify(1);
      setPeltierPWM(currentPeltierPWM);


      Serial.printf("[PREDICT-FUZZY]
SoilTemp: %.1fC | ARIMA: %.1fC | Ціль:
%.1fC | -> PWM: %d\n",
                    soilTemp,
active_prediction, target_soil_temp,
currentPeltierPWM);
    }
  }
  // Постійно ловимо статуси помпи і
зволожувача для графіків
  if (pumpActive) wasPumpActive = true;
  if (humidifierToggleActive)
wasHumidifierActive = true;
}
```

ДОДАТОК Б
(обов'язковий)

Лістинг вихідного коду серверної частини системи керування та модуля прогнозуючої аналітики

app.py:

```
from flask import Flask, render_template, request, redirect, url_for, session, flash, jsonify
import paho.mqtt.client as mqtt
import mysql.connector
import json
from datetime import datetime, timedelta
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
import warnings
from statsmodels.tools.sm_exceptions import ConvergenceWarning
from apscheduler.schedulers.background import BackgroundScheduler

warnings.simplefilter('ignore', ConvergenceWarning)
warnings.filterwarnings("ignore")

app = Flask(__name__)
app.secret_key = "super_secret_thesis_key"

ADMIN_USER = "admin"
ADMIN_PASS = "diploma2026"

MQTT_BROKER = "localhost"
MQTT_PORT = 1883
TOPIC_CONFIG = "greenhouse/config"
TOPIC_TELEMETRY = "greenhouse/telemetry"
TOPIC_FORECAST = "greenhouse/forecast/arima"

DB_CONFIG = {
    'host': 'localhost',
    'user': 'iot_user',
    'password': 'Gh_SecurePass2026!',
    'database': 'greenhouse_db'
}

def get_db_connection():
    return mysql.connector.connect(**DB_CONFIG)

def run_arima_forecast():
    print("\n[ARIMA] Запуск математичного прогнозування...")
    try:
        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)
        # Беремо дані за останні 2 години (цього достатньо для тенденції)
        cursor.execute("""
            SELECT timestamp, temperature_c, humidity_perc, soil_temp_c
            FROM sensor_telemetry
            WHERE timestamp >= NOW() - INTERVAL 2 HOUR
            ORDER BY timestamp ASC
        """)
        rows = cursor.fetchall()
```

```
        if len(rows) < 30:
            print("[ARIMA] Недостатньо
даних. Пропускаємо.")
            return

        # 1. ПЕРЕТВОРЮЄМО ДАНІ В PANDAS
        df = pd.DataFrame(rows)
        df['timestamp'] =
pd.to_datetime(df['timestamp'])
        df.set_index('timestamp',
inplace=True)

        # 2. РЕСЕМПЛІНГ (Агрегація по 1
хвилині)
        # Прибирає шум 5-секундних
датчиків
        df_resampled =
df.resample('1min').mean().ffill()

        # ФУНКЦІЯ БЕЗПЕЧНОЇ ARIMA
        def safe_arima(data_series):
            if data_series.std() < 0.1:
                return
round(float(data_series.iloc[-1]), 2)
            else:
                # Прогнозуємо на 30
КРОКІВ вперед (30 хвилин) і беремо
останнє значення [-1]
                model =
ARIMA(data_series.values, order=(2, 1,
2), enforce_stationarity=False,
enforce_invertibility=False)
                forecast_array =
model.fit().forecast(steps=30)
                return
round(float(forecast_array[-1]), 2)

        # 3. Рахуємо прогнози
        pred_t =
safe_arima(df_resampled['temperature_c'])
        pred_h =
safe_arima(df_resampled['humidity_perc'])
        pred_h = max(0.0, min(100.0,
pred_h))
        pred_st =
safe_arima(df_resampled['soil_temp_c'])

        target_time = datetime.now() +
timedelta(minutes=30)

        print(f"[ARIMA] Прогноз на 30 хв:
T_пов={pred_t}°C | Волог={pred_h}% |
T_грунт={pred_st}°C")

        cursor.execute("""INSERT INTO
arima_predictions
                          (target_time,
predicted_temp, predicted_hum,
predicted_soil_temp)
                          VALUES (%s, %s,
%s, %s)""",
                       (target_time,
pred_t, pred_h, pred_st))
        conn.commit()

        forecast_payload =
{"predicted_temp": pred_t,
"predicted_hum": pred_h,
"predicted_soil_temp": pred_st}
        client.publish(TOPIC_FORECAST,
json.dumps(forecast_payload),
retain=True)

    except Exception as e:
        print(f"[ARIMA ERROR] Помилка:
{e}")
    finally:
        if 'conn' in locals() and
conn.is_connected():
            cursor.close()
            conn.close()
```

```
def on_connect(client, userdata, flags,
rc):
    client.subscribe(TOPIC_TELEMETRY)

def on_message(client, userdata, msg):
    try:
        if msg.topic == TOPIC_TELEMETRY:
            data =
json.loads(msg.payload.decode('utf-8'))
            sensors = data.get("sensors",
{})
            actuators =
data.get("actuators", {}) # ЛОВИМО
АКТУАТОРИ

            conn = get_db_connection()
            cursor = conn.cursor()

            # Пишемо сенсори
            cursor.execute("""INSERT INTO
sensor_telemetry
                    (temperature_c,
humidity_perc, soil_moisture, light_lux,
soil_temp_c)
                    VALUES (%s, %s, %s,
%s, %s)""",

(sensors.get("temperature"),
sensors.get("humidity"),
sensors.get("soil_moisture"),
sensors.get("light_lux"),
sensors.get("soil_temp_c")))

            # Пишемо актуатори
            if actuators:
                cursor.execute("""INSERT
INTO actuator_logs
                        (peltier_pwm,
pump_status, humidifier_status,
window_angle, led_pwm)
                                VALUES (%s, %s,
%s, %s, %s)""",

(actuators.get("peltier_pwm", 0),
actuators.get("pump_status", 0),

actuators.get("humidifier_status", 0),
actuators.get("window_angle", 10),

actuators.get("led_pwm", 0)))

            conn.commit()
            cursor.close()
            conn.close()
    except Exception as e:
        pass

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(MQTT_BROKER, MQTT_PORT,
60)
client.loop_start()

scheduler = BackgroundScheduler()
scheduler.add_job(func=run_arima_forecast
, trigger="interval", minutes=10,
max_instances=2)
scheduler.start()

@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        if request.form['username'] ==
ADMIN_USER and request.form['password']
== ADMIN_PASS:
            session['logged_in'] = True
            return
redirect(url_for('dashboard'))
    return render_template('login.html')
```

```
@app.route('/dashboard', methods=['GET',
'POST'])
def dashboard():
    if not session.get('logged_in'):
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    if request.method == 'POST':
        t_air_t =
request.form.get('target_air_temp',
type=float)
        t_air_h =
request.form.get('target_air_hum',
type=float)
        t_soil_t =
request.form.get('target_soil_temp',
type=float)
        t_soil_m =
request.form.get('target_soil_moisture',
type=float)
        t_light =
request.form.get('target_light_lux',
type=float)

        cursor.execute("INSERT INTO
plant_requirements (target_air_temp,
target_air_hum, target_soil_temp,
target_soil_moisture, target_light_lux)
VALUES (%s, %s, %s, %s, %s)", (t_air_t,
t_air_h, t_soil_t, t_soil_m, t_light))
        conn.commit()

        client.publish(TOPIC_CONFIG,
json.dumps({
            "target_air_temp": t_air_t,
"target_air_hum": t_air_h,
"target_soil_temp": t_soil_t,
"target_soil_moisture": t_soil_m,
"target_light_lux": t_light
        }), retain=True)

    cursor.execute("SELECT * FROM
plant_requirements ORDER BY id DESC LIMIT
1")
    current_req = cursor.fetchone() or
{'target_air_temp': 25.0,
'target_air_hum': 50.0,
'target_soil_temp': 22.0,
'target_soil_moisture': 40.0,
'target_light_lux': 2000.0}

    cursor.execute("SELECT * FROM
arima_predictions ORDER BY id DESC LIMIT
1")
    last_pred = cursor.fetchone()

    cursor.execute("SELECT timestamp,
temperature_c, humidity_perc,
soil_temp_c, soil_moisture, light_lux
FROM sensor_telemetry WHERE timestamp >=
NOW() - INTERVAL 12 HOUR ORDER BY
timestamp ASC")
    telemetry = cursor.fetchall()

    # Витягуємо дані актуаторів
    cursor.execute("SELECT timestamp,
peltier_pwm, pump_status,
humidifier_status, window_angle, led_pwm
FROM actuator_logs WHERE timestamp >=
NOW() - INTERVAL 12 HOUR ORDER BY
timestamp ASC")
    act_logs = cursor.fetchall()

    cursor.close()
    conn.close()

    chart_data = {

"labels":[row['timestamp'].strftime('%H:%
M') for row in telemetry],
```

```
        "air_temp": [row['temperature_c']
for row in telemetry],
        "soil_temp": [row['soil_temp_c']
for row in telemetry],
        "air_hum": [row['humidity_perc']
for row in telemetry],
        "soil_moist":
[row['soil_moisture'] for row in
telemetry],
        "light_lux":[row['light_lux'] for
row in telemetry],

        # Дані актуаторів

"act_labels":[row['timestamp'].strftime('
%H:%M') for row in act_logs],
        "peltier_pwm":
[row['peltier_pwm'] for row in act_logs],
        "pump_status":
[row['pump_status'] for row in act_logs],
        "humidifier_status":
[row['humidifier_status'] for row in
act_logs],
        "window_angle":
[row['window_angle'] for row in
act_logs],
        "led_pwm": [row['led_pwm'] for
row in act_logs]
    }

    return
render_template('dashboard.html',
req=current_req, chart_data=chart_data,
prediction=last_pred)

@app.route('/logout')
def logout():
    session.clear()
    return redirect(url_for('login'))

@app.route('/api/chart_data')
def api_chart_data():
    if not session.get('logged_in'):
        return jsonify({'error':
'unauthorized'}), 401

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Витягуємо телеметрію
    cursor.execute("SELECT timestamp,
temperature_c, humidity_perc,
soil_temp_c, soil_moisture, light_lux
FROM sensor_telemetry WHERE timestamp >=
NOW() - INTERVAL 12 HOUR ORDER BY
timestamp ASC")
    telemetry = cursor.fetchall()

    # Витягуємо актуатори
    cursor.execute("SELECT timestamp,
peltier_pwm, pump_status,
humidifier_status, window_angle, led_pwm
FROM actuator_logs WHERE timestamp >=
NOW() - INTERVAL 12 HOUR ORDER BY
timestamp ASC")
    act_logs = cursor.fetchall()

    cursor.execute("SELECT target_time,
predicted_temp, predicted_hum,
predicted_soil_temp FROM
arima_predictions ORDER BY id DESC LIMIT
1")
    last_pred = cursor.fetchone()

    cursor.close()
    conn.close()

    # Пакуємо в JSON для графіків
    response_data = {
        "labels":
[row['timestamp'].strftime('%H:%M') for
row in telemetry],
```

```
        "air_temp": [row['temperature_c'] for row in telemetry],
        "soil_temp": [row['soil_temp_c'] for row in telemetry],
        "air_hum":[row['humidity_perc'] for row in telemetry],
        "soil_moist": [row['soil_moisture'] for row in telemetry],
        "light_lux": [row['light_lux'] for row in telemetry],
        "act_labels": [row['timestamp'].strftime('%H:%M') for row in act_logs],
        "peltier_pwm": [row['peltier_pwm'] for row in act_logs],
        "pump_status": [row['pump_status'] for row in act_logs],
        "humidifier_status": [row['humidifier_status'] for row in act_logs],
        "window_angle": [row['window_angle'] for row in act_logs],
        "led_pwm":[row['led_pwm'] for row in act_logs],
        "prediction": None
    }

    if last_pred:
        response_data["prediction"] = {
            "time": last_pred['target_time'].strftime('%H:%M') if last_pred['target_time'] else "",
            "air_t": last_pred['predicted_temp'],
            "air_h": last_pred['predicted_hum'],
            "soil_t": last_pred['predicted_soil_temp']
        }

    return jsonify(response_data)

if __name__ == '__main__':
    from waitress import serve
    print("🚀 Запуск Production сервера (Waitress) на порту 5000...")
    serve(app, host='0.0.0.0', port=5000, threads=4)
```

ДОДАТОК В
(обов'язковий)

Лістинг вихідного коду серверної частини системи керування та модуля предиктивної аналітики

login.html:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Вхід - Greenhouse Cloud</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light d-flex align-items-center" style="height: 100vh;">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-4">
                <div class="card shadow">
                    <div class="card-header bg-success text-white text-center">
                        <h4>Greenhouse Cloud IoT</h4>
                    </div>
                    <div class="card-body">
                        {% with messages = get_flashed_messages() %}
                          {% if messages %}
                            <div class="alert alert-danger">{{ messages[0] }}</div>
                          {% endif %}
                        {% endwith %}
                        <form method="POST">
                            <div class="mb-3">
                                <label>Логін:</label>
                                <input type="text" name="username" class="form-control" required>
                            </div>
                            <div class="mb-3">
                                <label>Пароль:</label>
                                <input type="password" name="password" class="form-control" required>
                            </div>
                            <button type="submit" class="btn btn-success w-100">Увійти</button>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

dashboard.html:

```
<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <title>Панель Керування Теплицею</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body class="bg-light">

    <nav class="navbar navbar-dark bg-success mb-4">

        <div class="container">

            <a class="navbar-brand" href="#">Cloud-Edge Greenhouse</a>

            <a href="/logout" class="btn btn-outline-light btn-sm">Вийти</a>

        </div>

    </nav>

    <div class="container-fluid px-4">

        <div class="row">



            <div class="col-md-4">

                <div class="card shadow border-0 mb-4">

                    <div class="card-header bg-primary text-white"><h6 class="mb-0">Поточні вимоги рослини</h6></div>

                    <div class="card-body bg-white">

                        <ul class="list-group list-group-flush">

                            <li class="list-group-item">Повітря: <b>{{ req.target_air_temp }}°C</b> | <b>{{ req.target_air_hum }}%</b></li>

                            <li class="list-group-item">Ґрунт: <b>{{ req.target_soil_temp }}°C</b> | <b>{{ req.target_soil_moisture }}%</b></li>

                            <li class="list-group-item">Освітлення: <b>{{ req.target_light_lux }} Lux</b></li>

                        </ul>

                    </div>

                </div>



                <div class="card shadow border-0 mb-4 border-start border-5 border-info">

                    <div class="card-header bg-white text-info fw-bold">

                        ШІ Прогноз (ARIMA)

                    </div>

                    <div class="card-body bg-white">

                        {% if prediction %}

                            <p class="text-center text-muted mb-3 small border-bottom pb-2">Очікується на: <b id="pred_time">{{ prediction.target_time.strftime('%H:%M') }}</b></p>

                            <div class="d-flex justify-content-between text-center">

                                <div>
```

```
                                    <h5
class="text-danger mb-0"
id="pred_air_t">{{
prediction.predicted_temp }}°C</h5>

<small class="text-muted">Повітря</small>
                                </div>
                                <div>
                                    <h5
class="text-primary mb-0"
id="pred_air_h">{{
prediction.predicted_hum }}%</h5>

<small class="text-
muted">Вологість</small>
                                </div>
                                <div>
                                    <h5
class="text-warning mb-0"
id="pred_soil_t">{{
prediction.predicted_soil_temp }}°C</h5>

<small class="text-muted">Ґрунт</small>
                                </div>
                            </div>
                        {% else %}
                            <p
class="text-center text-muted mb-0">Зібр
даних для моделі...</p>
                        {% endif %}
                    </div>
                </div>

                <div class="card shadow
border-0 mb-4">
                    <div class="card-
header bg-dark text-white"><h6 class="mb-
0">⚙ Змінити вимоги</h6></div>
                    <div class="card-body
bg-white">
                        {% with messages
= get_flashed_messages() %}{% if messages
%}<div class="alert alert-success">{{
messages[0] }}</div>{% endif %}{% endwith
%}
                        <form
method="POST">

<label>Температура повітря
(°C):</label><input type="number"
step="0.1" name="target_air_temp"
class="form-control mb-2" value="{{
req.target_air_temp }}" required>

<label>Вологість повітря
(%):</label><input type="number" step="1"
name="target_air_hum" class="form-control
mb-2" value="{{ req.target_air_hum }}"
required>

<label>Температура ґрунту
(°C):</label><input type="number"
step="0.1" name="target_soil_temp"
class="form-control mb-2" value="{{
req.target_soil_temp }}" required>

<label>Вологість ґрунту
(%):</label><input type="number" step="1"
name="target_soil_moisture" class="form-
control mb-2" value="{{
req.target_soil_moisture }}" required>

<label>Освітлення (Lux):</label><input
type="number" step="1"
name="target_light_lux" class="form-
control mb-3" value="{{
req.target_light_lux }}" required>

                            <button
type="submit" class="btn btn-success w-
100">Застосувати</button>
                        </form>
                    </div>
                </div>
```

```
            </div>


            <!-- ПРАВА КОЛОНКА З 3 ГРАФІКАМИ -->
            <div class="col-md-8">
                <div class="card shadow border-0 mb-4">
                    <div class="card-header bg-danger text-white"><h6 class="mb-0">Динаміка температур</h6></div>
                    <div class="card-body"><canvas id="tempChart" height="80"></canvas></div>
                </div>
                <div class="card shadow border-0 mb-4">
                    <div class="card-header bg-info text-white"><h6 class="mb-0">Динаміка вологості</h6></div>
                    <div class="card-body"><canvas id="humChart" height="80"></canvas></div>
                </div>
                <div class="card shadow border-0">
                    <div class="card-header bg-warning text-dark"><h6 class="mb-0">Динаміка освітлення (Lux)</h6></div>
                    <div class="card-body"><canvas id="lightChart" height="80"></canvas></div>
                </div>
                <div class="card shadow border-0 mt-4">
                    <div class="card-header bg-secondary text-white"><h6 class="mb-0">Витрати енергії (Актуатори)</h6></div>
                    <div class="card-body"><canvas id="actuatorChart" height="100"></canvas></div>
                </div>
            </div>
        </div>
    </div>


    <script>
        const chartData = {{ chart_data | tojson | safe }};

        // 1. Ініціалізація графіків
        const ctxTemp = document.getElementById('tempChart').getContext('2d');
        const myTempChart = new Chart(ctxTemp, { type: 'line', data: { labels: chartData.labels, datasets:[
            { label: 'Повітря (°C)', data: chartData.air_temp, borderColor: 'red', tension: 0.1 },
            { label: 'Ґрунт (°C)', data: chartData.soil_temp, borderColor: 'orange', tension: 0.1 }
        ]}});

        const ctxHum = document.getElementById('humChart').getContext('2d');
        const myHumChart = new Chart(ctxHum, { type: 'line', data: { labels: chartData.labels, datasets:[
            { label: 'Повітря (%)', data: chartData.air_hum, borderColor: 'blue', tension: 0.1 },
```

```
            { label: 'Ґрунт (%)', data:
chartData.soil_moist, borderColor:
'teal', tension: 0.1 }

        ]}});



        const ctxLight =
document.getElementById('lightChart').get
Context('2d');

        const myLightChart = new
Chart(ctxLight, { type: 'line', data: {
labels: chartData.labels, datasets:[

            { label: 'Освітлення (Lux)',
data: chartData.light_lux, borderColor:
'#ffc107', backgroundColor: 'rgba(255,
193, 7, 0.2)', fill: true, tension: 0.1 }

        ]}});



        const ctxAct =
document.getElementById('actuatorChart').
getContext('2d');

        const myActChart = new
Chart(ctxAct, {

            type: 'line',

            data: {

                labels:
chartData.act_labels,

                datasets:[

                    { label: 'Пельтьє
(ШІМ)', data: chartData.peltier_pwm,
borderColor: 'purple', backgroundColor:
'rgba(128, 0, 128, 0.1)', fill: true,
tension: 0.1, yAxisID: 'y_pwm' },

                    { label: 'LED Світло
(ШІМ)', data: chartData.led_pwm,
borderColor: 'yellow', backgroundColor:
'rgba(255, 206, 86, 0.1)', fill: true,
tension: 0.1, yAxisID: 'y_pwm' },

                    { label: 'Кватирка
(Кут 0-180°)', data:
chartData.window_angle, borderColor:
'green', stepped: true, yAxisID:
'y_angle' },

                    { label: 'Помпа',
data: chartData.pump_status, borderColor:
'blue', stepped: true, yAxisID:
'y_status' },

                    { label:
'Зволожувач', data:
chartData.humidifier_status, borderColor:
'cyan', stepped: true, yAxisID:
'y_status' }

                ]

            },

            options: {

                scales: {

                    y_pwm: { type:
'linear', position: 'left', min: 0, max:
1050, title: {display: true, text: 'ШІМ'}
},

                    y_angle: { type:
'linear', position: 'right', min: 0, max:
180, title: {display: true, text: 'Кут'}
},

                    y_status: { type:
'linear', position: 'right', min: 0, max:
2, ticks: { stepSize: 1 }, display: false
}

                }

            }

        });



        // 2. АСИНХРОННЕ АВТООНОВЛЕННЯ
(AJAX) кожні 5 секунд

        setInterval(() => {

            fetch('/api/chart_data')

                .then(response =>
response.json())

                .then(data => {
```

```
                    if (data.error)
return; // Якщо сесія закінчилася


                    // Оновлюємо дані
графіків

myTempChart.data.labels = data.labels;

myTempChart.data.datasets[0].data =
data.air_temp;

myTempChart.data.datasets[1].data =
data.soil_temp;
                    myTempChart.update();



myHumChart.data.labels = data.labels;

myHumChart.data.datasets[0].data =
data.air_hum;

myHumChart.data.datasets[1].data =
data.soil_moist;
                    myHumChart.update();



myLightChart.data.labels = data.labels;

myLightChart.data.datasets[0].data =
data.light_lux;

myLightChart.update();



myActChart.data.labels = data.act_labels;

myActChart.data.datasets[0].data =
data.peltier_pwm;

myActChart.data.datasets[1].data =
data.led_pwm;

myActChart.data.datasets[2].data =
data.window_angle;

myActChart.data.datasets[3].data =
data.pump_status;

myActChart.data.datasets[4].data =
data.humidifier_status;
                    myActChart.update();


                    if (data.prediction)
{

document.getElementById('pred_time').inne
rText = data.prediction.time;

document.getElementById('pred_air_t').inn
erText = data.prediction.air_t + '°C';

document.getElementById('pred_air_h').inn
erText = data.prediction.air_h + '%';

document.getElementById('pred_soil_t').in
nerText = data.prediction.soil_t + '°C';
                    }
                });
        }, 5000); // 5000 мс = 5 секунд
    </script>
</body>
</html>
```

ДОДАТОК Г
(обов’язковий)

Лістинг SQL-скриптів ініціалізації структури реляційної бази даних MySQL

```
-- Створення бази даних

CREATE DATABASE IF NOT EXISTS greenhouse_db CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;

USE greenhouse_db;


-- Таблиця 1: Зберігання часових рядів сенсорної телеметрії

CREATE TABLE IF NOT EXISTS sensor_telemetry (

    id INT AUTO_INCREMENT PRIMARY KEY,

    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    temperature_c FLOAT NOT NULL,

    humidity_perc FLOAT NOT NULL,

    soil_moisture INT NOT NULL,

    light_lux FLOAT NOT NULL,

    soil_temp_c FLOAT NOT NULL,

    INDEX idx_timestamp (timestamp) -- Індекс для швидкого пошуку ARIMA

);


-- Таблиця 2: Журналювання станів виконавчих механізмів (актуаторів)

CREATE TABLE IF NOT EXISTS actuator_logs (

    id INT AUTO_INCREMENT PRIMARY KEY,

    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

    peltier_pwm INT NOT NULL,

    pump_status TINYINT(1) NOT NULL,

    humidifier_status TINYINT(1) NOT NULL,

    window_angle INT NOT NULL,

    led_pwm INT NOT NULL,

    INDEX idx_timestamp (timestamp)

);
```

```
-- Таблиця 3: Зберігання результатів прогнозування моделі ARIMA
CREATE TABLE IF NOT EXISTS arima_predictions (
    id INT AUTO_INCREMENT PRIMARY KEY,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    target_time DATETIME NOT NULL,
    predicted_temp FLOAT NOT NULL,
    predicted_hum FLOAT NOT NULL,
    predicted_soil_temp FLOAT NOT NULL,
    INDEX idx_target_time (target_time)
);

-- Таблиця 4: Динамічні цільові вимоги рослини (уставки від оператора)
CREATE TABLE IF NOT EXISTS plant_requirements (
    id INT AUTO_INCREMENT PRIMARY KEY,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    target_air_temp FLOAT NOT NULL,
    target_air_hum FLOAT NOT NULL,
    target_soil_temp FLOAT NOT NULL,
    target_soil_moisture FLOAT NOT NULL,
    target_light_lux FLOAT NOT NULL
);

-- Спеціальний користувач для IoT системи та надання прав
CREATE USER IF NOT EXISTS 'iot_user'@'localhost' IDENTIFIED BY 'Gh_SecurePass2026!';
GRANT ALL PRIVILEGES ON greenhouse_db.* TO 'iot_user'@'localhost';
FLUSH PRIVILEGES;
```

ДОДАТОК Ґ
(обов’язковий)

Публікація

УДК 004.8:681.5:631.2

СИСТЕМА КЕРУВАННЯ МІКРОКЛІМАТОМ ТЕПЛИЦІ З ПРОГНОЗУВАННЯМ ТА ОПТИМІЗАЦІЄЮ РЕЖИМІВ РОБОТИ

Лісовий В. М., студент гр. КІ2м-24-1
Науковий керівник: д. т. н., професор Яцків В. В.
Хмельницький Національний Університет

Актуальність. Сучасні теплиці є складними нелінійними системами. Традиційні методи неефективно компенсують зовнішні збурення, що спричиняє температурний стрес рослин і перевитрати енергії.

Метою роботи є розробка енергоефективної відмовостійкої системи керування мікрокліматом теплиці на базі Cloud-Edge архітектури з використанням предиктивних моделей та нечіткої логіки.

Аналіз існуючих рішень. ПІД-регулятори схильні до перерегулювання та конфлікту виконавчих механізмів. Суто хмарні IoT-рішення критично залежать від інтернет-з'єднання, втрата якого є неприпустимою для біотехнічних об'єктів.

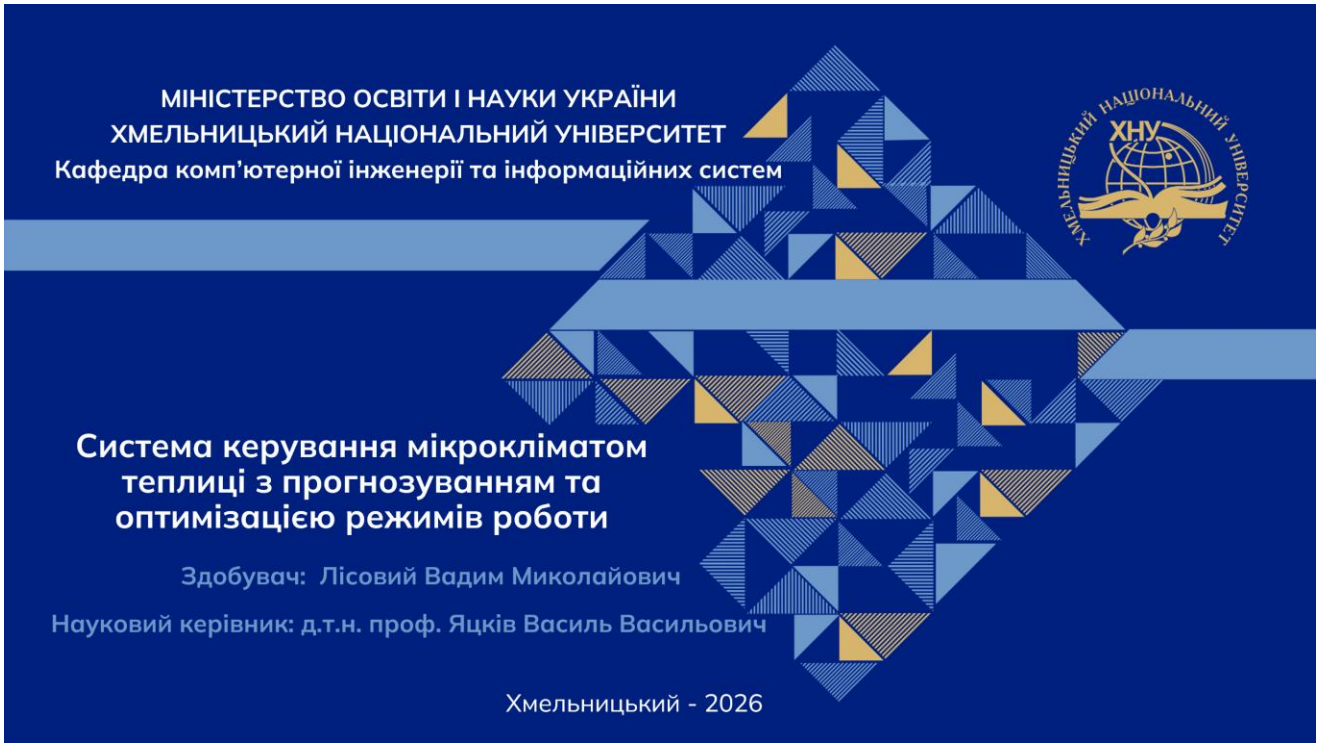
Результати. Пропонується розробити трирівневу Cloud-Edge архітектуру. У хмарі планується розгорнути предиктивну модель ARIMA для генерації прогнозу. На рівні ESP32 доцільно синтезувати нечіткий регулятор (Fuzzy Logic). Застосування правил Мамдані дозволить усунути перерегулювання та взаємовплив параметрів, а алгоритм Offline Survivability забезпечить автономне локальне керування при втраті зв'язку з сервером.

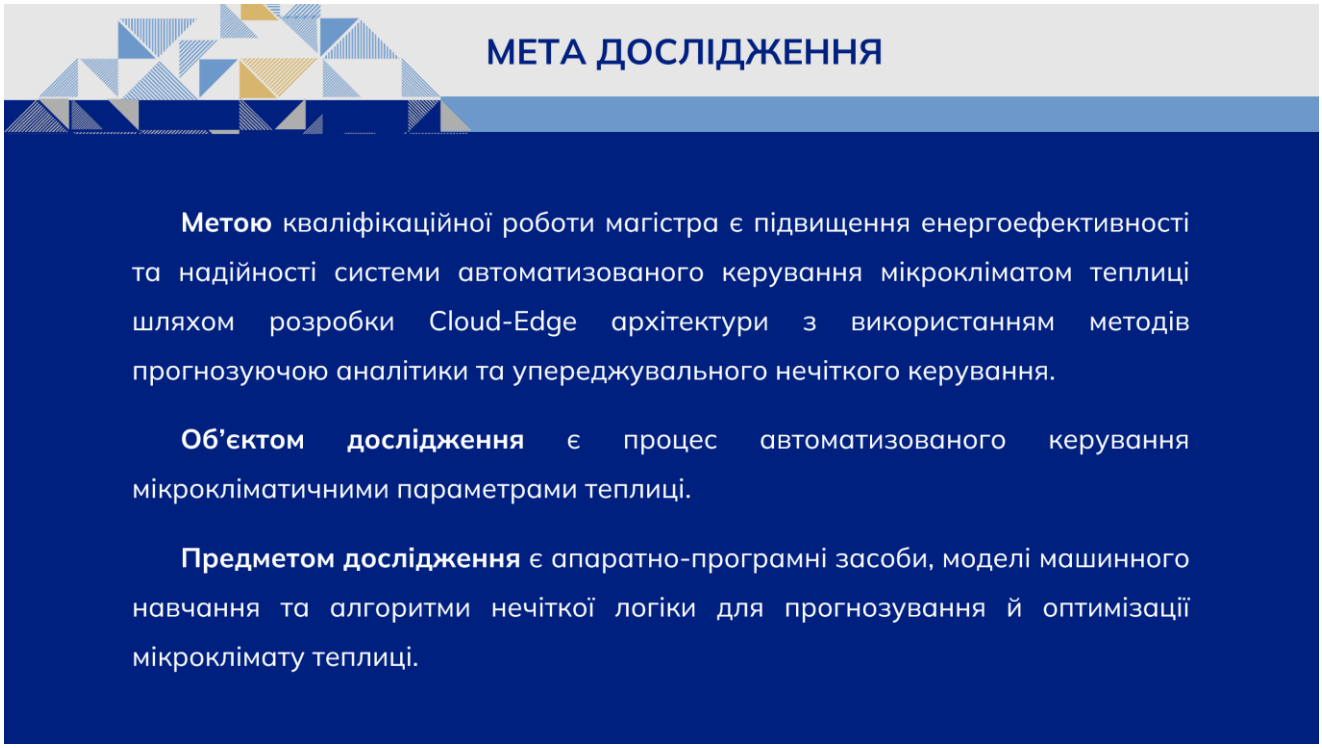
Крім того, передбачається проектування підсистеми живлення з ізоляцією логіки від актуаторів через понижуючі DC-DC перетворювачі. Механізм 10-бітного ШІМ-обмеження гарантуватиме безпечне функціонування системи під час пікових навантажень, які перевищують номінал джерела живлення, усуваючи ризик апаратних збоїв.

Висновки. Запропонована система вирішує проблему взаємовпливу параметрів та забезпечує високу апаратну стійкість. Синергія предиктивного моделювання у хмарі та нечіткого керування на Edge-рівні оптимізує енергоресурси у точному землеробстві.

ДОДАТОК Д
(обов'язковий)

Презентація





ЗАДАЧІ ДОСЛІДЖЕННЯ

Поставлена мета досягається розв'язанням таких основних завдань:

- провести аналіз існуючих методів та технологій керування мікрокліматом в агропромисловому секторі;
- розробити метод та математичне забезпечення системи: розробити прогнозуючу модель машинного навчання для аналізу часових рядів та двопортовий нечіткий регулятор керування "за відхиленням";
- розробити програмне забезпечення хмарного сервера та мікроконтролера;
- провести експериментальне дослідження розробленого методу для оцінки точності прогнозування та ефективності алгоритмів відмовостійкості.

НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Наукова новизна отриманих результатів:

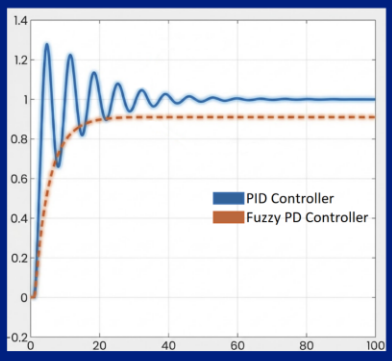
- набув подальшого розвитку метод прогнозуючого керування мікрокліматом, який, на відміну від існуючих, використовує двопортовий нечіткий регулятор, що дозволяє реалізувати упереджувальне керування інерційними виконавчими механізмами;
- набула подальшого розвитку інформаційна технологія забезпечення відмовостійкості в IoT-системах за рахунок алгоритму безпечної деградації прогнозу та збереження уставок у незалежній пам'яті, що гарантує автономне виживання біологічних об'єктів при втраті зв'язку.

Практична цінність отриманих результатів полягає у створенні готового до дослідно-промислової експлуатації програмно-апаратного комплексу.

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

- В умовах зростання енергоємності сучасних тепличних комплексів та ускладнення процесів підтримання мікроклімату, виникає гостра потреба у переході від класичних реактивних схем керування до інтелектуальних систем на базі технологій Інтернету речей та штучного інтелекту.
- Впровадження гібридних Cloud-Edge архітектур дозволяє поєднати потужну хмарну прогнозуючу аналітику з високою локальною надійністю мікроконтролерів, проте вимагає розробки спеціалізованих методів для компенсації значної теплової інерційності та нелінійності об'єкта.
- Ключовою проблемою стає забезпечення енергоефективності та відмовостійкості системи в умовах стохастичних зовнішніх збурень та можливих мережевих збоїв, що зумовлює необхідність створення алгоритмів упереджувального керування та механізмів автономного виживання системи.

АНАЛІЗ ВІДОМИХ МЕТОДІВ



Графік порівняння перехідних процесів при використанні звичайного ПІД-регулятора та нечіткого контролера

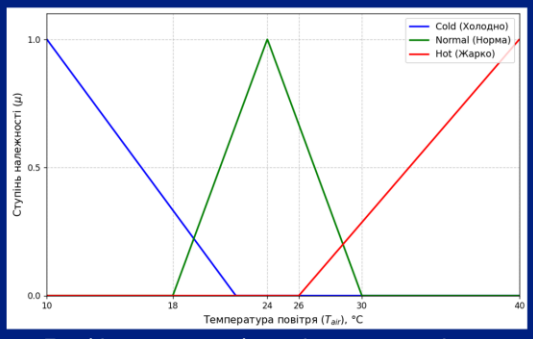
- Релейне керування (Вкл/Викл): Постійні циклічні вмикання актуаторів на повну потужність спричиняють різкі температурні коливання, що створює фізіологічний стрес для рослин. Такий режим роботи призводить до прискореного механічного зношування електронних компонентів.
- Класичні ПІД-регулятори: Через значну теплову інерційність теплиці та затримки в поширенні тепла виникає суттєве перерегулювання, яке в типових системах може сягати 15–20%. Це створює ризик короткочасного перегріву кореневої системи, що є критичним для рослин.
- Використання лінгвістичних правил Мамдані (нечітка логіка) дозволяє реалізувати плавне, асимптотичне наближення до уставки без динамічних стрибків потужності. Це забезпечує стабільний вихід керуючого сигналу на рівноважний стан, виключаючи автоколивання та знижуючи енерговитрати.

АНАЛІЗ ВІДОМИХ МЕТОДІВ

- Cloud-only: має ризики критичних затримок та зупинки системи при втраті зв'язку.
- Cloud-Edge: забезпечує автономність і локальну надійність керування.
- LSTM / Нейромережі: вимагають значних обчислювальних ресурсів і великих масивів даних для навчання.
- Модель ARIMA: гарантує високу точність прогнозу при мінімальному навантаженні на обчислювальні ресурси сервера.

Трирівнева архітектура IoT-системи

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ЛОКАЛЬНОГО НЕЧІТКОГО РЕГУЛЯТОРА



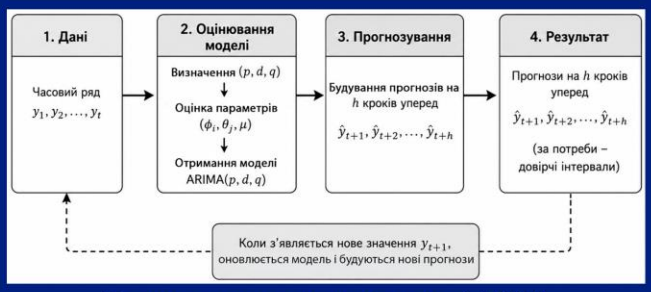
Графік трикутних функцій належності для похибки температури

Поточна T	Прогноз T_{ARIMA}	Дія Пельтьє (ШІМ)	Дія Сервоприводу (Кватирка)
Cold	Спадає	High (Макс. нагрів)	Closed (Закрито)
Cold	Зростає	Low (Підтримка)	Closed (Закрито)
Normal	Стабільно	Off (Вимкнено)	Closed (Закрито)
Normal	Зростає	Off (Вимкнено)	(Half-Open) (Привідкрито)
Hot	Зростає	Off (Вимкнено)	Open (Відкрито повністю)

Фрагмент бази правил нечіткого регулятора

- Метод логічного виведення: Алгоритм Мамдані.
- Форма функцій: Трикутні (найбільш обчислювально ефективні для ESP32).
- Упереджувальне керування: Використовує 2 входи (Поточна похибка + Прогнозована похибка від хмари).

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ARIMA



Структурна схема прогнозу моделі ARIMA

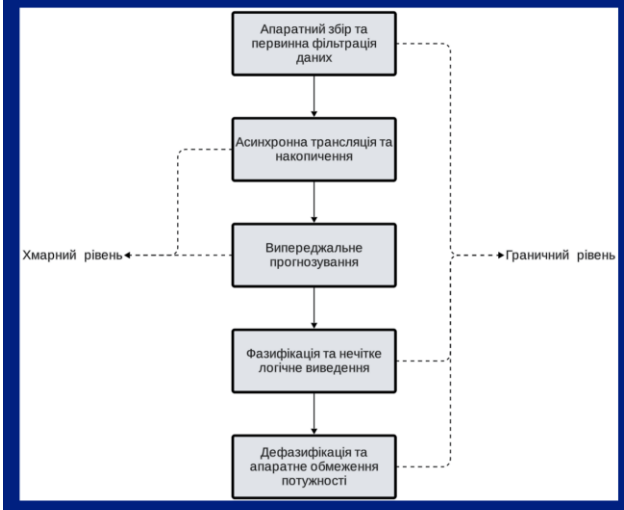
- Головна компактна формула ARIMA:

$$\phi(B)(1-B)^d Y_t = c + \theta(B)\varepsilon_t$$

- ϕ – коефіцієнти авторегресії ($p = 2$).
- B – оператор зсуву назад.
- d – порядок інтегрування ($d = 1$).
- Y_t – значення температури в поточний момент часу t.
- c – базовий рівень температури.
- θ – коефіцієнти ковзного середнього ($q = 2$).
- ε_t – випадкова похибка в поточний момент t.

- Модель: Авторегресійна інтегрована модель ковзного середнього.
- Оптимальна конфігурація: ARIMA (2, 1, 2).
- Оптимізація обчислень: Метод усереднення даних щохвилини для фільтрації високочастотного шуму сенсорів.

МЕТОД ПРОГНОЗУЮЧОГО ІНТЕЛЕКТУАЛЬНОГО КЕРУВАННЯ

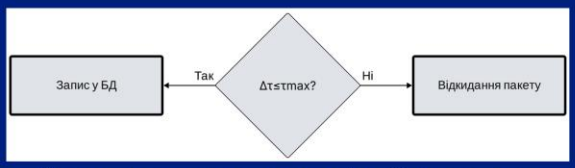


Блок-схема методу прогнозуючого інтелектуального керування

- Система розподіляє задачі між хмарним сервером для складного прогнозування моделлю ARIMA та граничним вузлом ESP32 для виконання алгоритмів нечіткої логіки в реальному часі.
- Двопортовий нечіткий регулятор одночасно обробляє поточну похибку від датчиків та прогноз із хмари, що дозволяє завчасно компенсувати теплову інерційність середовища.

МЕТОД ІНФОРМАЦІЙНОЇ ВЗАЄМОДІЇ ТА ВІДМОВОСТІЙКОСТІ

- Програмна архітектура мікроконтролера реалізована як неблокуючий кінцевий автомат із повною відмовою від блокуючих функцій затримки.
- Завдяки використанню спеціалізованої бібліотеки *Preferences.h*, система автоматично резервує цільові біологічні уставки у внутрішній Flash-пам'яті мікроконтролера.
- Усі розраховані прогнози та команди керування публікуються брокером із параметром *Retain=True*.
- Алгоритм здійснює постійний моніторинг актуальності даних від моделі ARIMA, автоматично анулюючи прогноз у разі відсутності оновлень понад 60 хвилин.



Фрагмент алгоритму синхронізації та перевірки вірності даних

Граф станів алгоритму автономного виживання

АПАРАТНА РЕАЛІЗАЦІЯ ГРАНИЧНОГО ВУЗЛА

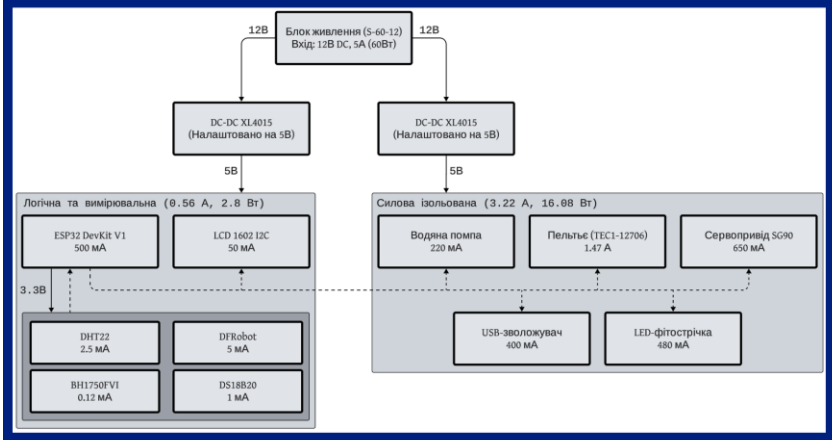
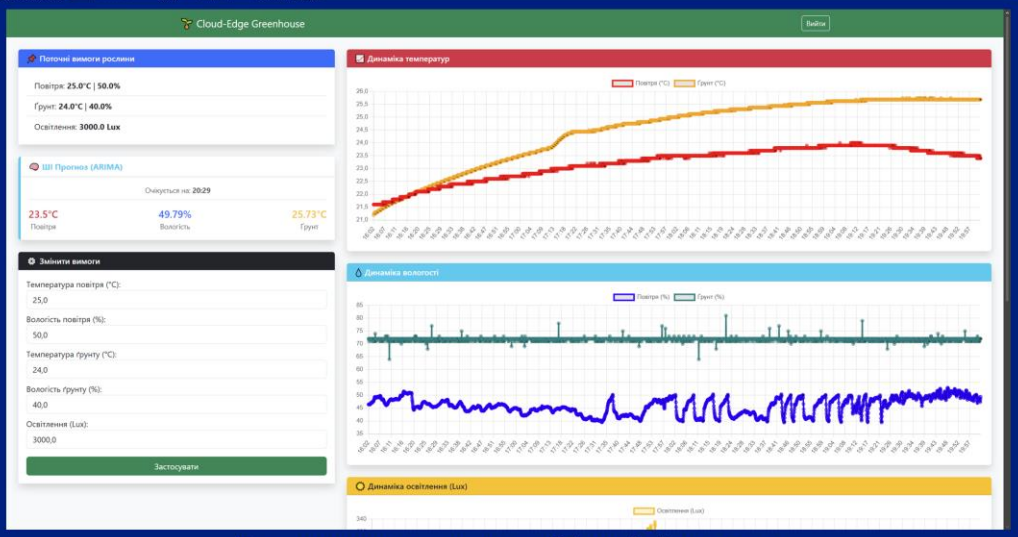


Схема розподілу ліній живлення

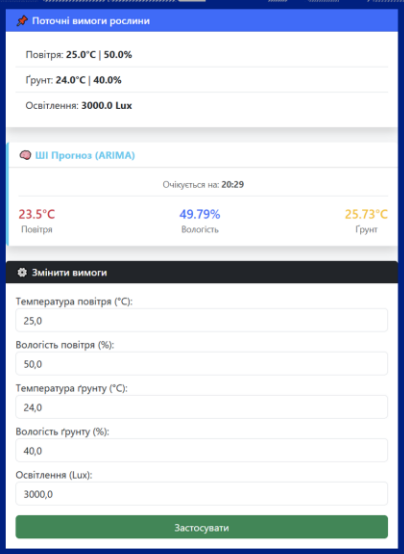
- Граничний контролер: ESP32 DevKit V1 (двоядерна архітектура, Wi-Fi).
- Розділення логічної та силової ліній для захисту від перешкод.
- Безперервне зчитування показників мікроклімату (повітря, ґрунт, освітлення).
- Формування керуючих ШІМ- та цифрових сигналів для виконавчих механізмів.

ПРОГРАМНА РЕАЛІЗАЦІЯ ХМАРНОГО ІНТЕРФЕЙСУ



Головна панель керування веб-інтерфейсу системи

ПРОГРАМНА РЕАЛІЗАЦІЯ ХМАРНОГО ІНТЕРФЕЙСУ

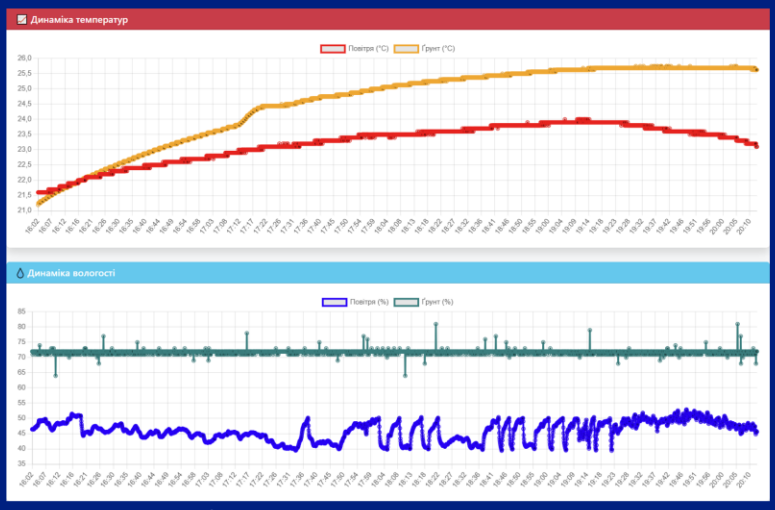


Панель контролю вимог та прогнозу

- Інтерактивні панелі дозволяють оператору встановлювати цільові біологічні норми, які миттєво зберігаються у базі даних MySQL та передаються на граничний вузол. Використання механізму MQTT Retain гарантує, що мікроконтролер отримає останні актуальні вимоги одразу після відновлення з'єднання, забезпечуючи безперервність технологічного процесу.
- Панель прогнозів відображає результати обчислень моделі ARIMA, надаючи оператору візуальний контроль за передбаченим станом мікроклімату з горизонтом у 30 хвилин. Це дозволяє оператору завчасно оцінювати температурні тенденції та підтверджує ефективність роботи алгоритмів прогнозуючої аналітики хмарного ядра.

ПРОГРАМНА РЕАЛІЗАЦІЯ ХМАРНОГО ІНТЕРФЕЙСУ

- Одночасний моніторинг показників повітря та ґрунтового середовища дозволяє комплексно контролювати стан мікроклімату.
- Суміщення графіків наочно демонструє різницю у швидкості нагрівання й охолодження ґрунту відносно повітря.
- Оновлення графіків в реальному часі. Асинхронне підвантаження даних кожні 5 секунд забезпечує миттєве оновлення графіків без перезавантаження сторінки.



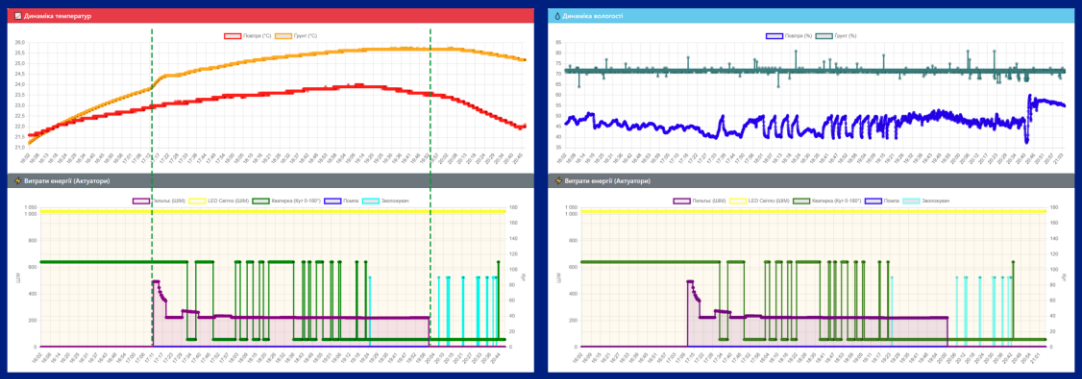
Графіки динаміки температур та вологості

ПРОГРАМНА РЕАЛІЗАЦІЯ ХМАРНОГО ІНТЕРФЕЙСУ

- Моніторинг природного освітлення для оцінки компенсації дефіциту світла.
- Використання двох осей ординат дозволяє одночасно аналізувати неперервні ШІМ-сигнали (Пельтьє, LED) та дискретні стани реле (помпа, кватирка).

Графіки динаміки освітлення та витрат енергії

ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА



- Використання алгоритмів нечіткого виведення Мамдані дозволяє системі уникати автоколивань, характерних для класичних релейних регуляторів. Завдяки плавній зміні ШІМ-сигналу забезпечується асимптотичне наближення до цільової температури, що повністю виключає термічний стрес для агрокультур.
- Відмова від бінарної логіки (Вкл/Викл) на користь прогнозуючої ШІМ мінімізує пікові навантаження на систему живлення. Це дозволяє підтримувати стабільний рівноважний стан актуаторів, знижуючи загальне споживання електроенергії за рахунок усунення циклічних перехідних процесів.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень:

- проаналізовано відомі методи автоматизованого керування мікрокліматом та сучасні архітектури IoT-систем;
- розроблено метод прогнозуючого інтелектуального керування мікрокліматом на базі гібридної Cloud-Edge архітектури;
- здійснено дослідження ефективності прогнозуючої моделі ARIMA, локального нечіткого регулятора та алгоритмів автономного виживання;
- реалізовано запропонований метод у вигляді повноцінного програмно-апаратного комплексу з хмарним HMI-дашбордом.

Запропонований метод забезпечує плавну підтримку мікроклімату без ефекту перерегулювання завдяки упереджувальному реагуванню на теплову інерцію теплиці. Підтверджено перевагу прогнозуючого нечіткого керування над класичними релейними підходами. Досягнуто зниження пікових енергозатрат апаратної частини та забезпечено 100% відмовостійкість системи при обривах інтернет-з'єднання, що доводить високу ефективність розробленого методу.

ПУБЛІКАЦІЇ

- Лісовий В. Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи. ПерСиК 2026, Харків, Україна, 23 квіт. 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Вадим ЛІСОВИЙ

Тема: Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень – ; кількість сторінок записки 147

1. Короткий зміст роботи та прийнятих рішень У роботі розроблено гібридну Cloud-Edge систему автоматизованого керування мікрокліматом теплиці. Прийнято рішення щодо використання прогнозуючої моделі ARIMA на хмарному рівні для аналізу тенденцій та двопортового нечіткого регулятора Мамдані на рівні граничного вузла. Це дозволило реалізувати стратегію упереджувального керування інерційними процесами та забезпечити високу енергоефективність системи.

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проаналізовано методи керування мікрокліматом теплиць, застосування штучного інтелекту в агрономії та архітектури IoT-систем. У другому розділі розроблено математичні моделі динаміки мікроклімату, локального нечіткого регулятора та структурно-енергетичну модель апаратної платформи. У третьому розділі розроблено метод прогнозуючого керування, алгоритми забезпечення відмовостійкості та оптимізовано базу нечітких правил. У четвертому розділі реалізовано програмно-апаратну систему, розроблено веб-інтерфейс та наведено результати експериментів, що підтверджують ефективність рішень.

4. Позитивні сторони роботи: Вагомою перевагою є реалізація механізмів відмовостійкості, що гарантують автономну роботу системи при втраті зв'язку. Також позитивно відзначається створення адаптивного веб-інтерфейсу з асинхронним оновленням даних.

5. Негативні сторони роботи: Доцільно було б навести порівняльний аналіз ефективності моделі ARIMA з іншими методами прогнозування та детальніше розглянути перспективи масштабування системи для великих агрокомплексів.

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на належному науково-технічному рівні.

8. Інші зауваження: –

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «добре» 85.00 (В)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Едуард Мануров доц. каф КН

"1 травня" _____ 2026р. _____

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Вадим ЛІСОВИЙ

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-24-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи
Автор Вадим ЛІСОВИЙ
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д.т.н., професор Василь ЯЦКІВ

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 3,2% і адресується до 33 першоджерела; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

12.05.2026

Завідувач кафедри _____ Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми _____ Підпис — Олег САВЕНКО (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ Підпис — Василь ЯЦКІВ (Ім'я, ПРІЗВИЩЕ)

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Вадим ЛІСОВИЙ

Співавтор:

Назва: Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи

Експерт: Василь ЯЦКІВ

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3.2%

Коефіцієнт подібності 2:0.93%

Мікропробіли: 54

Заміна букв: 30

Інтервали: 0

Білі знаки: 6

Дата створення звіту: 2026-05-12 18:14:01.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-05-12 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 271393 Назва: МКР Система керування мікрокліматом теплиці з прогнозуванням та оптимізацією режимів роботи Додано в БД: 2026-05-12 Автора: Вадим ЛІСОВИЙ Керівники: Василь ЯЦКІВ Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	136572	1003	1588 (1%)	22 (2%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми