НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету	**Завідувач кафедри**
інформаційних технологій	**комп'ютерних наук**
_____ __Ігор Болбот____	_____ _Белла Голуб__
(підпис) (Ім'я ПРІЗВИЩЕ)	(підпис) (Ім'я ПРІЗВИЩЕ)
"___" _____ 20__ р.	"___" _____ 20__ р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Система автоматизації опалення приміщення з прогнозуванням температурних умов на основі технологій інтернету речей

Спеціальність _____122 «Комп'ютерні науки»_____
(код і найменування)

Освітня програма _____Інформаційні управляючі системи та технології__
(назва)

Орієнтація освітньої програми_____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

_____доцент. к.т.н._____ _____ _____Белла Голуб____
(науковий ступінь та вчене звання) (підпис) (Ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи

_____доцент. к.т.н._____ _____ _____Віталій Сватко_____
(науковий ступінь та вчене звання) (підпис) (Ім'я ПРІЗВИЩЕ)

Виконав _____ _____Богдан Галайда_____
(підпис) (Ім'я ПРІЗВИЩЕ здобувача)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук
доцент, к.т.н. Белла Голуб
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"01" листопада 2024 року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Галайді Богдану Миколайовичу
(прізвище, ім'я, по батькові)

Спеціальність 122 «Комп'ютерні науки»
(код і назва)

Освітня програма Інформаційні управляючі системи та технології
(назва)

Орієнтація освітньої програми освітньо-професійна

Тема магістерської кваліфікаційної роботи Система автоматизації опалення приміщення з прогнозуванням температурних умов на основі технологій інтернету речей.

затверджена наказом ректора НУБіП України від "01" листопада 2024р. №1964 «С»

Термін подання завершеної роботи на кафедру _____
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи: набори даних від IoT-датчиків, історичні кліматичні дані, технічні дані системи опалення та відкриті набори даних.

Перелік питань, що підлягають дослідженню:

1. Аналіз ключових проблем автоматизації опалювальних систем, які можна вирішити за допомогою інтелектуального аналізу даних (зайві витрати енергії, відсутність адаптації до погодних умов та присутності людей)
2. Використання методів OLAP та Data Mining для аналізу історичних даних з метою виявлення закономірностей у зміні температури та споживанні енергії.
3. Аналіз впливу зовнішніх та поведінкових факторів (температури повітря, вологості, присутності чи відсутності людей, часу доби) на ефективність роботи системи.
4. Розробка моделі прийняття рішень для автоматизованого управління системою, що буде мінімізувати витрати енергії та забезпечить комфортні умови перебування.

Перелік графічного матеріалу (за потреби)

Дата видачі завдання "01" листопада 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Віталій Сватко
(підпис) (ім'я ПРІЗВИЩЕ)

Завдання прийняв до виконання _____ Богдан Галайда
(підпис) (ім'я ПРІЗВИЩЕ)

Календарний план

№ з/п	Назва етапів виконання магістерської кваліфікаційної роботи	Строк виконання етапів магістерської кваліфікаційної роботи	Примітка
1	Видача завдання	01.11.2024	
2	Аналіз предметної області	01.11.2024-10.12.2025	
3	Моделювання предметної області	11.12.2024-23.01.2025	
4	Розробка системи	24.01.2024-18.05.2025	
5	Аналіз результатів	19.05.2025-31.08.2025	
6	Оформлення записки	01.09.2025-01.11.2025	
7	Постерна сесія	28.10.2025-29.10.2025	
8	Перевірка на плагіат	13.11.2025	
9	Попередній захист	01.12.2025	
10	Захист	16.12.2025	

Студент _____ Богдан Галайда
(підпис) (ім'я ПРІЗВИЩЕ)

Керівник магістерської кваліфікаційної роботи _____ Віталій Сватко
(підпис) (ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Магістерська кваліфікаційна робота присвячена розробленню інтелектуальної системи автоматизації опалення приміщення з прогнозуванням температурних умов на основі технологій Інтернету речей та методів аналітичної обробки даних.

Об'єкт дослідження – процес автоматизованого керування тепловими режимами приміщень із використанням сенсорної мережі та прогнозних моделей.

Предмет дослідження – методи моделювання, обробки, зберігання, аналізу й прогнозування даних мікроклімату, а також алгоритми адаптивного управління системами опалення.

Використані методи. У роботі застосовано системний аналіз, UML-моделювання, об'єктно-орієнтоване проєктування, багатовимірний аналіз (OLAP), методи Data Mining, моделі машинного навчання для прогнозування температури, а також алгоритми модельно-прогнозного керування (MPC). Реалізація програмної системи виконана із застосуванням Python, PyQt6, SQLite та протоколу MQTT для збору телеметрії.

Мета роботи – створення аналітичної системи автоматизованого керування опаленням, що поєднує IoT-сенсори, моделі прогнозування та інтелектуальні методи оптимізації для зниження енергоспоживання і підвищення стабільності температурного режиму.

Наукова складова. У роботі запропоновано інтелектуальну архітектуру керування опаленням, яка об'єднує сенсорну інфраструктуру, модуль прогнозування на основі машинного навчання та модельно-прогнозне керування. Інтеграція OLAP-аналізу, IoT-телеметрії та алгоритмів ML забезпечує адаптивну реакцію системи на зміну зовнішніх та поведінкових факторів, що є розвитком сучасних підходів до кіберфізичних систем мікроклімату.

Практична значущість. Розроблена система забезпечує скорочення енергоспоживання, підвищує точність підтримання температури та може

бути впроваджена у побутових, промислових або адміністративних будівлях. Програмне забезпечення включає модулі збору телеметрії, прогнозування температури, візуалізації даних та оптимізації керуючих впливів.

Рекомендації щодо впровадження. Система може використовуватися як складова «розумного дому», бути інтегрована з існуючими IoT-платформами або розширена для роботи з багатозональними системами опалення. Також можливе застосування у дослідницьких задачах енергоефективності та розробленні кіберфізичних систем управління мікрокліматом.

Обсяг роботи:

кількість сторінок – 97

кількість рисунків – 41

кількість таблиць – 14

кількість додатків – 1

кількість джерел – 30.

ABSTRACT

The master's thesis is devoted to the development of an intelligent heating automation system with temperature forecasting capabilities based on Internet of Things technologies and analytical data-processing methods.

Object of research – the process of automated control of indoor thermal conditions using a sensor network and predictive models.

Subject of research – methods of modelling, processing, storing, analysing and forecasting microclimate data, as well as algorithms for adaptive control of heating systems.

Methods used. The study employs system analysis, UML modelling, object-oriented design, multidimensional OLAP analysis, Data Mining techniques, machine-learning models for temperature prediction, and Model Predictive Control (MPC) algorithms. The software implementation is based on Python, PyQt6, SQLite, and the MQTT protocol for telemetry acquisition.

Purpose of the work – to develop an analytical system for automated heating control that integrates IoT sensors, forecasting models, and intelligent optimisation methods to reduce energy consumption and improve temperature stability.

Scientific contribution. The thesis proposes an intelligent control architecture that unifies a sensor infrastructure, a machine-learning-based forecasting module, and model-predictive temperature regulation. The integration of OLAP analysis, IoT telemetry, and ML algorithms provides adaptive system behaviour in response to external and behavioural factors, advancing contemporary approaches to cyber-physical microclimate control systems.

Practical significance. The developed system reduces energy consumption, increases temperature-control accuracy, and can be deployed in residential, industrial, or administrative buildings. The software includes modules for telemetry collection, temperature forecasting, data visualisation, and optimisation of control actions.

Implementation recommendations. The system may be used as a component of a smart home solution, integrated with existing IoT platforms, or extended for multi-zone heating systems. It can also be applied in research related to energy efficiency and the development of cyber-physical microclimate control technologies.

Thesis parameters:

number of pages – 97

number of figures – 41

number of tables – 14

number of appendices – 1

number of references – 30

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. API — програмний інтерфейс взаємодії застосунків
2. REST — архітектурний стиль клієнт–серверної взаємодії
3. HTTPS — протокол захищеної передачі даних
4. TLS — протокол криптографічного захисту транспортного рівня
5. LAN — локальна мережа
6. MQTT — брокерський протокол обміну повідомленнями у IoT
7. QoS — рівень гарантій доставки MQTT-повідомлень
8. JSON — формат структурування даних
9. GPIO — універсальні порти вводу-виводу контролера
10. I2C — шина двонаправленого цифрового зв’язку
11. 1-Wire — однопровідна шина обміну з сенсорами
12. ESP32 — мікроконтролер IoT з Wi-Fi
13. DHT22 — сенсор температури та вологості
14. BMP280 — сенсор тиску і температури
15. Firmware — прошивка контролера
16. ML — машинне навчання
17. MPC — модельно-прогнозне керування
18. Dataset — набір даних
19. Feature — вхідна ознака моделі
20. Target — вихідна змінна для прогнозування
21. Cluster — група даних кластеризації
22. Reaction class — класифікація швидкості реакції (“Fast”, “Slow”)
23. Prediction horizon (H) — горизонт прогнозування
24. T — температура всередині приміщення (°C)
25. T_out — температура зовнішнього повітря (°C)
26. RH — відносна вологість (%)
27. u — керуючий вплив (потужність нагріву)

28. U_min / U_max — мінімальне та максимальне керування
29. T_setpoint — цільова температура (°C)
30. Δt — інтервал дискретизації вимірювань
31. J(T, u) — оптимізаційний критерій MPC
32. w_energy — коефіцієнт енергоспоживання
33. w_tracking — коефіцієнт точності стеження
34. DB — база даних
35. Config DB — база конфігурацій зон
36. TimeSeries DB — база часових рядів телеметрії
37. Forecasts — таблиця прогнозних значень
38. Heating Logs — журнал реалізованих планів керування
39. Report Storage — сховище HTML-звітів
40. API Gateway — центральний шлюз запитів
41. ForecastService — модуль прогнозування температури
42. ControlService — модуль MPC-керування
43. AuthService — модуль автентифікації
44. DesktopApp — клієнтський застосунок PyQt6
45. MQTT Broker — брокер повідомлень (Mosquitto)
46. OLAP — багатовимірна аналітична обробка даних
47. KPI — ключовий показник ефективності
48. KPI Efficiency — індикатор ефективності опалення
49. Execution time — час виконання MPC
50. Cluster ID — ідентифікатор кластера поведінки

ВСТУП

Система автоматизації опалення з прогнозуванням температурних умов є актуальним напрямом розвитку розумних технологій, що поєднує енергозбереження, комфорт користувачів і гнучке керування кліматом у приміщеннях. Зростання цін на енергоресурси та потреба у зменшенні тепловтрат стимулюють впровадження інтернету речей (IoT), який забезпечує збір і обробку даних у режимі реального часу. Використання сенсорів, мікроконтролерів і хмарних сервісів дозволяє створити адаптивну систему, здатну передбачати зміни температури й автоматично коригувати роботу опалювального обладнання для досягнення оптимального балансу між споживанням енергії та комфортом [1].

Актуальність теми теми зумовлена зростанням попиту на системи інтелектуального керування мікрокліматом, які здатні зменшити енергоспоживання та підвищити комфорт користувачів.

Метою кваліфікаційної роботи є розроблення інтелектуальної системи автоматизації опалення приміщення з функцією прогнозування температурних умов на основі технологій інтернету речей (IoT) та методів машинного навчання, що забезпечує підвищення енергоефективності та комфортності мікроклімату.

Для виконання поставленої мети необхнідно виконати наступні **завдання:**

1. Провести системний аналіз предметної області та визначити ключові проблеми існуючих систем автоматизації опалення;
2. Сформувати специфікацію вимог до розроблюваної інформаційної системи;
3. Провести моделювання предметної області за допомогою UML і ER-діаграм;
4. Виконати огляд сучасних IoT-рішень і методів інтелектуального керування опаленням, включаючи алгоритми прогнозного управління;

5. Розробити логічну модель бази даних і основні діаграми проєктування (класи, компоненти, пакети);

6. Застосувати методи OLAP і Data Mining для аналізу історичних даних сенсорів, виявлення закономірностей у зміні температури та споживанні енергії.

7. Виконати вибір інструментальних засобів і здійснити реалізацію програмного та інформаційного забезпечення системи, включно з модулем прогнозування і керування;

8. Провести тестування, оцінювання ефективності роботи системи та розробити рекомендації щодо її впровадження й подальшої експлуатації.

Об'єктом дослідження є процес автоматизованого керування тепловими режимами приміщень із використанням технологій інтернету речей та прогнозного аналізу даних.

Предметом дослідження виступають методи моделювання, обробки, збереження й візуалізації даних сенсорної телеметрії**,** а також алгоритми прогнозування температурних умов для оптимізації енергоспоживання.

Для досягнення поставленої мети застосовано методи системного аналізу, UML-моделювання, об'єктно-орієнтованого проєктування, побудови баз даних і програмної реалізації з використанням Python**.** Практична частина базується на технологіях PyQt6 для створення графічного інтерфейсу користувача, SQLite3 для збереження даних, а також NumPy**,** Pandas і Scikit-learn для аналітики, прогнозування та оптимізації процесів опалення.

Наукова новизна полягає у розробленні уніфікованої архітектури системи, що поєднує сенсорну мережу, аналітичний модуль та адаптивну модель прогнозування температури на основі методів машинного навчання. Вона дозволяє не лише відстежувати стан мікроклімату, а й динамічно коригувати параметри роботи системи відповідно до прогнозу погодних умов і поведінкових факторів користувачів.

Практична цінність полягає у створенні інтегрованого рішення, яке може бути впроваджене як у побутових, так і в промислових або адміністративних

будівлях. Запропонована система забезпечує економію енергії, підвищує комфортність перебування людей у приміщенні та сприяє реалізації концепції «розумного дому». Вона також може бути використана як основа для подальших досліджень у сфері енергоефективних технологій, управління мікрокліматом і побудови предиктивних кіберфізичних систем.

Апробація

1. Б.М.Галайда, В. В. Сватко: СИСТЕМА АВТОМАТИЗАЦІЇ ОПАЛЕННЯ З ПРОГНОЗУВАННЯМ ТЕМПЕРАТУРНИХ УМОВ НА ОСНОВІ ТЕХНОЛОГІЙ ІНТЕРНЕТУ РЕЧЕЙ Збірник матеріалів II Міжнародно науково–практичної конференції «АКТУАЛЬНІ ПИТАННЯ РОЗВИТКУ НАУКИ ТА ТЕХНІКИ В УМОВАХ ГЛОБАЛІЗАЦІЇ», 14.05.2025. НУБіП України, ВСП «Боярський фаховий коледж НУБіП України», Боярка – С. 128–130.

Структура кваліфікаційної роботи включає чотири розділи. У першому розділі здійснено аналіз предметної області, визначено проблеми існуючих рішень і вимоги до майбутньої системи. Другий розділ присвячений моделюванню структури даних і побудові UML-діаграм. У третьому розділі подано архітектуру, вибір інструментальних засобів і реалізацію програмного забезпечення. Четвертий розділ містить результати тестування, рекомендації щодо впровадження та оцінку ефективності розробленої системи.

1 СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Предметна область системи автоматизації опалення з прогнозуванням температурних умов охоплює взаємодію сенсорної інфраструктури, інтелектуальних контролерів, хмарних аналітичних сервісів та користувацьких інтерфейсів, що утворюють єдину кіберфізичну екосистему. Центральним елементом є IoT-контролер, який об'єднує потоки даних від сенсорів температури, вологості та тиску, здійснює попередню обробку сигналів і забезпечує комунікацію з аналітичним модулем через протоколи MQTT або HTTP. Хмарна платформа виконує агрегацію та зберігання даних у базі, реалізує механізми прогнозування температурних трендів за допомогою моделей машинного навчання та формує рекомендації для регулювання теплового режиму. На рівні користувача функціонує веб-інтерфейс або мобільний застосунок, що забезпечує моніторинг у реальному часі, перегляд історії вимірювань і дистанційне керування опаленням (рис. 1.1). Така архітектура реалізує принцип розподіленої обробки даних, знижує затримки реакції системи та підвищує стабільність управління в динамічних умовах навколишнього середовища [1].

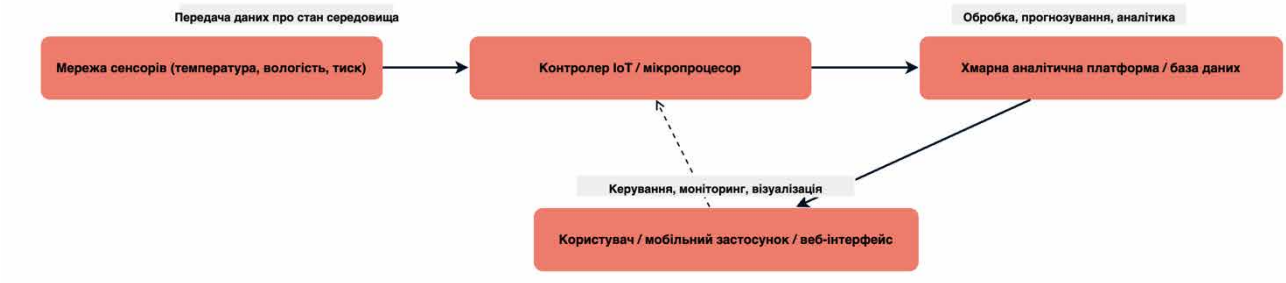


Рис. 1.1 – Структура предметної області системи автоматизації опалення на основі технологій інтернету речей

Інформаційна взаємодія між компонентами системи представлена у таблиці 1.1, де наведено джерела, типи, частоту передавання й призначення даних. Сенсорна мережа передає вимірювані параметри мікроклімату на

контролер у режимі реального часу, який, у свою чергу, здійснює попередню обробку й надсилає агреговані дані в аналітичне сховище. Хмарна платформа виконує прогнозування майбутніх температурних значень і повертає результати користувацькому застосунку для візуалізації та корекції режимів роботи системи. Такий механізм забезпечує цілісність даних, адаптивність алгоритмів і можливість масштабування системи відповідно до вимог користувача або технічної інфраструктури [2].

Таблиця 1.1 – Основні інформаційні потоки між компонентами системи автоматизації опалення

Джерело даних	Отримувач	Тип даних	Частота передавання	Призначення
Сенсорна мережа	Контролер IoT	Температура, вологість, тиск	1 раз/5 с	Моніторинг мікроклімату
Контролер IoT	Хмарна платформа	Агреговані дані	1 раз/30 с	Аналітика й прогнозування
Хмарна платформа	Користувач	Прогноз, рекомендації	1 раз/1 хв	Візуалізація, контроль
Користувач	Контролер IoT	Команди керування	За подією	Регулювання опалення

Завдяки такій структурі предметна область системи забезпечує узгоджене функціонування апаратних і програмних компонентів, що дозволяє реалізувати замкнений цикл “збір – аналіз – реакція”, мінімізувати втрати енергії та підвищити точність підтримання комфортних температурних параметрів у приміщеннях. Це створює основу для подальшого вдосконалення аналітичних модулів і впровадження предиктивного управління в системах розумного дому.

1.2 Теоретико-методологічні засади та стан наукових досліджень

Теоретико-методологічні основи побудови інтелектуальних систем керування мікрокліматом ґрунтуються на поєднанні кіберфізичних компонентів, математичних моделей теплових процесів та алгоритмів автоматичного

регулювання, інтегрованих у розподілену інфраструктуру Інтернету речей. Сучасні дослідження підкреслюють, що базовим рівнем такої системи є апаратна архітектура, яка забезпечує автономне живлення, оброблення первинних сигналів, бездротовий обмін даними та взаємодію з сенсорно-виконавчими модулями. У роботі Su (2024) наведено типовий апаратний комплекс, побудований на мікроконтролері STM32, де продемонстровано структурну взаємодію модулів живлення, схеми скидання, модулів зберігання та комунікацій із центральним процесорним ядром (рис. 1.2). Така конфігурація відображає класичний підхід до формування сенсорного шару IoT-систем, що забезпечує надійність, захищеність та стійкість до зовнішніх перешкод.

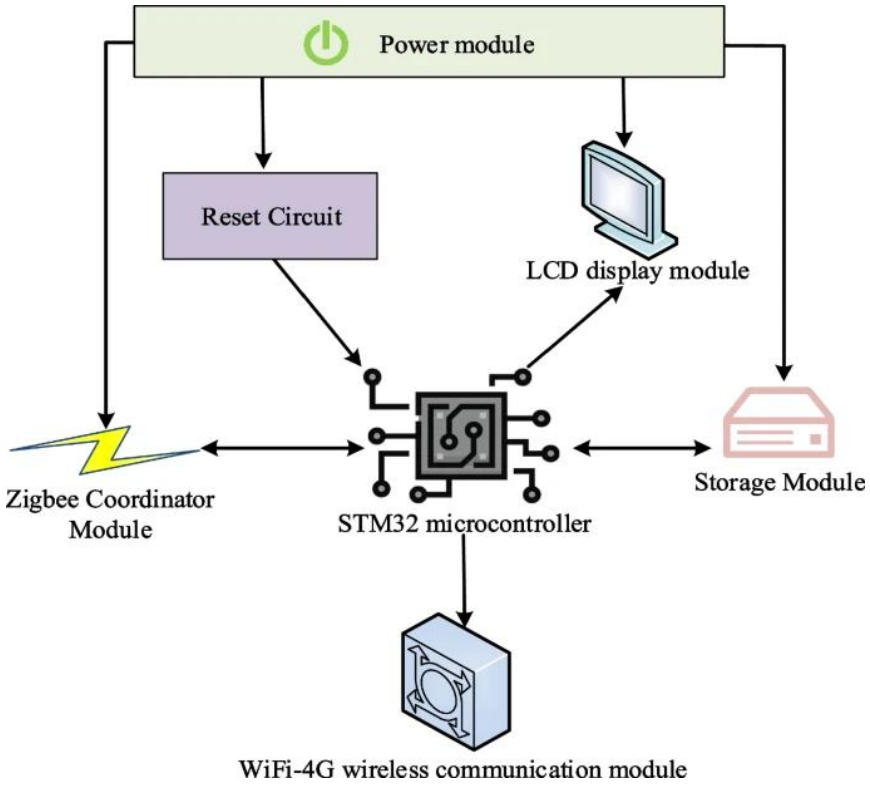


Рис. 1.2 – Структурна апаратна модель STM32-комплексу для системи інтелектуального керування [Su, 2024].

Другим ключовим елементом є комунікаційна інфраструктура, яка поєднує фізичні сенсори, координатори ZigBee, домашні шлюзи та хмарні сервери. У дослідженні Michailidis et al. (2025) підкреслено, що саме зв’язок «сенсорна мережа – локальний шлюз – хмарний сервіс» формує основу для реалізації

віддаленого керування та аналітики, забезпечуючи масштабовність та інтеграцію з мобільними клієнтами (рис. 1.3).

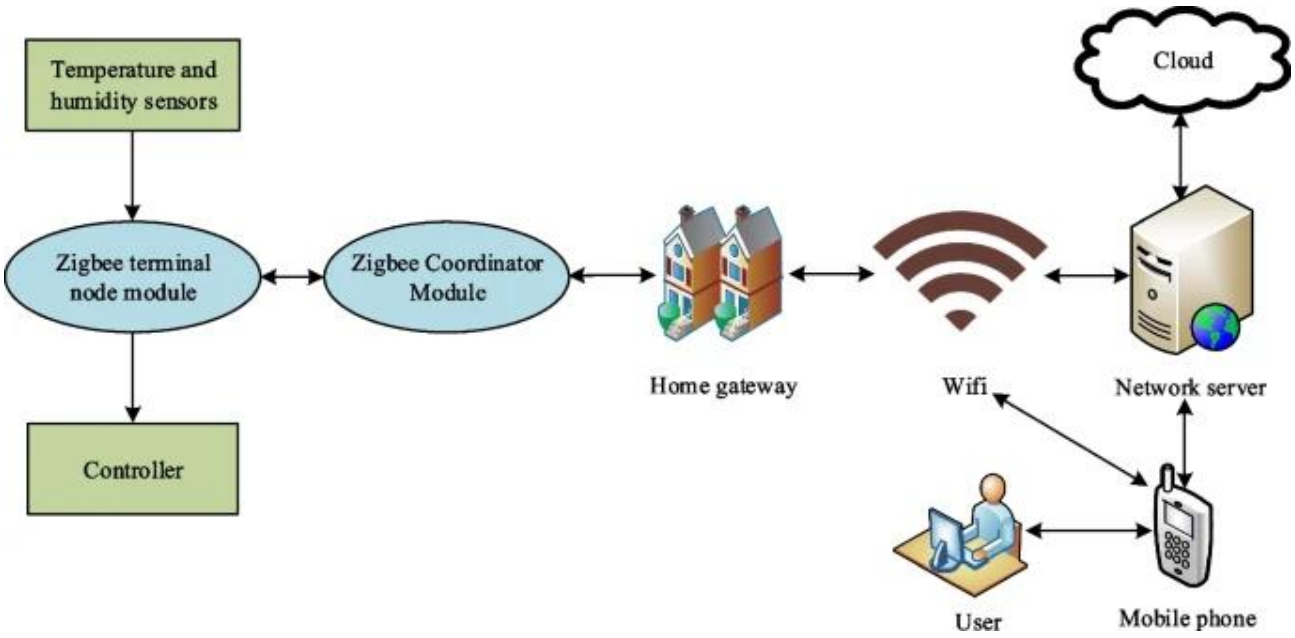


Рис. 1.3 – Комунікаційна схема ZigBee – Gateway – Cloud для інтелектуальних систем мікроклімату [Michailidis et al., 2025]

Подібні архітектурні рішення особливо важливі для систем опалення, де необхідне швидке реагування на зміну зовнішніх умов, а також підтримка асинхронної телеметрії у режимі реального часу. У зазначених роботах наголошується, що ZigBee-орієнтовані мережі демонструють високу енергоефективність і надійність передачі даних, що робить їх оптимальними для розподілених систем побутового теплового контролю.

Алгоритмічне ядро більшості систем керування мікрокліматом базується на математичних моделях регулювання, серед яких найбільш поширеним є PID-контролер, що оперує пропорційною, інтегральною та диференційною складовими впливу. Згідно з оглядом Kathirgamanathan et al. (2021), PID-структури, попри свою простоту, широко застосовуються у низці інженерних систем завдяки стабільності, відтворюваності та зрозумілості моделі (рис. 1.4). Проте автори підкреслюють, що класичні PID-алгоритми є недостатньо адаптивними у середовищах із суттєвими коливаннями зовнішньої температури, обмеженнями виконавчих механізмів та нелінійностями теплових характеристик

будівель, що актуалізує потребу у прогнозному регулюванні або методах машинного навчання.

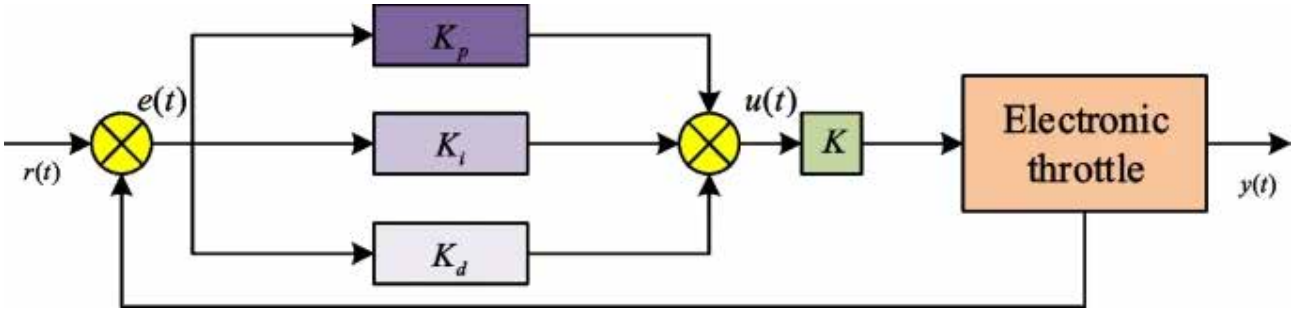


Рис. 1.4 – Структурна модель PID-регулятора для систем температурного контролю [Kathirgamanathan et al., 2021]

На рівні експериментальних досліджень Zhuang et al. (2023) демонструють, що поведінка внутрішньої температури істотно залежить не лише від теплових властивостей приміщення, але й від швидкодії зовнішніх факторів, таких як динамічні зміни погоди, сезонні впливи та теплові навантаження на огороджувальні конструкції. Це підтверджується експериментальними кривими зміни температури (рис. 1.5)

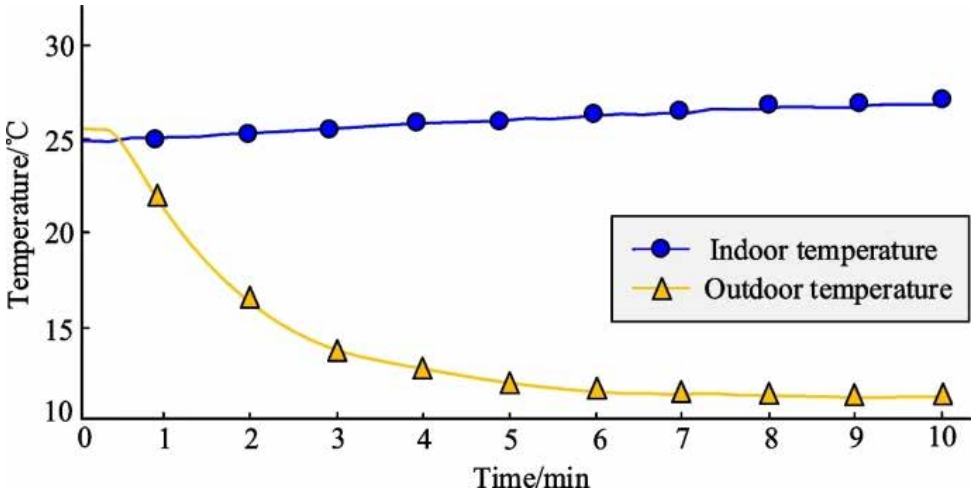


Рис. 1.5 – Динаміка зміни внутрішньої та зовнішньої температури у експериментальному середовищі [Zhuang et al., 2023].

Та вологості (рис. 1.6), що відображають нерівномірний характер теплових процесів, стохастичні коливання та проблему теплової інерційності.

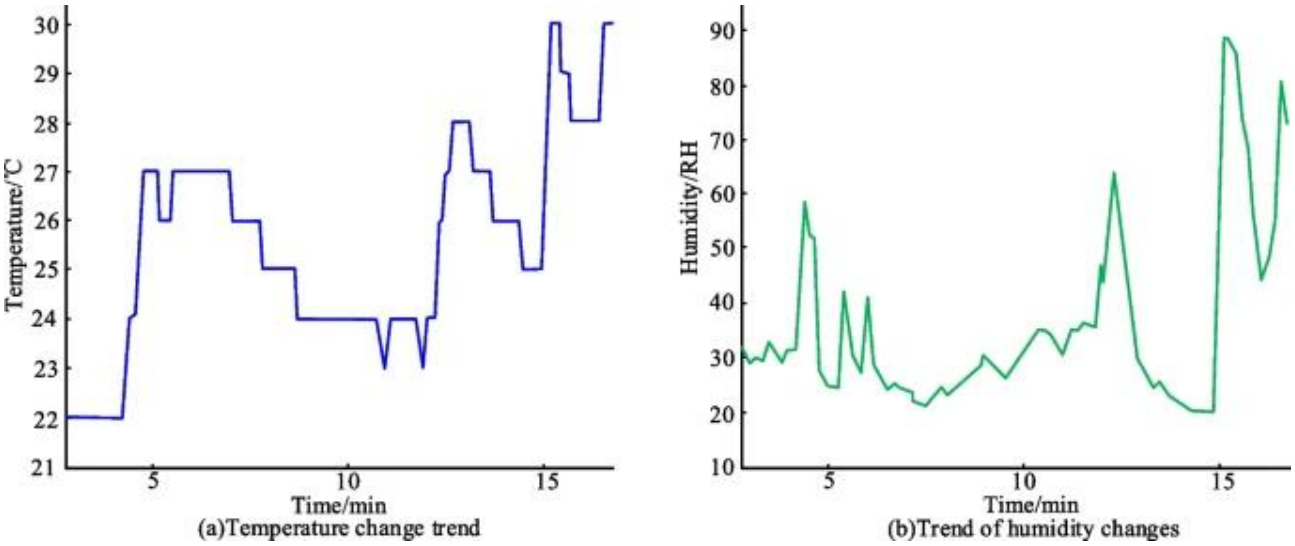


Рис. 1.6 – Тренди зміни температури та вологості у часовому вимірі [Zhuang et al., 2023].

Наведені результати дозволяють зробити висновок про те, що сучасні системи контролю опалення потребують інтеграції модулів прогнозування, які здатні моделювати майбутні стани системи та коригувати роботу виконавчих елементів з урахуванням прогнозованих змін, а не лише поточних вимірів.

Узагальнюючи результати проаналізованих наукових джерел, можна зазначити, що сучасні наукові підходи тяжіють до комплексних систем інтелектуального керування, які поєднують IoT-інфраструктуру, моделі машинного навчання, предиктивні алгоритми управління та адаптивні комунікаційні технології. На цьому тлі наукова новизна даної роботи полягає у розробленні цілісної інтелектуальної системи опалення з функцією прогнозування температурних характеристик на основі машинного навчання, інтегрованого з модифікованим алгоритмом керування, що забезпечує підвищену стабільність, адаптивність та енергоефективність у порівнянні з класичними системами PID-регулювання та традиційними IoT-рішеннями.

1.3 Аналіз сучасних IoT-технологій і систем

Розвиток технологій інтернету речей спричинив появу розгалуженого ринку програмно-апаратних систем для керування кліматом у приміщеннях, які поєднують сенсорні мережі, аналітику та адаптивне прогнозування температури. Одним із провідних рішень є Google Nest Learning Thermostat, що застосовує машинне навчання для автоматичного формування графіків опалення, враховуючи поведінку користувача, температуру зовнішнього середовища та рівень вологості. Мобільний застосунок забезпечує віддалений доступ і рекомендації щодо оптимального енергоспоживання (див. рис. 1.7).

Рис. 1.7 – Інтерфейс системи Google Nest для прогнозування та регулювання температури

Іншим ефективним прикладом є Tado Smart Thermostat, який використовує геолокаційні механізми для автоматичного зниження температури у випадку відсутності користувача. Система підтримує багатозональне керування, інтеграцію з хмарною аналітикою та API для зв’язку з іншими платформами інтернету речей (наведено на рис. 1.8).

Рис. 1.8 – Апаратно-програмний комплекс Tado Smart Thermostat із геолокаційним контролем

До числа інноваційних рішень також належить Ecobee Smart Thermostat, що поєднує аналітику даних і голосове керування. Завдяки системі бездротових сенсорів Ecobee здійснює балансування температури у різних приміщеннях і прогнозує теплове навантаження з урахуванням погодних умов (рис. 1.9).

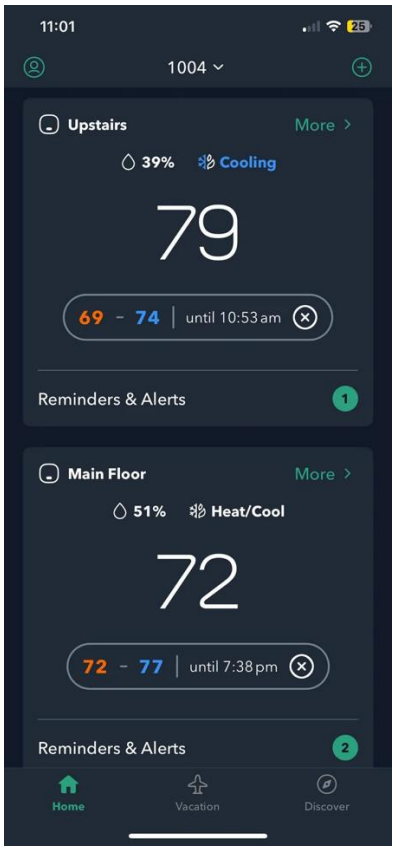


Рис. 1.9 – Програмний інтерфейс системи Ecobee із підтримкою голосового керування

Система Honeywell Home Evohome реалізує централізоване управління опаленням у межах будинку. Архітектура складається з головного контролера з сенсорним дисплеєм, бездротових терморегуляторів і програмної платформи для створення сценаріїв опалення. Завдяки зручному інтерфейсу користувач може налаштовувати індивідуальні режими температури для кожної зони (зображено на рис. 1.10).

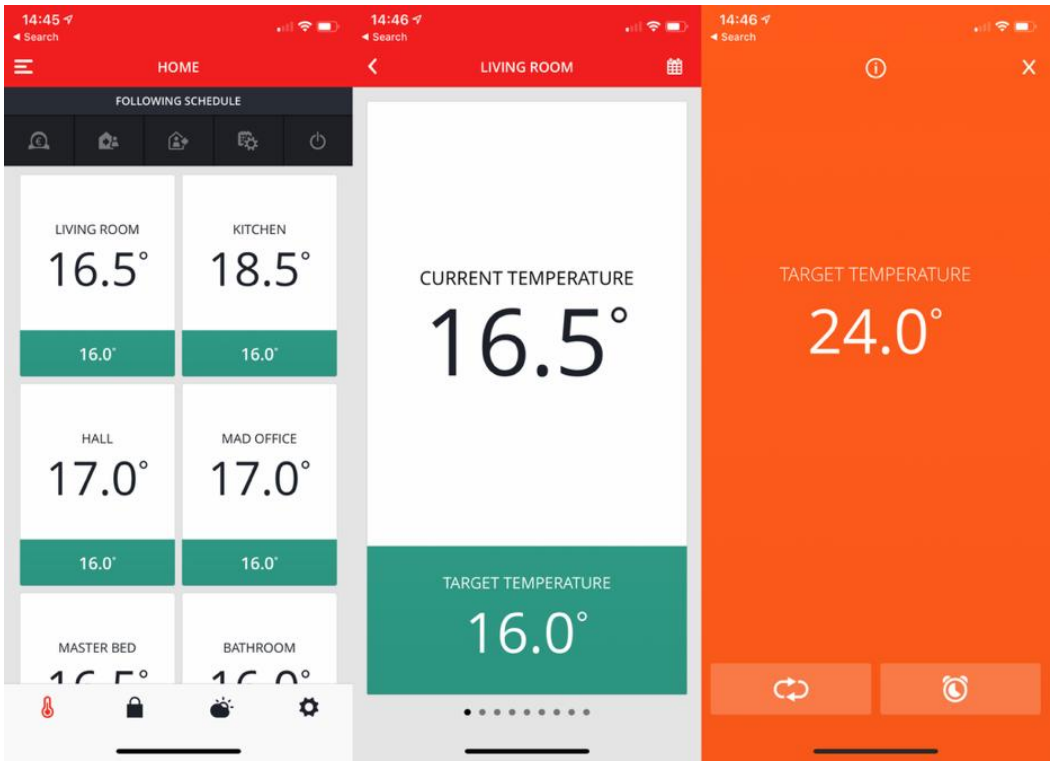


Рис. 1.10 – Інтерфейс і апаратні модулі системи Honeywell Home Evohome

Окремо слід зазначити Heatmiser Neo System, що поєднує програмний і апаратний рівні керування для побудови масштабованої системи “розумного дому”. Її відмінністю є гнучка логіка зонального опалення, можливість інтеграції з голосовими асистентами та підтримка аналітики у локальному режимі без постійного підключення до хмари (див. рис. 1.6).

Рис. 1.11 – Структура системи Heatmiser Neo із багатозональним контролем

Для порівняння основних характеристик наведених систем та розроблюваної системи автоматизації опалення з прогнозуванням температурних умов використано узагальнену аналітичну таблицю (табл. 1.2).

Таблиця 1.2 – Порівняльна характеристика існуючих рішень і розроблюваної системи

Система	Прогнозування температури	Підтримка IoT-пристроїв	Хмарна аналітика	Геолокаційне керування	Відкритість API	Рівень інтеграції
Google Nest Thermostat	Так	Високий	Так	Ні	Частково	Високий
Tado Smart Thermostat	Так	Високий	Так	Так	Середній	Високий
Ecobee Smart Thermostat	Так	Високий	Так	Так	Так	Високий
Honeywell Evohome	Частково	Середній	Частково	Ні	Ні	Середній

Heatmiser Neo	Так	Середній	Частково	Ні	Так	Середній

Продовження таблиці 1.2

Розроблювана система	Так (на основі локальної та хмарної аналітики)	Високий	Так	Так	Так (REST API)	Високий

Проведений аналіз підтверджує, що наявні рішення орієнтовані переважно на комерційні потреби користувача й мають обмеження у відкритості коду та можливостях інтеграції з зовнішніми аналітичними системами. Запропонована розробка спрямована на усунення цих недоліків шляхом поєднання модульної архітектури, відкритих інтерфейсів і прогнозної аналітики, здатної адаптувати режими роботи системи до реальних умов експлуатації. У подальшому виклад буде присвячено моделюванню предметної області, аналізу функціональних вимог та постановці задачі проектування, що забезпечать формування технічного підґрунтя для реалізації програмної системи.

1.4 Моделювання предметної області

Моделювання предметної області системи автоматизації опалення з прогнозуванням температурних умов здійснюється для формалізації взаємодії між користувачами, пристроями та сервісами, що утворюють інтегроване середовище керування тепловими процесами. Основна увага приділена визначенню ролей, сценаріїв використання та потоків даних, які забезпечують функціонування системи в реальному часі. На діаграмі прецедентів на рис. 1.12 показано взаємодію між головними акторами - користувачем, адміністратором будівлі, сервісним інженером та зовнішніми службами, такими як погодний сервіс, енергопостачальник, служба аутентифікації та IoT-пристрої.

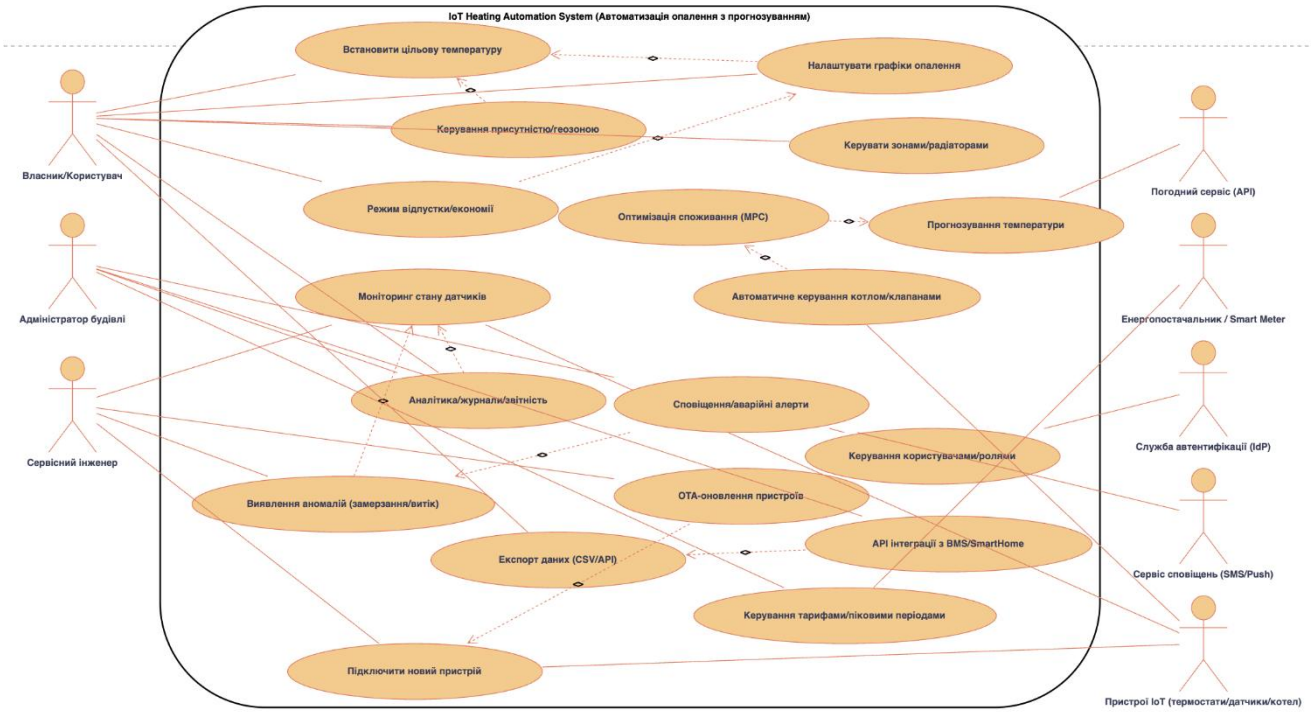


Рис. 1.12 – UML-діаграма прецедентів системи автоматизації опалення з прогнозуванням

Кожен актор реалізує окремий набір функцій: користувач керує температурами та зонами, адміністратор контролює стан сенсорів, а сервісний інженер відповідає за обслуговування й оновлення пристроїв. Таке моделювання дозволяє відобразити як користувацькі, так і технічні сценарії функціонування системи [8].

Подальший етап моделювання реалізовано через діаграму послідовності (рис. 1.13), що демонструє порядок взаємодії компонентів під час встановлення цільової температури та оновлення розкладу. Комунікація здійснюється через API-шлюз, який передає запити від мобільного застосунку до хмарного сервера, де відбувається авторизація користувача, обробка запиту та отримання прогнозу від погодного сервісу. Модуль MPC (Model Predictive Control) формує оптимізаційний план, який через MQTT-шлюз надсилається до термостатів і котлів. Взаємодія компонентів відбувається циклічно, з повторною оптимізацією при виявленні відхилень або аномалій у температурних показниках, що представлено на рис.1.8.

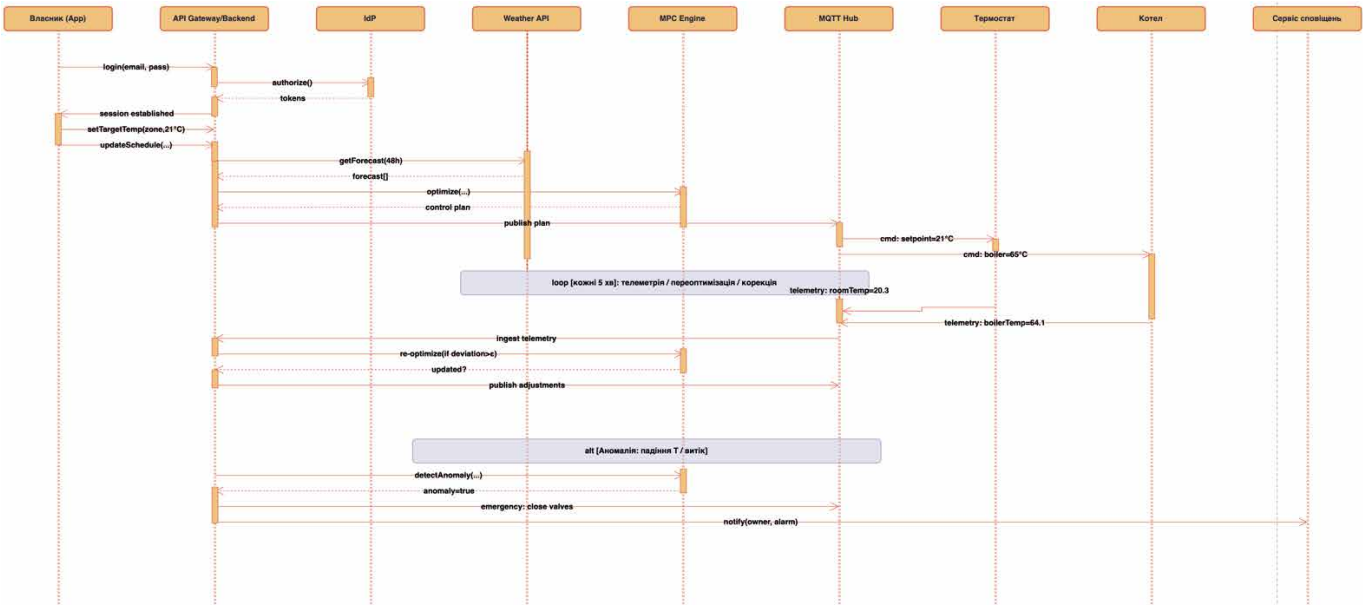


Рис. 1.13 – Діаграма послідовності процесу прогнозування та керування опаленням

Динамічні аспекти роботи системи деталізовано за допомогою діаграми активності (рис. 1.14), яка відображає логіку прийняття рішень у процесі оптимізації. Користувач встановлює цільові параметри, система отримує погодні прогнози, оцінює стан пристроїв, виконує оптимізацію за критеріями енергоефективності та комфорту, а у випадку аномалії (наприклад, падіння температури чи витоку) здійснює аварійне закриття клапанів і надсилає сповіщення. Таким чином, модель описує замкнений цикл збору даних, аналізу, керування та контролю, що відповідає принципам кіберфізичних систем.

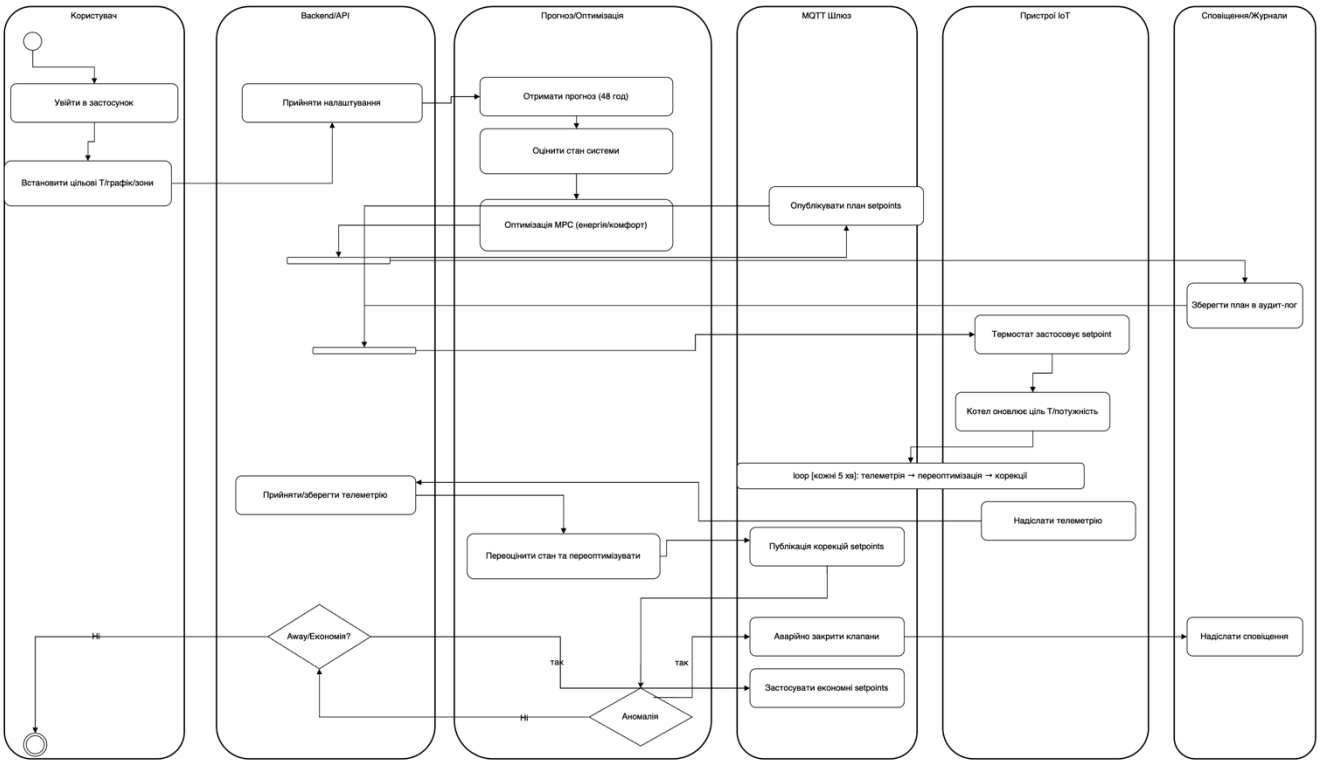


Рис. 1.14 – Діаграма активності процесів оптимізації, контролю та реагування на аномалії

У таблиці 1.3 наведено відповідність основних елементів моделі їх функціональному призначенню, що дозволяє формалізувати зв’язки між компонентами та оцінити їх внесок у загальну архітектуру. Аналіз моделі демонструє узгодженість між рівнями користувацької взаємодії, передаванням даних і процесами аналітики. Отримані результати створюють основу для подальшого аналізу вимог і постановки задачі розроблення системи, що забезпечить побудову її логічної та фізичної моделі.

Таблиця 1.3 – Функціональне призначення основних компонентів моделі системи

Компонент	Призначення	Тип взаємодії	Ключові процеси
Користувач / застосунок	Керування параметрами, моніторинг	REST / HTTPS	Задання температури, розкладу
Контролер IoT	Збір і передавання даних сенсорів	MQTT / CoAP	Фільтрація, буферизація, відправка
Хмарна аналітика	Прогнозування та оптимізація	API / AI-модель	Обчислення MPC-плану

Продовження таблиці 1.3

Пристрої опалення	Виконання команд керування	MQTT	Нагрів, регулювання потужності
Сервіс сповіщень	Повідомлення користувача	SMS / Push	Аварійні та статусні сповіщення

Моделювання предметної області дало змогу структурно описати функціональні взаємозв'язки системи, визначити її динаміку та підґрунтя для розроблення вимог і подальшого проєктування програмної архітектури. Наступом етапом буде визначення вимог системи у пункті 1.4.

1.5 Аналіз вимог програмної системи

Аналіз вимог до програмної системи автоматизації опалення з прогнозуванням температурних умов є ключовим етапом проєктування, який дозволяє визначити межі функціональності, якість роботи, технічні параметри та апаратну сумісність майбутнього рішення. Розроблювана система орієнтована на реалізацію інтелектуального керування температурними режимами в реальному часі, використовуючи сенсорну мережу, хмарну аналітику та модуль прогнозування на основі машинного навчання.

Функціональні вимоги відображають основний набір можливостей, які система повинна забезпечити для користувача, адміністратора та сервісного інженера. Вони охоплюють усі процеси - від реєстрації користувачів і збору даних із сенсорів до аналітики, прогнозування температурних умов і формування керуючих команд. Ці вимоги наведено у таблиці 1.4.

Таблиця 1.4 – Функціональні вимоги системи автоматизації опалення

№	Функція	Опис	Результат виконання
1	Реєстрація та автентифікація користувачів	Забезпечення безпечного доступу через механізм авторизації з використанням PyQtAuth	Ідентифікований вхід у систему

Продовження таблиці 1.4

2	Встановлення цільових температур і графіків	Формування гнучких сценаріїв опалення залежно від часу доби та присутності користувачів	Автоматизоване регулювання зон
3	Збір телеметрії з IoT-пристроїв	Отримання температурних, вологісних та енергетичних даних через MQTT-шлюз	Збереження даних у базі SQLite
4	Прогнозування температурних умов	Аналіз історичних даних та прогнозування за допомогою бібліотек NumPy і Scikit-learn	Оптимізація споживання енергії
5	Аналітика та звітність	Візуалізація результатів за допомогою Matplotlib і PyQtGraph	Побудова графіків, оцінка ефективності
6	Виявлення аномалій	Автоматичне виявлення витоків або критичних падінь температури	Генерація аварійних сповіщень користувачу

Нефункціональні вимоги визначають якісні характеристики системи, що забезпечують її продуктивність, стабільність, безпеку та зручність використання. Особливу увагу приділено масштабованості та безпеці обміну даними через протоколи HTTPS і MQTT із підтримкою TLS. Ці параметри відображено у таблиці 1.5.

Таблиця 1.5 – Нефункціональні вимоги програмної системи

№	Категорія	Вимога	Критерій прийнятності
1	Продуктивність	Затримка відповіді інтерфейсу не перевищує 2 секунд	Менше 2 с при 100 активних користувачах
2	Безпека	Реалізація автентифікації через токени OAuth 2.0	100% запитів проходять перевірку
3	Масштабованість	Підтримка підключення понад 1000 пристроїв одночасно	Система стабільна під навантаженням
4	Надійність	Автоматичне відновлення з'єднання після втрати мережі	Реконект ≤10 секунд
5	Зручність використання	Інтерфейс на базі PyQt6 з адаптивною верстkою	Повна функціональність на екранах ≥720p

Технічні вимоги конкретизують програмні компоненти, інструменти та бібліотеки, які забезпечують реалізацію системи на мові Python. Основу складає графічний інтерфейс користувача, побудований за допомогою PyQt6, локальна база даних SQLite, а також аналітичний блок із використанням NumPy, Pandas, Scikit-learn та Matplotlib. Технічні вимоги представлено у таблиці 1.6.

Таблиця 1.6 – Технічні вимоги до програмної системи

№	Компонент	Технологія / Бібліотека	Призначення
1	Інтерфейс користувача	PyQt6	Реалізація графічного застосунку
2	База даних	SQLite3	Збереження конфігурацій, телеметрії, журналів
3	Комунікаційний модуль	MQTT (paho-mqtt)	Зв’язок із сенсорами та контролерами
4	Аналітика даних	NumPy, Pandas, Scikit-learn	Аналіз і прогнозування температур
5	Візуалізація	Matplotlib, PyQtGraph	Графічне представлення даних
6	Хмарна взаємодія	Requests, Flask API	Інтеграція з погодними та енергетичними сервісами

Окрім програмних компонентів, система потребує відповідного апаратного забезпечення, що забезпечить обробку даних у режимі реального часу та безперервну комунікацію між пристроями. Апаратні вимоги наведено у таблиці 1.7.

Таблиця 1.7 – Апаратні вимоги системи автоматизації опалення

№	Компонент	Мінімальні характеристики	Рекомендовані характеристики
1	Серверна станція	CPU ≥ 4 ядра, RAM ≥ 8 GB	CPU ≥ 8 ядер, RAM ≥ 16 GB
2	IoT-контролер	ESP32 або Raspberry Pi	Wi-Fi, датчики температури/вологості
3	Клієнтський ПК	2 GHz, RAM 4 GB	3 GHz, RAM 8 GB, GPU прискорення
4	Сенсорна мережа	Датчики DHT22, BMP280	Підтримка калібрування, низьке енергоспоживання

У результаті аналізу вимог сформовано узгоджене технічне бачення майбутньої системи, що об’єднує компоненти прогнозування, збору даних та адаптивного керування. Така архітектура забезпечує енергоефективність, надійність і масштабованість рішення, побудованого на відкритих бібліотеках Python і технологіях інтернету речей. Заключним етапом даного розділу є постановка завдання, у межах якої буде деталізовано алгоритмічну структуру, схему взаємодії модулів і логічну модель системи.

1.6 Постановка завдання

Постановка завдання для магістерської кваліфікаційної роботи з розроблення системи автоматизації опалення з прогнозуванням температурних умов на основі технологій інтернету речей полягає у визначенні мети, вихідних даних, об’єкта, предмета та комплексу дослідницьких питань, вирішення яких забезпечить досягнення очікуваного результату. Метою дослідження є створення інтелектуальної програмної системи, здатної аналізувати дані з сенсорної мережі, прогнозувати температурні зміни та формувати оптимальні керуючі дії для підтримання комфортного мікроклімату при мінімальних енерговитратах.

Вихідні дані до магістерської кваліфікаційної роботи включають кілька груп джерел, що забезпечують достовірність аналітичної та модельної частин. По-перше, це набори даних від IoT-датчиків, які передають у режимі реального часу показники температури, вологості, тиску та стану обладнання. По-друге, історичні кліматичні дані, що зберігаються у відкритих метеорологічних архівах (NOAA, OpenWeatherMap, Meteostat), використовуються для тренування моделей прогнозування. По-третє, застосовуються технічні дані системи опалення, зокрема параметри теплових котлів, ефективність теплообмінників, дані про тепловтрати будівлі. Крім того, для калібрування аналітичних моделей використовуються відкриті набори даних, що містять зразки енергоспоживання у подібних кліматичних і конструктивних умовах.

На основі зібраних даних формулюється перелік дослідницьких питань, які визначають логіку наукового пошуку та напрям практичної реалізації:

1. Аналіз ключових проблем автоматизації опалювальних систем, які можна розв'язати за допомогою інтелектуального аналізу даних, зокрема перевитрати енергії, неадаптивність до погодних умов і відсутність персоналізованого регулювання залежно від присутності людей у приміщенні.

2. Застосування методів OLAP та Data Mining для багатовимірного аналізу історичних даних з метою виявлення закономірностей у зміні температури та рівнях споживання енергії, що дозволяє формувати адаптивні сценарії управління.

3. Оцінка впливу зовнішніх і поведінкових факторів - температури повітря, вологості, часу доби, присутності або відсутності людей - на динаміку системи опалення, що дає змогу побудувати регресійну модель енергоспоживання.

4. Розроблення моделі прийняття рішень для автоматизованого керування системою, яка базується на алгоритмах оптимізації енерговитрат із використанням машинного навчання (Scikit-learn, TensorFlow) та адаптивного контролю на основі Model Predictive Control (MPC).

Для реалізації системи обрано стек технологій (який буде детальніше розглянутий у третьому розділі) на мові Python, що включає бібліотеки PyQt6 для побудови графічного інтерфейсу користувача, SQLite3 для зберігання даних телеметрії, NumPy і Pandas для аналітичної обробки, Scikit-learn для побудови моделей прогнозування, Matplotlib для візуалізації результатів, а також paho-mqtt для комунікації між контролерами IoT та аналітичним сервером. Така конфігурація забезпечує збалансованість між легкістю розгортання, швидкістю обробки даних і масштабованістю системи.

1.7 Висновки до першого розділу

У першому розділі було проведено комплексний системний аналіз предметної області автоматизації опалення з прогнозуванням температурних умов на основі технологій Інтернету речей. Проаналізовано сучасний стан галузі, визначено ключові тенденції розвитку інтелектуальних систем керування мікрокліматом, а також основні проблеми, що потребують вирішення для підвищення енергоефективності будівель та оптимізації споживання ресурсів.

У результаті огляду існуючих рішень виявлено, що більшість комерційних і промислових систем опалення зосереджені на ручному налаштуванні або обмеженому автоматичному контролі без урахування зовнішніх чинників і прогнозів. Розглянуто системи таких виробників, як Nest, Tado, Ecobee, Honeywell Evohome та Heatmiser, які демонструють ефективність завдяки адаптивним алгоритмам і сенсорній інтеграції, однак не забезпечують комплексного підходу до аналізу даних та прогнозування температури з урахуванням історичних і зовнішніх показників. Це обґрунтовує необхідність розроблення більш гнучкої, масштабованої системи, побудованої на принципах розподіленої аналітики та машинного навчання.

У ході моделювання предметної області було сформовано основні сутності, процеси та інформаційні потоки системи, що дало змогу описати взаємозв’язки між користувачами, сенсорами, контролерами, аналітичними модулями та базою даних. Побудовані UML-діаграми використання, послідовності та активності відобразили логіку функціонування системи – від збору IoT-даних і їх передавання через MQTT-протокол до обробки, прогнозування та візуалізації результатів у клієнтському застосунку.

Проведений аналіз вимог дозволив сформулювати як функціональні, так і нефункціональні вимоги до системи, включно з технічними та безпековими параметрами. Визначено показники продуктивності, затримки обробки даних, точності прогнозів, рівня відмовостійкості та гарантії безпеки з’єднань. Постановка завдання окреслила мету дослідження – створення інтелектуальної

IoT-системи, здатної не лише автоматизувати процеси опалення, а й прогнозувати майбутні температурні зміни для оптимального використання енергоресурсів.

Таким чином, у результаті виконання першого розділу сформовано концептуальні та науково-методологічні засади для побудови системи, визначено її об’єкт, предмет, мету, основні функціональні модулі й архітектурні принципи. Отримані результати стали основою для подальшого етапу проєктування логічної моделі даних, розроблення UML-діаграм програмних компонентів і створення повноцінної архітектури програмного забезпечення системи автоматизації опалення.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Логічна модель даних у вигляді ER-діаграми

Логічна модель даних системи автоматизації опалення з прогнозуванням температурних умов на основі технологій Інтернету речей відображає структуру інформаційних об’єктів, що формують основу для збору, обробки, аналізу та прогнозування показників мікроклімату. Побудована модель (рис. 2.1) забезпечує системну узгодженість усіх компонентів - від рівня сенсорних пристроїв до аналітичних модулів прогнозування, створюючи єдину інформаційну базу для управління енергоспоживанням і підтримання комфортних умов.

На рис. 2.1 наведено ER-діаграму логічної моделі даних системи, що демонструє зв’язки між сутностями та ієрархію потоків даних у межах архітектури IoT-контролю.

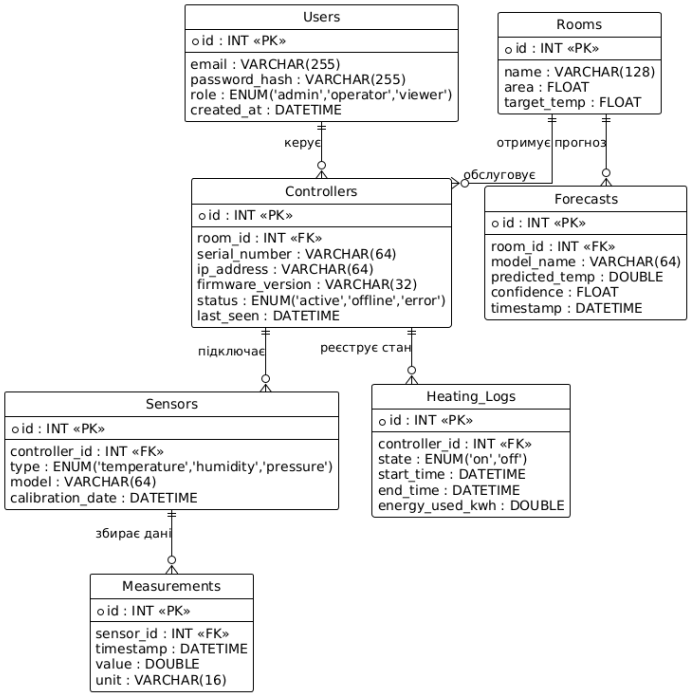


Рис. 2.1 – Логічна модель даних системи автоматизації опалення

У межах моделі виділено ключові сутності: Users, Rooms, Controllers, Sensors, Measurements, Forecasts та Heating_Logs, що охоплюють повний цикл

роботи системи. Сутність Users забезпечує авторизацію користувачів і визначає їх ролі в межах керування IoT-інфраструктурою. Rooms містить опис приміщень, для яких ведеться моніторинг та регулювання температурних параметрів. Controllers виступають центральними вузлами IoT-мережі, що керують сенсорами та фіксують зміну станів опалювальних приладів. Сутність Sensors відображає характеристики фізичних вимірювальних пристроїв, а Measurements - накопичені дані вимірювань (температура, вологість, тиск) у часовій динаміці. Forecasts зберігає результати аналітичного прогнозування на основі моделей машинного навчання, а Heating_Logs фіксує історію вмикання опалення й витрат енергії.

Структура ER-моделі побудована з урахуванням принципів нормалізації до третьої нормальної форми (3НФ), що усуває надлишковість даних і забезпечує логічну цілісність інформації. Кожна сутність має чітко визначені первинні й зовнішні ключі, завдяки чому підтримується коректність зв’язків один-до-багатьох між рівнями користувачів, контролерів і сенсорів. Такий підхід дозволяє реалізувати масштабовану базу даних, готову до розширення новими типами сенсорів, прогнозних моделей або сценаріїв автоматизації.

У табл. 2.1 представлено структурну специфікацію основних сутностей і атрибутів логічної моделі.

Таблиця 2.1 – Основні сутності та їх атрибути логічної моделі даних

Сутність	Основні атрибути	Ключові поля	Призначення
Users	email, password_hash, role, created_at	id (PK)	Зберігає облікові записи користувачів системи керування
Rooms	name, area, target_temp	id (PK)	Описує приміщення, для яких здійснюється моніторинг і прогноз
Controllers	room_id, serial_number, ip_address, firmware_version, status, last_seen	id (PK), room_id (FK)	Відповідає за комунікацію між сенсорами та сервером

Продовження таблиці 2.1

Sensors	controller_id, type, model, calibration_date	id (PK), controller_id (FK)	Зберігає інформацію про сенсорні пристрої
Measurements	sensor_id, timestamp, value, unit	id (PK), sensor_id (FK)	Містить телеметричні дані, отримані з сенсорів
Forecasts	room_id, model_name, predicted_temp, confidence, timestamp	id (PK), room_id (FK)	Зберігає результати прогнозів температурних показників
Heating_Logs	controller_id, state, start_time, end_time, energy_used_kwh	id (PK), controller_id (FK)	Реєструє режими роботи системи та витрати енергії

Розроблена логічна модель даних забезпечує узгоджене функціонування компонентів IoT-системи, формуючи єдине інформаційне середовище для аналізу, прогнозування та керування тепловими процесами. Така модель створює основу для подальшого переходу до фізичної моделі бази даних і реалізації аналітичних модулів прогнозного керування.

2.2 Діаграма класів і кооперації

Діаграма класів є центральним елементом структурного проєктування системи автоматизації опалення з прогнозуванням температурних умов. Вона формалізує архітектуру програмних компонентів, описує взаємозв’язки між об’єктами та визначає їхні атрибути і методи. Такий підхід забезпечує цілісність моделі та її узгодженість із логічною структурою даних, що була нормалізована до третьої нормальної форми (3НФ) для уникнення дублювання інформації та підвищення узгодженості між модулями.

На рис. 2.2 подано UML-діаграму класів, яка відображає логіку взаємодії між основними об’єктами системи: User, Zone, Room, Sensor, TelemetryRecord, IoTController, HVACUnit, Forecast, ControlPolicy (MPC) та SetpointPlan. Модель побудована за принципами об’єктно-орієнтованого аналізу, де кожен клас

відповідає окремій функціональній одиниці. Зокрема, користувач (User) формує плани температурних режимів (SetpointPlan) для певних зон (Zone), які контролюються IoT-пристроями (IoTController) та сенсорними модулями (Sensor). Модуль Forecast здійснює прогнозування температури на основі історичних телеметричних даних (TelemetryRecord), тоді як клас ControlPolicy (MPC) реалізує алгоритм керування на основі моделі прогнозного контролю. У результаті формується команда, що передається до HVACUnit для регулювання потужності та режиму роботи системи опалення.

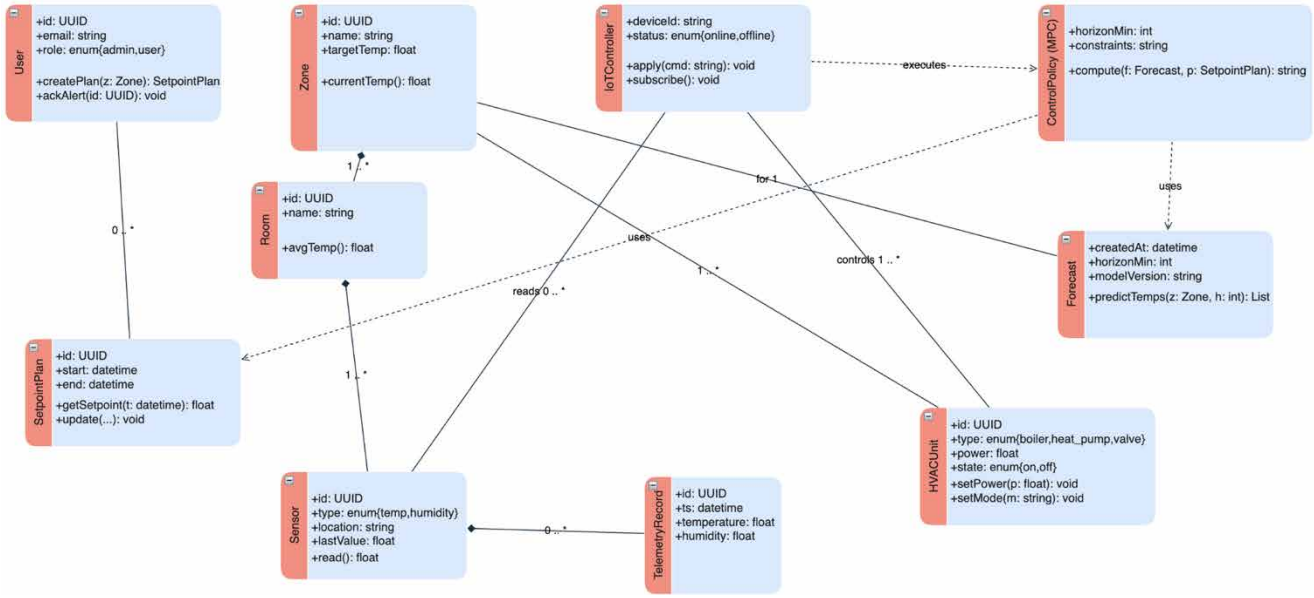


Рис. 2.2 – UML-діаграма класів системи автоматизації опалення

На рис. 2.3 представлено діаграму кооперації, яка деталізує послідовність взаємодії об’єктів під час прогнозного керування температурними параметрами. Зокрема, користувач ініціює створення плану температурного режиму, після чого система використовує дані сенсорів і прогностичний модуль для розрахунку оптимальної дії. Отримані команди передаються контролеру, який взаємодіє з фізичними нагрівальними пристроями через MQTT або HTTP-інтерфейси.

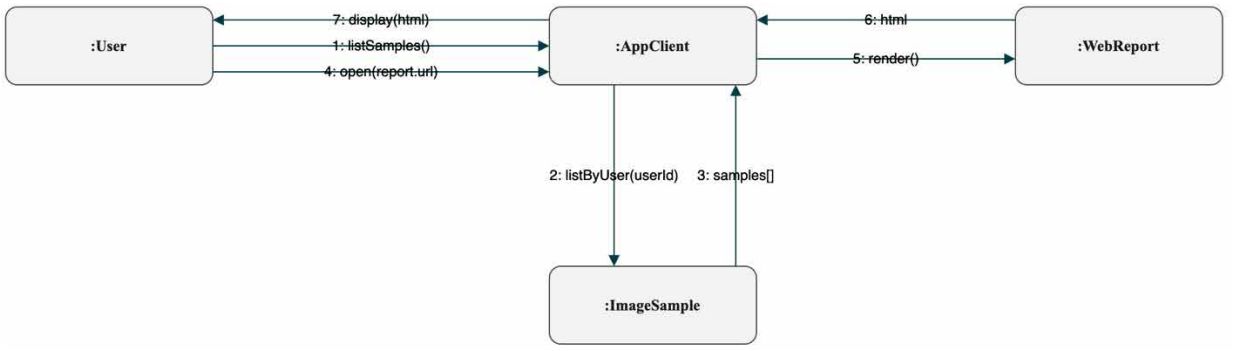


Рис. 2.3 – UML-діаграма кооперації процесу створення плану керування

На рис. 2.4 наведено іншу кооперацію, що описує сценарій збору телеметричних даних: сенсор фіксує поточні значення температури та вологості, створює запис TelemetryRecord і передає дані до модуля прогнозування. Це дозволяє системі реалізувати циклічний процес самонавчання та адаптивного регулювання.

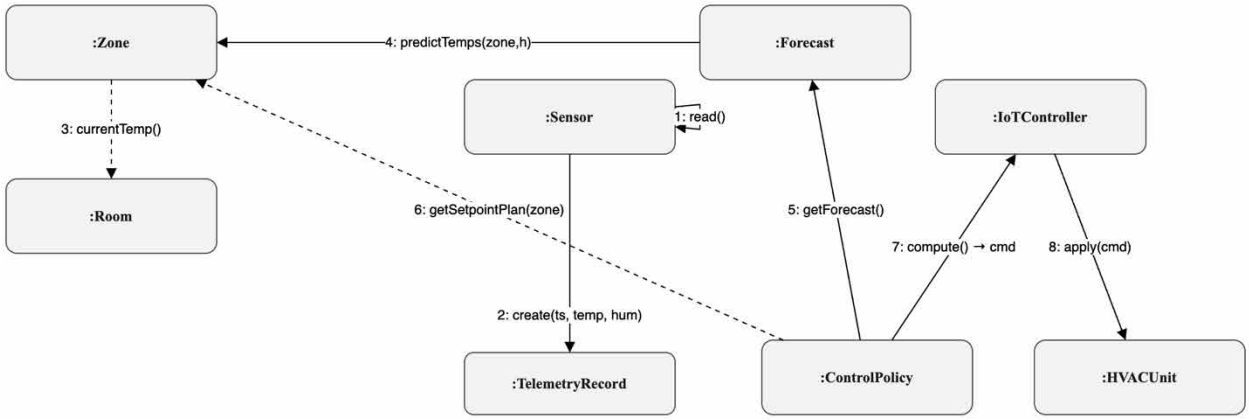


Рис. 2.4 – UML-діаграма кооперації процесу збору телеметричних даних

Додатково на рис. 2.5 відображено кооперацію взаємодії між модулем прогнозування та компонентом ControlPolicy, де прогнозовані температурні ряди (predictTemps()) передаються у модуль керування, який виконує оптимізаційний розрахунок і надсилає результат у вигляді керувальної команди контролеру.

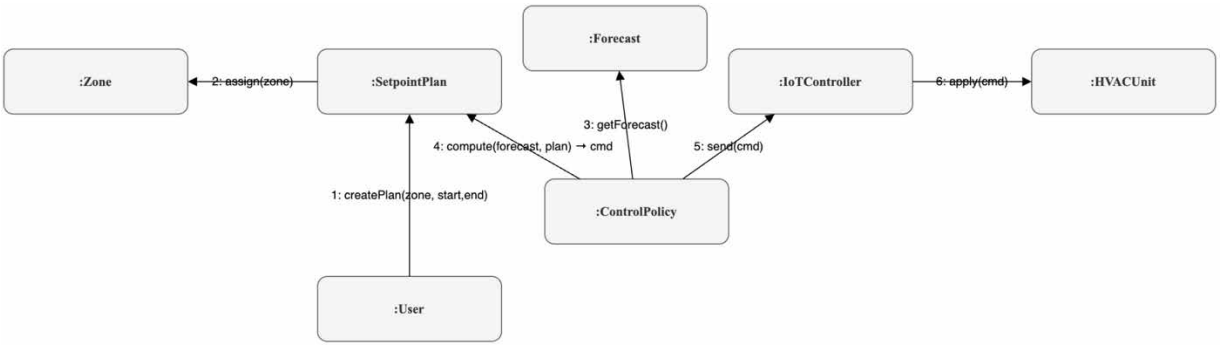


Рис. 2.5 – UML-діаграма кооперації взаємодії прогнозного та керуючого модулів

Усі класи системи характеризуються чітко визначеними атрибутами, методами та зв'язками. Їхні взаємозв'язки, представлені у табл. 2.2, забезпечують повну структурну цілісність програмної моделі.

Таблиця 2.2 – Основні класи та функціональне призначення

Клас	Основні атрибути	Призначення
User	id, email, role	Формування планів температурного керування, підтвердження сповіщень
Zone	name, targetTemp	Логічне об'єднання кімнат, для яких створюються індивідуальні режими
Room	name, avgTemp()	Представлення фізичних приміщень з вимірювальними пристроями
Sensor	type, location, read()	Збір даних про температуру, вологість, тиск тощо
TelemetryRecord	ts, temperature, humidity	Збереження виміряних показників у базі даних
Forecast	modelVersion, predictTemps()	Формування прогнозу на основі попередніх вимірювань
ControlPolicy (MPC)	horizonMin, compute()	Алгоритм прогнозно-оптимізаційного керування температурою
IoTController	deviceId, apply()	Зв'язок між програмними модулями і фізичними пристроями
HVACUnit	type, power, setMode()	Регулювання роботи нагрівальних і охолоджувальних пристроїв
SetpointPlan	start, end, getSetpoint()	Задання цільових температурних траєкторій для зон

Діаграма класів і кооперацій системи створюють повну структурно-поведінкову модель, яка відображає усі ключові процеси автоматизованого

прогнозно-аналітичного керування мікрокліматом. Вона забезпечує реалізацію гнучкої, масштабованої архітектури, що поєднує IoT-технології, телеметрію, машинне навчання та елементи MPC-контролю для підвищення енергоефективності системи опалення.

2.2 Діаграма компонентів

Діаграма компонентів відображає архітектурну організацію системи автоматизації опалення з прогнозуванням температурних умов на основі технологій Інтернету речей, її логічне структурування та взаємозв'язки між основними програмними і апаратними підсистемами. Ця модель визначає взаємодію між компонентами різних рівнів - від IoT-пристроїв збору телеметрії до аналітичних і керуючих сервісів, що реалізують прогнозно-оптимізаційне управління мікрокліматом. Такий підхід дає змогу досягти високої гнучкості, масштабованості та незалежності компонентів, що відповідає принципам сервісно-орієнтованої архітектури (SOA).

На рис. 2.6 подано UML-діаграму компонентів, яка відображає основні модулі системи та інтерфейси їхньої взаємодії. Архітектура побудована за принципами мікросервісного підходу, де кожен компонент виконує окрему функцію, що може бути незалежно розгорнута, оновлена та протестована. Взаємодія між сервісами здійснюється через API-шлюз (FastAPI), який координує запити між клієнтськими застосунками, аналітичними модулями та сховищами даних.

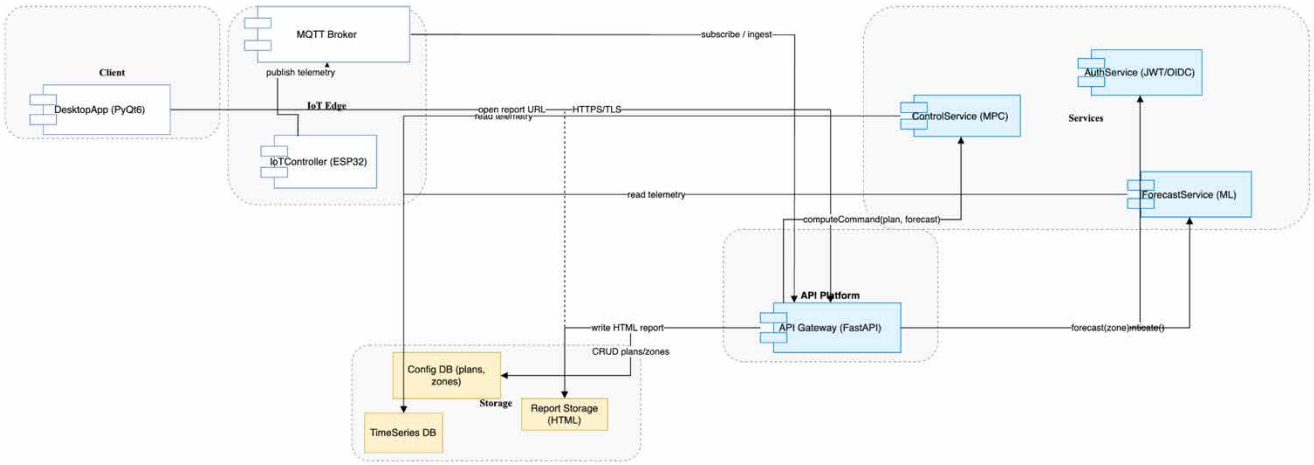


Рис. 2.6 – Діаграма компонентів системи автоматизації опалення

На рівні Client функціонує десктопний застосунок DesktopApp (PyQt6), який є основним інтерфейсом користувача. Через нього здійснюється вхід до системи, формування планів опалення, моніторинг параметрів та перегляд звітів. Застосунок використовує протокол HTTPS/TLS для безпечної взаємодії з сервером і відображення HTML-звіту, що генерується автоматично за результатами аналітики.

Рівень IoT Edge відповідає за збирання й попередню обробку даних безпосередньо на периферійних пристроях. Контролер IoTController (ESP32) приймає сигнали від сенсорів температури й вологості, передає телеметрію через MQTT Broker, а також отримує команди від центрального серверу для регулювання режимів опалення. Цей рівень забезпечує низьку затримку передачі даних і підтримку автономної роботи у випадку тимчасових втрат зв’язку.

Рівень Services / API Platform реалізує логіку аналітичної обробки, прогнозування та керування системою. Основним компонентом виступає API Gateway (FastAPI) -центральний вузол маршрутизації запитів, який координує взаємодію клієнтів із внутрішніми сервісами. На його основі реалізовані модулі:

– AuthService (JWT/OIDC) - відповідає за автентифікацію користувачів, видачу токенів доступу та захист API-запитів;

– ForecastService (ML) - використовує моделі машинного навчання для прогнозування температурних трендів у зонах, аналізуючи часові ряди та погодні показники;

– ControlService (MPC) - реалізує алгоритм Model Predictive Control для обчислення оптимальних команд керування опалювальними пристроями на основі прогнозу й поточних показників;

– Storage Subsystem – включає Config DB (зони, плани, конфігурації), TimeSeries DB (телеметричні дані) та Report Storage (HTML), що забезпечує зберігання аналітичних звітів і історичних даних.

Інформаційна взаємодія між компонентами здійснюється за допомогою двох ключових протоколів - MQTT для публікації та підписки на телеметрію між контролерами й брокером та HTTPS/TLS для взаємодії з серверними компонентами. Завдяки цьому забезпечується гарантована доставка даних, їх шифрування та підтримка наскрізної безпеки системи.

У табл. 2.3 наведено узагальнену характеристику основних компонентів та їхніх функціональних ролей у системі.

Таблиця 2.3 – Основні компоненти системи автоматизації опалення

Компонент	Функціональне призначення	Тип взаємодії
DesktopApp (PyQt6)	Інтерфейс користувача, створення планів, перегляд аналітичних звітів	REST / HTTPS
IoTController (ESP32)	Збір телеметрії, виконання команд керування	MQTT / TLS
MQTT Broker	Передача повідомлень між сенсорами та сервером	MQTT
API Gateway (FastAPI)	Централізована маршрутизація запитів, CRUD-операції, взаємодія з ML-та MPC-модулями	REST API
ForecastService (ML)	Прогнозування температурних параметрів на основі часових рядів	Internal API
ControlService (MPC)	Розрахунок оптимальних керувальних дій для зон	Internal API
AuthService (JWT/OIDC)	Ідентифікація користувачів, управління доступом	OAuth 2.0 / OIDC

Config DB	Збереження параметрів зон і користувацьких планів	SQL / ORM
TimeSeries DB	Акумуляція телеметричних даних і прогнозних рядів	NoSQL / InfluxDB
Report Storage (HTML)	Збереження сформованих HTML-звітів і графічних результатів	Filesystem

Загальна архітектура системи забезпечує модульність, гнучке масштабування та високу надійність. Взаємодія компонентів через єдину API-платформу спрощує інтеграцію з зовнішніми сервісами (ERP, HRM, SCADA) і підтримує розширення функціональності без порушення цілісності проєкту. Побудована модель компонентів демонструє науково обґрунтовану, технологічно узгоджену структуру програмного комплексу, оптимізовану для інтелектуального керування енергоспоживанням і підвищення ефективності опалювальних систем.

2.4 Діаграма пакетів

Діаграма пакетів відображає логічну структуру системи автоматизації опалення з прогнозуванням температурних умов на основі технологій Інтернету речей, її внутрішню організацію та взаємозв’язки між підсистемами. Метою побудови даної моделі є структуризація програмного забезпечення на взаємопов’язані, але автономні модулі, що забезпечує зрозумілу архітектуру, модульність і масштабованість системи. Завдяки цьому кожен компонент можна розробляти, тестувати та вдосконалювати незалежно, що є принципом сучасної сервісно-орієнтованої архітектури (SOA) і відповідає вимогам надійних IoT-систем.

На рис. 2.7 наведено UML-діаграму пакетів, що демонструє багаторівневу структуру системи. Архітектура поділена на чотири основні рівні -client.desktop, iot.edge, api.backend та persistence, які взаємодіють через чітко визначені інтерфейси й протоколи. Така декомпозиція забезпечує гнучкість при модифікаціях програмного комплексу, мінімізує зв’язність між модулями та

дозволяє ефективно розподіляти обчислювальні навантаження між клієнтськими пристроями, периферією та серверною інфраструктурою.

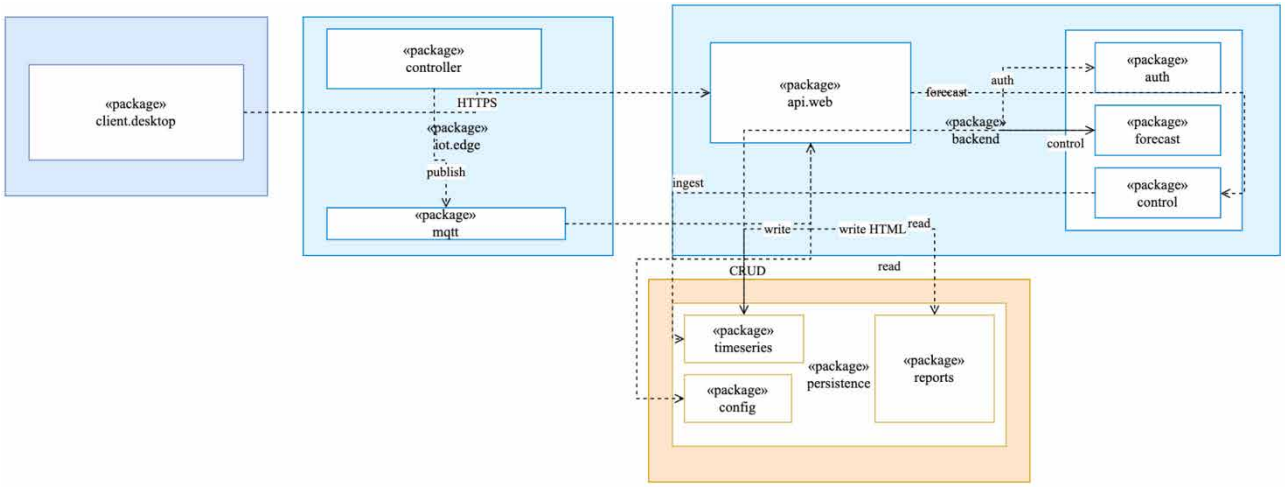


Рис. 2.7 – Діаграма пакетів системи автоматизації опалення

Рівень client.desktop відповідає за взаємодію користувача із системою через десктопний застосунок, реалізований засобами PyQt6. Його функціонал охоплює авторизацію, налаштування зон, створення планів опалення, візуалізацію прогнозів і перегляд HTML-звітів. Клієнт безпосередньо взаємодіє з веб-інтерфейсом системи через захищене HTTPS/TLS-з'єднання, отримуючи доступ лише до відкритих API-методів без прямого втручання у внутрішню логіку.

Пакет iot.edge реалізує периферійну обробку даних, зокрема збір телеметрії з фізичних сенсорів та керування опалювальними пристроями. Його підпакети controller та mqtt відповідають відповідно за взаємодію з мікроконтролерами (ESP32) і публікацію даних до брокера повідомлень. Передача інформації про температуру, вологість і стан системи опалення відбувається за протоколом MQTT, що гарантує низьку затримку і високу стабільність обміну даними навіть у нестійких мережевих умовах.

Рівень api.backend (або server layer) є ядром системи, яке реалізує її бізнес-логіку, алгоритми прогнозування та керування. Його структуровано на декілька взаємопов'язаних підпакетів:

– api.web - відповідає за обробку зовнішніх HTTP-запитів, маршрутизацію між сервісами та взаємодію з клієнтським застосунком;

– backend.auth - реалізує механізми автентифікації й авторизації користувачів через JWT/OIDC, забезпечуючи рольову модель доступу;

– backend.forecast - містить алгоритми машинного навчання для прогнозування температурних змін з використанням часових рядів і даних сенсорів;

– backend.control - реалізує методи прогнозно-оптимізаційного керування (Model Predictive Control), які формують керувальні команди для периферійних контролерів з урахуванням прогнозу та поточних умов.

Комунікація між сервісами відбувається через REST-виклики з внутрішніми авторизаційними перевірками. Таким чином забезпечується узгоджена взаємодія аналітичних, керуючих і користувацьких модулів, що формують цілісний контур «прогнозування -> оптимізація -> дія».

Рівень persistence реалізує централізоване збереження даних у підпакетах config, timeseries та reports. Пакет config містить метадані системи, параметри зон, користувацькі плани і конфігурації контролерів. Timeseriesзберігає великі обсяги телеметрії у вигляді часових рядів, що дає змогу виконувати аналітику й прогнозування на основі історичних даних. Reports відповідає за формування аналітичних звітів у форматі HTML і їх подальше збереження у файловій структурі або хмарному сховищі.

У табл. 2.4 наведено функціональне призначення основних пакетів системи та характер їхньої взаємодії.

Таблиця 2.4 – Основні пакети системи та їх функціональні ролі

Пакет	Призначення	Протокол / взаємодія
client.desktop	Клієнтський застосунок для моніторингу, налаштування зон і перегляду звітів	HTTPS / REST
iot.edge.controller	Збір сенсорних даних, виконання команд від центрального контролера	MQTT / TLS
iot.edge.mqtt	Публікація телеметрії до брокера, забезпечення зв’язку між IoT-вузлами	MQTT
api.web	Маршрутизація клієнтських запитів, взаємодія з аналітичними сервісами	REST API

backend.auth	Автентифікація користувачів, токенізація, розмежування прав доступу	JWT / OIDC
backend.forecast	Прогнозування температури за ML-моделями	Internal API
backend.control	Формування керувальних сигналів на основі прогнозу (MPC)	Internal API
persistence.config	Збереження параметрів зон, планів і конфігурацій	SQL / ORM
persistence.timeseries	Акумуляція телеметрії в часових рядах	NoSQL / InfluxDB
persistence.reports	Формування, збереження та надання аналітичних звітів	HTML / Filesystem

Архітектура пакетів системи побудована за принципами багаторівневої модульності та ізоляції залежностей. Кожен рівень має чітко визначену відповідальність і взаємодіє з іншими лише через стандартизовані інтерфейси, що забезпечує стабільність, безпечність і масштабованість програмного комплексу. Така структура створює єдиний інтегрований простір для обробки IoT-даних, прогнозного аналізу й автоматичного керування, формуючи цілісну, логічно узгоджену систему енергоефективного контролю мікроклімату.

2.5 Висновки до другого розділу

У другому розділі було здійснено системне моделювання програмної системи автоматизації опалення з прогнозуванням температурних умов на основі технологій Інтернету речей. Побудовані моделі – логічна модель даних, діаграми класів, кооперації, компонентів і пакетів – забезпечили всебічне формалізоване представлення архітектури майбутнього програмного комплексу, визначили його структурні зв’язки, інформаційні потоки та функціональні взаємозалежності.

Логічна модель даних (рис. 2.1) дала змогу визначити сутності предметної області, атрибути та зв’язки між ними, що забезпечує узгодженість інформаційних потоків і відображає принципи нормалізації бази даних. Структура таблиць, побудована відповідно до вимог до цілісності й незалежності

даних, унеможливлює дублювання інформації та гарантує коректність під час виконання операцій запису, читання й оновлення. Такий підхід забезпечує оптимальну продуктивність при роботі з телеметрією, прогнозними рядами та історичними звітами.

Діаграми класів і кооперації (рис. 2.2–2.5) описують взаємодію між об’єктами системи на рівні прикладної логіки. Модель класів формалізує програмні сутності – користувача, зони, контролер, сенсори, прогнозні модулі, аналітичні служби та виконавчі пристрої HVAC – а також їхні атрибути, методи й залежності. Діаграми кооперації, у свою чергу, відтворюють послідовність обміну повідомленнями між об’єктами під час виконання ключових сценаріїв, зокрема збору даних, прогнозування температури, формування планів керування та застосування оптимізаційних рішень. Це дозволило формалізувати логіку системи як єдиний цикл «збір даних – прогнозування – керування – аналітика».

Діаграма компонентів (рис. 2.6) відобразила архітектурну структуру системи з позиції розподілу програмних і апаратних модулів між рівнями. Вона показала взаємодію між клієнтським застосунком, периферійними IoT-пристроями, аналітичними сервісами, API-шлюзом і системою збереження даних. Така структуризація забезпечує модульність, ізольованість функцій, можливість гнучкого масштабування та інтеграції з зовнішніми сервісами ERP, SCADA та аналітичними платформами. Реалізація механізмів взаємодії через MQTT і HTTPS гарантує безпечний обмін даними та мінімальні затримки при передачі телеметрії.

Діаграма пакетів (рис. 2.7) деталізувала логічну організацію програмного забезпечення за рівнями client.desktop, iot.edge, api.backend і persistence. Вона формалізувала архітектуру системи на рівні модулів, що визначають структурну незалежність компонентів, узгодженість інтерфейсів і стандартизовані механізми взаємодії між рівнями. Такий підхід створює передумови для впровадження CI/CD-процесів, підвищує надійність системи й забезпечує ефективність розподіленої обробки IoT-даних у режимі реального часу.

У результаті виконаного моделювання було сформовано науково обґрунтовану архітектуру програмного комплексу, що поєднує переваги об’єктно-орієнтованого проектування, сервісно-орієнтованих принципів і сучасних IoT-технологій. Розроблені діаграми дозволяють перейти до етапу алгоритмізації програмних модулів, визначення внутрішніх процесів взаємодії між ними та реалізації функціональної логіки керування опалювальною системою на основі прогнозно-аналітичних моделей. Таким чином, результати другого розділу заклали фундамент для програмної реалізації системи, що відповідає сучасним вимогам енергоефективності, масштабованості, безпеки та інтелектуальної адаптації до зовнішніх умов.

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір технологій та інструментальних засобів реалізації системи

Розроблення системи автоматизації опалення з прогнозуванням температурних умов потребує вибору технологій, здатних забезпечити стабільну обробку телеметричних даних, інтеграцію з IoT-пристроями, реалізацію прогнозних моделей і формування керувальних дій у реальному часі. Стек технологій визначався з урахуванням вимог до масштабованості, надійності, продуктивності та можливості подальшого функціонального розширення, що було встановлено у першому розділі. Основою стала платформа Python, яка поєднує інструменти для машинного навчання, обробки часових рядів, візуалізації даних і швидкого створення прикладних модулів.

Для побудови клієнтської частини обрано PyQt6, що забезпечує формування кросплатформеного інтерфейсу та підтримує інтеграцію з HTML-звітами, графічними панелями моніторингу й модулем відображення телеметрії. Такий вибір дозволяє створити повноцінний десктопний застосунок із мінімальними вимогами до апаратного забезпечення, зберігаючи високу продуктивність і плавність відображення графіків. Локальне збереження конфігурацій, історичних даних та журналів подій реалізовано за допомогою SQLite3, що гармонійно узгоджується з моделлю даних, побудованою у розділі 2, та не вимагає складної серверної інфраструктури при розгортанні системи.

Для прогнозування внутрішньої температури використано алгоритм лінійної регресії (Linear Regression) із бібліотеки Scikit-learn. У процесі моделювання також розглядалися альтернативні підходи — Random Forest та LSTM, проте базова регресійна модель забезпечила найкращий баланс між точністю та швидкістю прогнозу. Вибір цих інструментів обумовлений їх сумісністю з Python, високою швидкістю роботи з матрицями та можливістю

масштабування прогнозних моделей у майбутньому. Візуалізація даних здійснюється засобами Matplotlib і PyQtGraph, що дозволяє формувати інтерактивні графіки, діаграми трендів і аналітичні звіти, інтегровані у користувацький інтерфейс.

Комунікація із сенсорними пристроями здійснюється через MQTT-протокол, реалізований бібліотекою paho-mqtt, яка забезпечує асинхронну передачу телеметрії, низьку затримку обміну та підтримку QoS-рівнів для гарантування доставки повідомлень. Такий вибір повністю узгоджується з апаратною платформою на базі ESP32, яка підтримує енергоефективний збір даних і оновлення станів контролерів. Для інтеграції з погодними ресурсами та зовнішніми сервісами використано Flask API та бібліотеку Requests, що дає змогу виконувати HTTP-запити, отримувати прогнозні значення та включати їх у модель оптимізації температурних режимів.

Окреме місце займають механізми безпеки, що реалізовані через TLS, підтримку токенів доступу та контроль цілісності телеметрії. Таке рішення забезпечує захист даних у каналі зв’язку, унеможливлює несанкціоноване втручання у роботу системи й гарантує коректність команд, які надходять до контролера. Узгодженість між використаними інструментами формує модульну архітектуру, яка підтримує розподілений характер опрацювання даних: частина функцій виконується на IoT-пристроях, тоді як прогнозування та аналітика зосереджені на центральному рівні.

Обраний технологічний стек відображає баланс між гнучкістю, сумісністю, швидкістю розгортання й можливістю інтеграції з зовнішніми системами. Використання Python, PyQt6, SQLite3, MQTT та бібліотек машинного навчання забезпечує створення повноцінної інтелектуальної системи, здатної працювати в реальному часі, прогнозувати зміни температурних умов і приймати оптимізаційні рішення для енергоефективного керування опаленням.

3.2 Інтелектуально-аналітична модель оцінювання реакції системи на основі багатовимірних даних

Багатовимірна модель даних стала фундаментом аналітичної підсистеми, що забезпечує обробку телеметрії, формування похідних показників та реалізацію алгоритмів класифікації й кластеризації для оцінювання реакційної здатності системи опалення. На рис. 3.1 подано зіркоподібну структуру схеми даних, у центрі якої розташована таблиця фактів FactSystemData, що містить значення часу реакції, середні параметри мікроклімату та посилання на вимірні таблиці DimLocation, DimTemperature, DimSettings і DimDate. Така організація забезпечує можливість багатовимірних запитів і створює основу для подальшого інтелектуального аналізу.

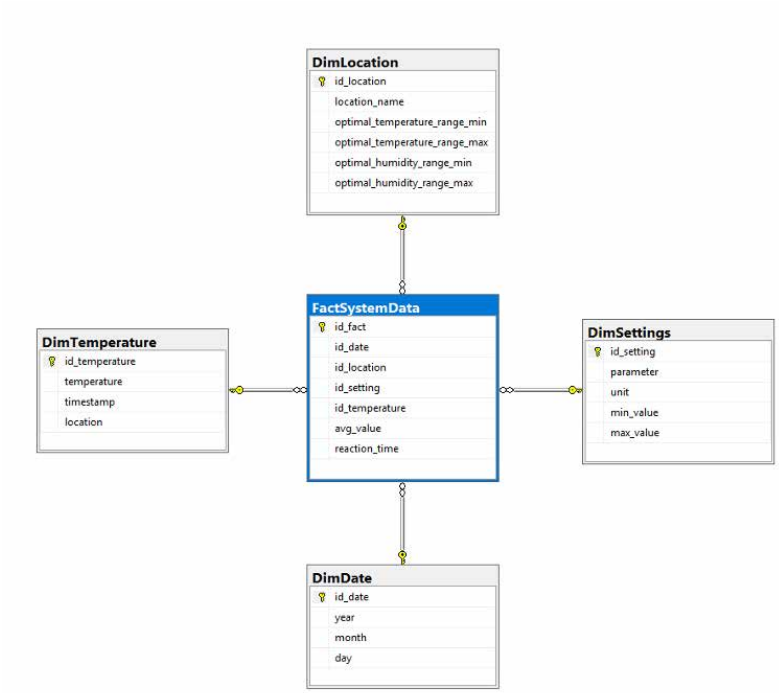
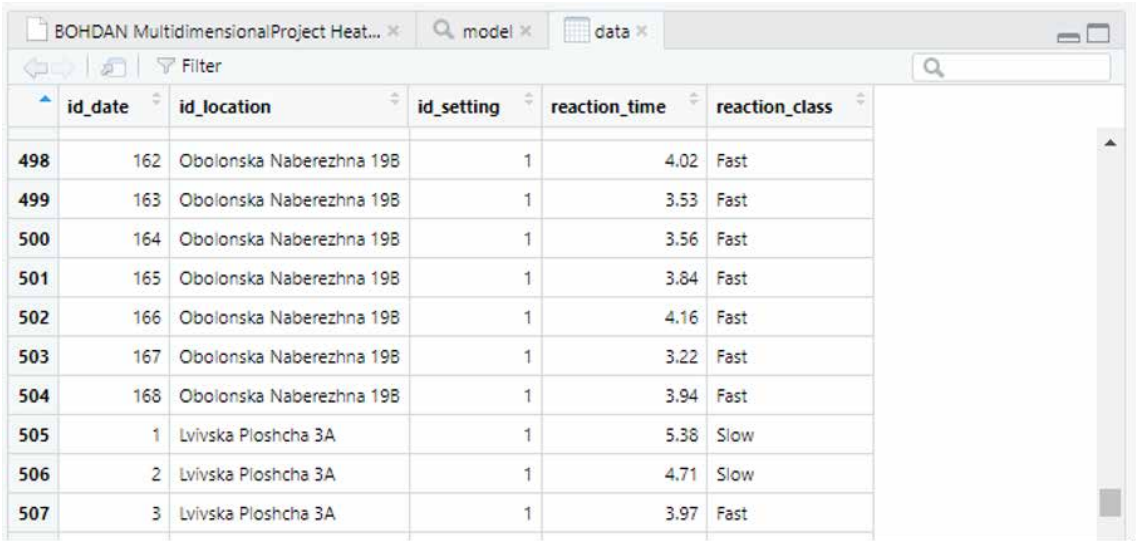


Рис. 3.1 – Багатовимірна модель даних системи автоматизації опалення

На основі цієї структури формується набір фактів для подальшої класифікації. Фрагмент наповнення фактної таблиці, показаний на рис. 3.2, демонструє поєднання часових вимірів, локацій і системних налаштувань із похідною ознакою reaction_class, що автоматично визначається на основі технологічних критеріїв. Ця похідна змінна виступає ключовим параметром для

алгоритмів машинного навчання, оскільки дає змогу формалізувати якість реакції системи в конкретних умовах експлуатації.


BOHDAN MultidimensionalProject Heat... × model × data ×

Filter

	id_date	id_location	id_setting	reaction_time	reaction_class
498	162	Obolonska Naberezhna 19B	1	4.02	Fast
499	163	Obolonska Naberezhna 19B	1	3.53	Fast
500	164	Obolonska Naberezhna 19B	1	3.56	Fast
501	165	Obolonska Naberezhna 19B	1	3.84	Fast
502	166	Obolonska Naberezhna 19B	1	4.16	Fast
503	167	Obolonska Naberezhna 19B	1	3.22	Fast
504	168	Obolonska Naberezhna 19B	1	3.94	Fast
505	1	Lvivska Ploshcha 3A	1	5.38	Slow
506	2	Lvivska Ploshcha 3A	1	4.71	Slow
507	3	Lvivska Ploshcha 3A	1	3.97	Fast

Рис. 3.2 – Фрагмент фактної таблиці з атрибутами часу реакції та класом реакції

Подальший аналіз виконується за допомогою класифікаційних моделей, що розділяють локації на групи Fast та Slow відповідно до ймовірнісних характеристик реакційної здатності. На рис. 3.3 подано результати роботи моделі, де для кожної будівлі визначено клас і обчислено значення апостеріорних ймовірностей. Такий підхід дозволяє створити профіль ефективності для кожної локації, оцінити відповідність роботи системи бажаним SLA та виявити об’єкти з підвищеним ризиком повільної реакції.

Classification Results | Performance Analysis

Location	Class	Probability Fast	Probability Slow
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Fast	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Pecherska 17/3	Slow	0.04	99.96
Prospekt Peremohy 3	Fast	97.82	2.18
Prospekt Peremohy 3	Fast	97.89	2.11
Prospekt Peremohy 3	Fast	97.88	2.12
Prospekt Peremohy 3	Slow	97.89	2.11
Prospekt Peremohy 3	Fast	97.87	2.13
Prospekt Peremohy 3	Fast	97.81	2.19
Prospekt Peremohy 3	Slow	97.93	2.07
Prospekt Peremohy 3	Fast	97.94	2.06
Prospekt Peremohy 3	Fast	97.81	2.19
Prospekt Peremohy 3	Fast	97.91	2.09
Prospekt Peremohy 3	Fast	97.91	2.09
Prospekt Peremohy 3	Fast	97.9	2.1
Prospekt Peremohy 3	Fast	97.84	2.16
Prospekt Peremohy 3	Fast	97.81	2.19
Prospekt Peremohy 3	Fast	97.85	2.15
Prospekt Peremohy 3	Fast	97.85	2.15
Prospekt Peremohy 3	Fast	97.95	2.05

Рис. 3.3 – Результати класифікації локацій за швидкістю реакції системи

Для виявлення прихованих закономірностей застосовано кластерний аналіз часових рядів. На рис. 3.4 наведено метод ліктя та силуетний коефіцієнт, які дозволили формально визначити оптимальну кількість кластерів. Результати показали, що чотирикластерна структура найкраще відображає патерни роботи системи, оскільки забезпечує баланс між щільністю внутрішніх груп і чіткістю міжкластерного розділення.

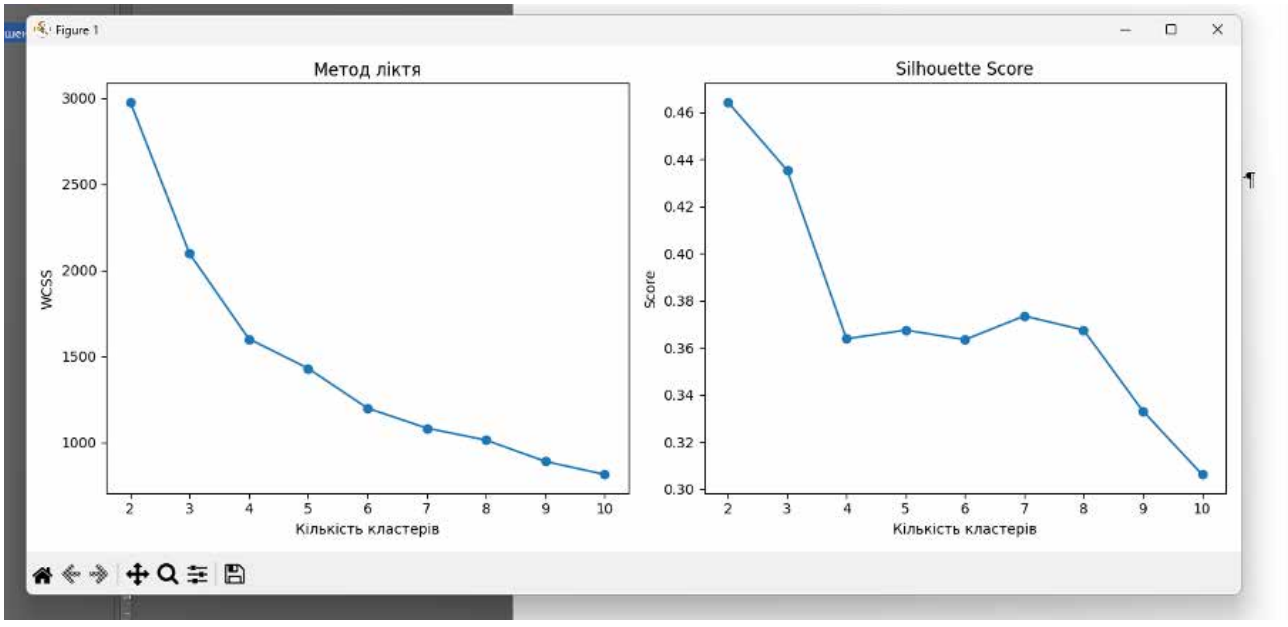


Рис. 3.4 – Визначення оптимальної кількості кластерів за методом ліктя та Silhouette Score

Візуалізація кластерів, подана на рис. 3.5, демонструє просторову поведінку системи за параметрами «температура – вологість – час реакції». Кожен кластер відповідає певному експлуатаційному режиму: стабільно швидкі реакції, нестійкі режими з коливаннями вологості, затримки реакції при підвищених температурах або нестандартні сценарії роботи обладнання. Така інтерпретація є важливим інструментом для подальшої оптимізації роботи системи.

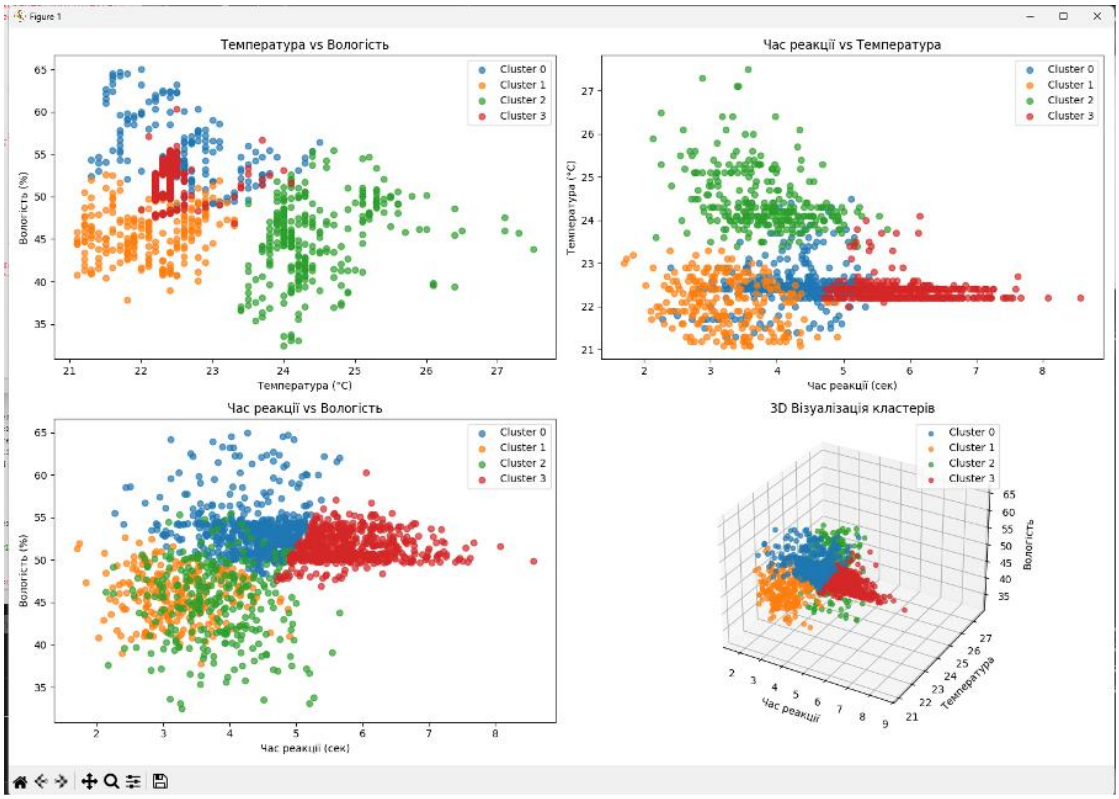


Рис. 3.5 – Кластеризація режимів роботи системи за трьома параметрами мікроклімату

Наукова новизна розробленої підсистеми полягає у впровадженні нового аналітичного контуру, що поєднує OLAP-модель, автоматичне формування похідних характеристик системи, імовірнісну класифікацію реакцій та багатовимірну кластеризацію. На відміну від традиційних систем, які аналізують лише температуру або середні значення за добу, запропонований підхід враховує структуру часової реакції, відхилення від оптимальних налаштувань, волого-температурні патерни та дозволяє створити цифровий профіль роботи кожної локації.

З метою формалізації ключових технічних аспектів новизни підсистеми у таблицю 3.1 інтегровано основні науково-практичні досягнення, що реалізовані у даному дослідженні.

Таблиця 3.1 – Технічні аспекти наукової новизни запропонованої аналітичної моделі

№	Технічний аспект	Суть новизни	Практичний ефект
1	Похідна ознака reaction_class	Автоматичне маркування реакції системи (Fast/Slow)	Оцінювання відповідності системи SLA
2	OLAP-модель FactSystemData	Інтеграція телеметрії, налаштувань і часових вимірів	Багатовимірні аналітичні запити
3	Ймовірнісна класифікація	Розрахунок Probability Fast/Slow для кожної локації	Виявлення ризикових зон
4	Критерії оптимального k	Спільне використання WCSS та Silhouette Score	Формальний підбір кластерної структури
5	3D-кластеризація режимів	Виявлення патернів «температура – вологість – реакція»	Діагностика екстремальних сценаріїв роботи
6	Інтеграція Data Mining з IoT	Узгодження моделей ML із даними сенсорів	Підвищення точності прогнозів у наступних модулях

Підсумовуючи, запропонована аналітична модель дозволяє оцінювати реакційну здатність системи опалення на якісно новому рівні. Поєднання багатовимірної структури даних, імовірнісної класифікації та формальної кластеризації створює потужний інструмент для виявлення неефективних режимів, оптимізації параметрів роботи обладнання та подальшого вдосконалення системи на етапі впровадження прогнозного керування.

3.3 Архітектура системи та проектування функціональних модулів

Архітектура інтелектуальної системи автоматизації опалення побудована на багаторівневій моделі, що об’єднує сенсорну мережу, IoT-контролер, серверні сервіси прогнозування та керування, а також клієнтський застосунок для

моніторингу й аналізу. На рис. 3.6 подано функціональну схему архітектури, у якій визначено основні потоки даних, маршрути комунікації та логічні взаємозв'язки між ключовими компонентами. Архітектура базується на гібридній моделі протоколів: MQTT використовується для низьколатентної доставки телеметрії, тоді як REST забезпечує доступ до ML-сервісів та механізмів оптимізаційного керування. Такий підхід дає можливість інтегрувати результати аналітичного модуля (класифікація реакції, кластери поведінки, прогноз температури) у цикл прийняття рішень у реальному часі.

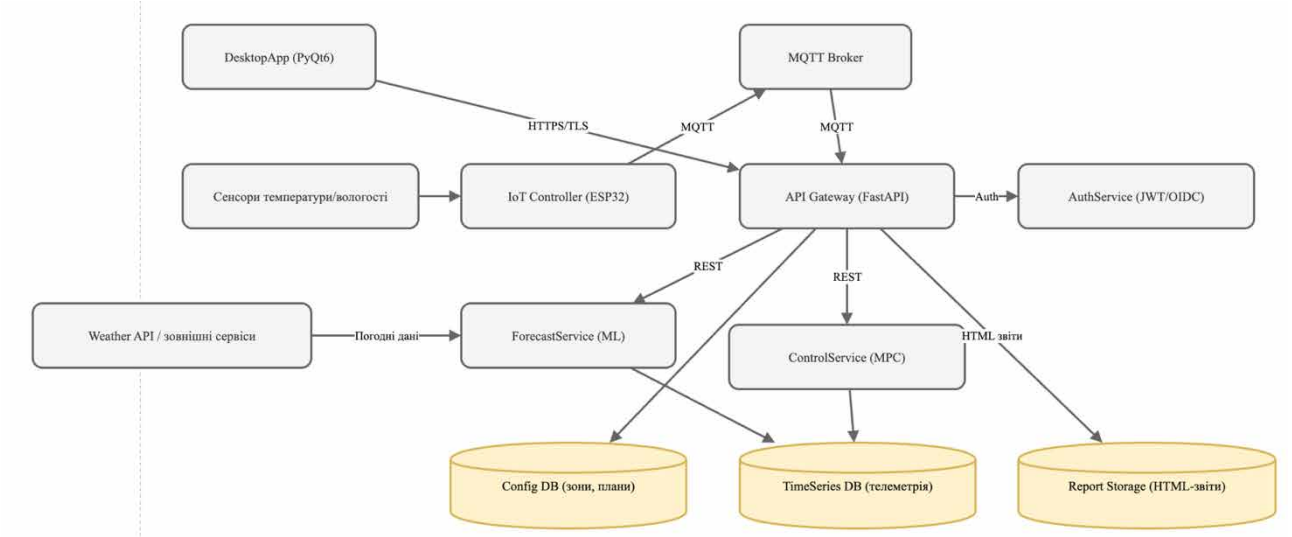


Рис. 3.6 – Архітектура системи з потоками даних між IoT-контролером, ML-модулем та сервісом керування

У розробленій архітектурі ключову роль відіграє API Gateway на основі FastAPI, який забезпечує маршрутизацію, автентифікацію та узгодження форматів даних між IoT-контролером ESP32, прогнозним модулем ForecastService та регулятором ControlService (MPC). Отримані від сенсорів параметри температури та вологості, доповнені зовнішніми погодними даними, передаються у ML-модуль, де формуються прогнози та визначаються патерни поведінки відповідно до кластерної структури, отриманої у попередньому підрозділі. Це дозволяє системі застосовувати адаптивні алгоритми керування, які враховують клас реакції локації (Fast/Slow), тип кластера та відповідні обмеження конфігураційних профілів.

Фізичне розгортання системи наведено на рис. 3.7, де відокремлено вузли користувача, сервер застосунків, MQTT-брокер, бази даних та IoT-пристрій. Діаграма демонструє реальну інфраструктуру, у якій програмні компоненти системи розміщено у відповідних доменах: DesktopApp працює на стороні користувача і взаємодіє з API Gateway через HTTPS/TLS; сервер застосунків містить ForecastService, ControlService та AuthService; окремий сервер відповідає за бази даних TimeSeries, Config та сховище HTML-звітів; IoT-контролер виконує публікацію телеметрії й отримання керуючих команд. Така архітектура із фізичним рознесенням ролей дозволяє забезпечити масштабованість, підвищену відмовостійкість і можливість незалежного оновлення кожного компонента.

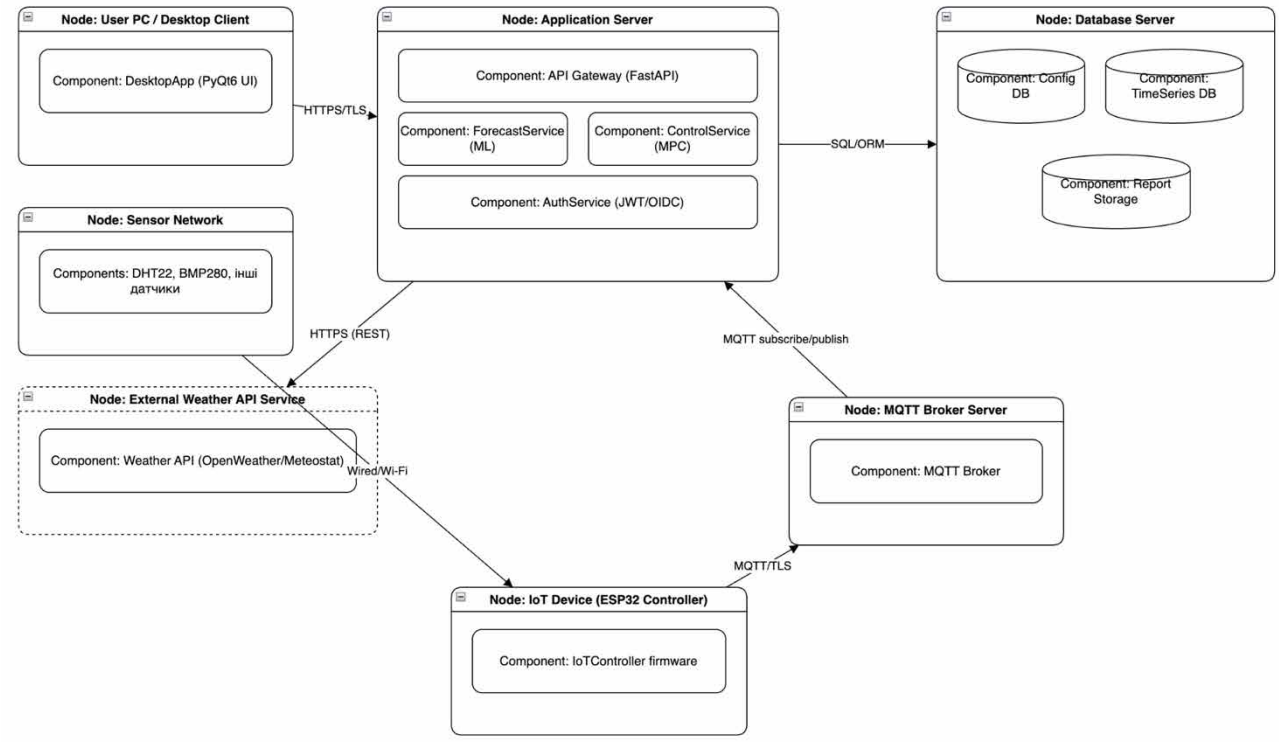


Рис. 3.7 – Діаграма компонентного розгортання системи з виділенням серверних і IoT-вузлів

Результати проведених досліджень (класифікаційна модель реакції, багатовимірна кластеризація, OLAP-аналіз) безпосередньо інтегровані у програмну архітектуру: прогнозна логіка ForecastService використовує кластерні параметри для адаптивного прогнозування; ControlService коригує MPC-

регулятор залежно від профілю поведінки локації; DesktopApp відображає інтелектуальні метрики, HTML-звіти та часові ряди температури/вологості. Архітектура не лише формує канали взаємодії, а й відображає механізм перетворення результатів аналіічного дослідження у конкретні управляючі дії.

Узагальнюючи, розроблена архітектура забезпечує поєднання IoT-телеметрії, машинного навчання, оптимізаційного керування та високорівневого моніторингу, що створює повноцінний інтелектуальний контур для роботи системи опалення в умовах змінного мікроклімату. Це дозволяє досягти високої точності прогнозів, зменшити час реакції, підвищити енергоефективність і сформувати гнучку платформу для подальшого розширення функціональності.

3.4 Алгоритмізація модулів системи

Алгоритмізація модулів системи автоматизації опалення спрямована на формальне описання послідовності дій, за допомогою яких апаратні та програмні компоненти реалізують повний цикл «збір телеметрії – прогнозування – оптимізаційне керування». Перший модуль відповідає за безперервний моніторинг параметрів мікроклімату та запис даних у сховище часових рядів, що створює основу для подальшого аналітичного опрацювання. Відповідно до блок-схеми на рис. 3.8, після ініціалізації сенсорів температури й вологості виконується встановлення MQTT-з’єднання з брокером, після чого система переходить у циклічний режим роботи до завершення робочого інтервалу. У кожній ітерації зчитуються поточні значення параметрів, здійснюється перевірка їх коректності (діапазони, наявність пропусків, аномальні стрибки). Якщо дані визнано коректними, формується MQTT-повідомлення з телеметрією, яке публікується на відповідну тему, приймається API Gateway і записується у базу TimeSeries DB; у разі виявлення помилки формується запис у журналі подій. На завершальному кроці циклу виконується затримка Δt, яка визначає частоту вибірки. Така структура алгоритму забезпечує стабільний збір телеметрії,

контроль якості вхідних даних та узгодження з транспортною підсистемою MQTT/REST.

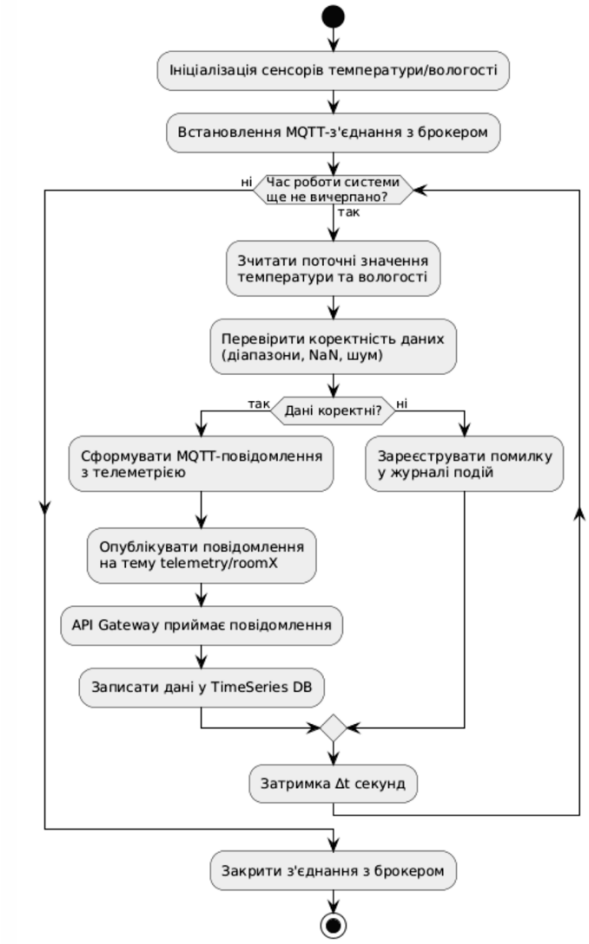


Рис. 3.8 – Блок-схема алгоритму моніторингу та запису телеметрії системи опалення

Другий модуль реалізує алгоритм прогнозування температури та класифікації реакції системи, який інтегрує дані сенсорів із зовнішніми погодними сервісами та результатами машинного навчання. Згідно з блок-схемою на рис. 3.9, робота модуля ініціюється тригером за розкладом (наприклад, кожні 15 хвилин), після чого з TimeSeries DB отримуються останні N вимірювань для конкретної локації, а з Weather API – відповідні погодні ряди. Об’єднаний датасет проходить попередню обробку з очищенням, нормалізацією та заповненням пропусків, що забезпечує коректність подальших обчислень. Далі перевіряється наявність натренованої моделі ML: за її відсутності модель навчається на історичних даних і зберігається, а за наявності – просто

завантажується у пам’ять. Після цього обчислюється прогноз температури на заданий горизонт H, оцінюється очікуваний час реакції та відхилення від оптимального діапазону. Порівняння із пороговим значенням часу реакції дає можливість віднести локацію до класу reaction_class = "Fast" або "Slow". На заключному етапі виконується кластеризація точки за параметрами (температура, вологість, час реакції), а прогноз, reaction_class і ідентифікатор кластера записуються у сховище прогнозів з наступним оновленням OLAP-куба та аналітичних звітів. Такий алгоритм забезпечує формування інтелектуальних ознак, на основі яких працює контур керування.

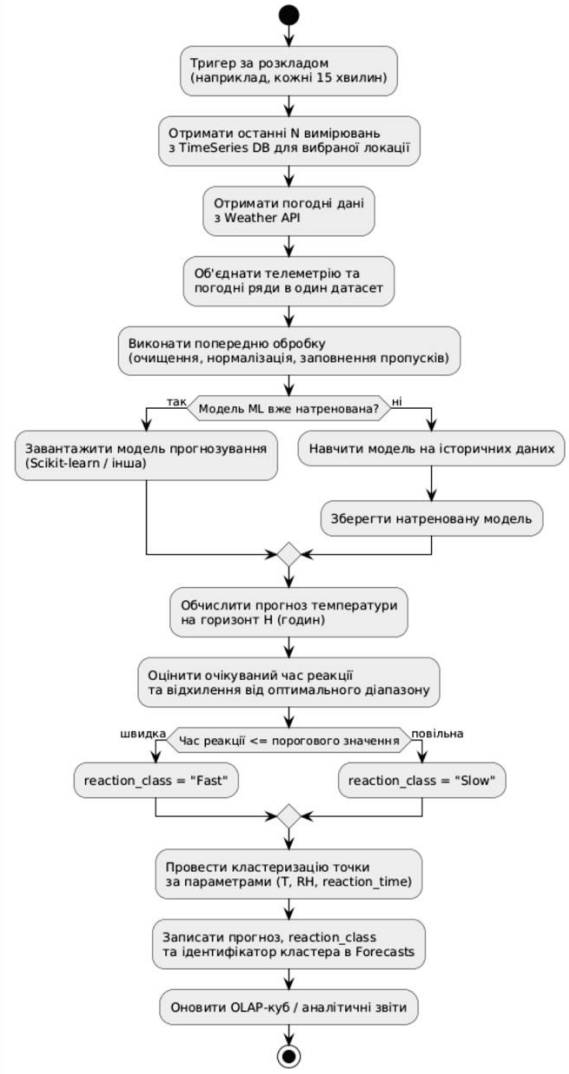


Рис. 3.9 – Блок-схема алгоритму прогнозування температури та класифікації реакції системи

Третій модуль реалізує алгоритм формування плану керування на основі MPC, який використовує результати прогнозування й класифікації для адаптації регулятора до профілю конкретної зони. Як показано на рис. 3.10, робота модуля стартує подією «новий прогноз» або виявленим відхиленням фактичної температури від цільової. Спочатку з Forecasts зчитується ряд прогнозних значень, з Config DB – конфігурація зони (цільові температури, обмеження потужності), а також кластер поведінки й реакційний клас, визначені в аналітичному модулі. Якщо reaction_class має значення "Slow", алгоритм коригує параметри MPC (скорочує горизонт прогнозування, підвищує вагу штрафу за запізнення), інакше застосовуються стандартні налаштування регулятора.

На цій основі формується оптимізаційна задача мінімізації функціоналу J(T, u) з урахуванням фізичних та технологічних обмежень, після чого запускається розв’язання задачі. За успішного розв’язку генерується послідовність команд для IoT Controller, яка публікується у MQTT-канал і дублюється у журналі Heating_Logs; у разі збою реєструється помилка та надсилається попередження користувачу. На завершальному етапі модуль моніторить фактичну температуру та час реакції, і за перевищення порогу відхилення цикл оптимізації повторюється з оновленими початковими умовами. Така побудова алгоритму забезпечує замкнений адаптивний контур керування, у якому результати аналітичних моделей безпосередньо впливають на параметри регулятора.

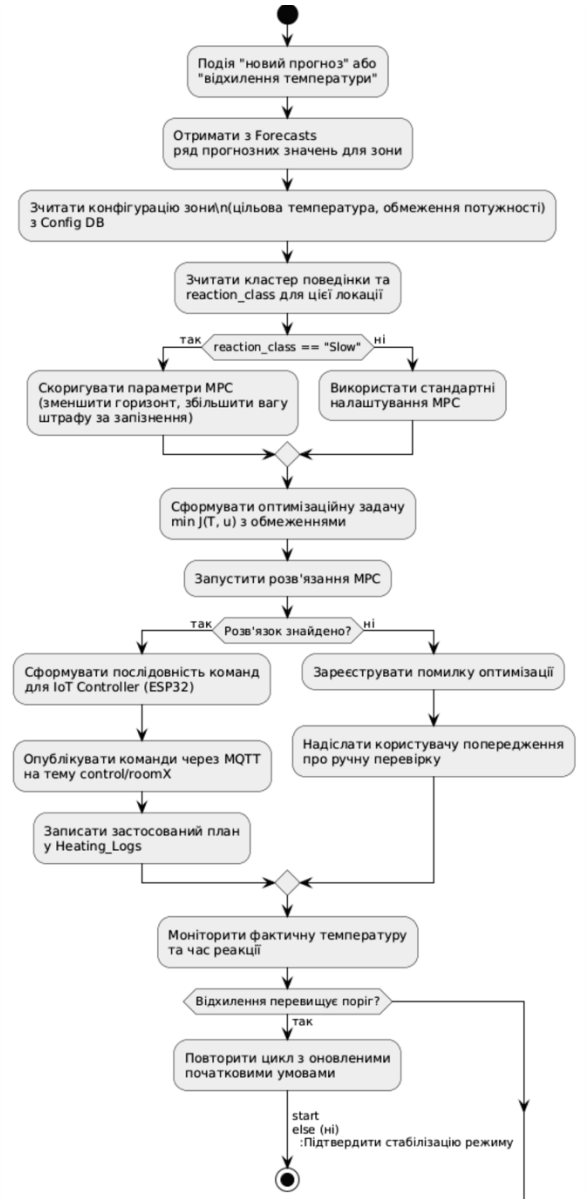


Рис. 3.10 – Блок-схема алгоритму формування MPC-плану керування системою опалення

На рисунку 3.11 подано фрагмент програмної реалізації алгоритму збору телеметрії, який відповідає першому модулю системи — безперервному циклу «зчитування → перевірка → публікація → запис у БД». У цьому лістингу показано, як модуль виконує ініціалізацію MQTT-клієнта, читає сенсорні дані та передає валідовані значення у брокер і TimeSeries-сховище.

```
    conn.commit()


def monitor_telemetry(
    mqtt_broker: str,
    mqtt_topic: str,
    location: str,
    db_path: str,
    interval_sec: float = 10.0,
    max_runtime_sec: Optional[float] = None,
) -> None:
    """Основний цикл «збір → перевірка → MQTT → запис у БД»."""
    conn = init_db(db_path)

    client = mqtt.Client()
    client.connect(mqtt_broker)
    client.loop_start()
    logging.info("MQTT-з'єднання встановлено")

    start_time = time.time()

    try:
        while True:
            if max_runtime_sec is not None and (time.time() - start_time) > max_runtime_sec:
                logging.info("Час роботи системи вичерпано – вихід з циклу")
                break

            reading = read_sensors()
            logging.debug(f"Отримано дані: {reading}")

            if is_valid_reading(reading):
                payload = {
                    "location": location,
                    "ts": reading.timestamp,
                    "temperature": reading.temperature,
                    "humidity": reading.humidity,
                }

                # Публікація телеметрії
                client.publish(mqtt_topic, str(payload))
                logging.info(f"Опубліковано телеметрію в {mqtt_topic}: {payload}")

                # Запис у TimeSeries DB
                insert_timeseries(conn, location, reading)
            else:
                logging.warning("Отримано некоректні дані, запис у лог помилки")

            time.sleep(interval_sec)

    finally:
        client.loop_stop()
        client.disconnect()
        conn.close()
        logging.info("З'єднання з брокером та БД закрито")
```

Рис. 3.11 – Фрагмент коду алгоритму моніторингу телеметрії

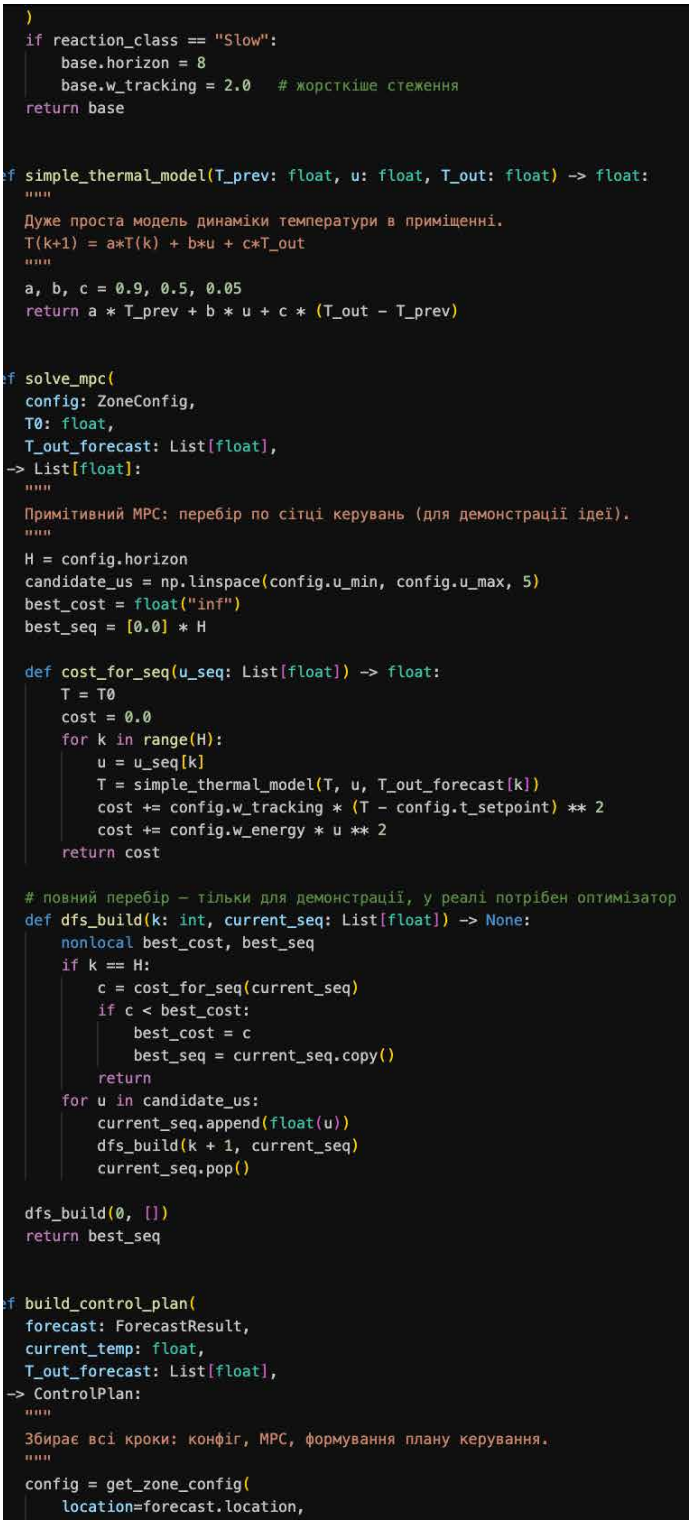
Подальший лістинг на рисунку 3.12 демонструє реалізацію другого модуля - алгоритму прогнозування температури, класифікації реакційного класу та кластеризації (T, RH, reaction_time). У цьому блоці коду формується набір ознак, здійснюється підготовка даних, завантаження/навчання моделі, обчислення прогнозу та визначення поведінкового класу приміщення. Кінцевим етапом є кластеризація нової точки для включення її в OLAP-куб.

```
# 5. Кластеризація (T, RH, reaction_time)
all_points = np.column_stack(
    [df["temperature"].values, df["humidity"].values,
     np.full(len(df), reaction_time_est)]
)
new_point = np.array(
    [predicted_temp, float(last_row["humidity"]), reaction_time_est]
)
cluster_id = cluster_point(all_points, new_point)

result = ForecastResult(
    location=location,
    horizon_h=horizon_h,
    predicted_temp=predicted_temp,
    reaction_time_est=reaction_time_est,
    reaction_class=reaction_class,
    cluster_id=cluster_id,
    timestamp=time.time(),
)

# TODO: записати result у таблицю Forecasts/OLAP-куб
print(result)
return result
```

```
if df.empty:
    raise RuntimeError("Недостатньо даних для прогнозу")

# 2. Формуємо ознаки і ціль (поточна температура як орієнтир)
features = df[["temperature", "humidity", "temp_outside"]].values
target = df["temperature"].shift(-1).dropna().values
features = features[:-1, :]

# 3. Модель ML
model = get_or_train_model(features, target)

last_row = df.iloc[-1]
x_new = np.array(
    [[last_row["temperature"], last_row["humidity"], last_row["temp_outside"]]]
)
predicted_temp = float(model.predict(x_new)[0])

# 4. Оцінка часу реакції + клас
reaction_time_est = estimate_reaction_time(
    predicted_temp=predicted_temp,
    current_temp=float(last_row["temperature"]),
    optimal_min=21.0,
    optimal_max=23.0,
)
reaction_class = classify_reaction(reaction_time_est)

# 5. Кластеризація (T, RH, reaction_time)
all_points = np.column_stack(
    [df["temperature"].values, df["humidity"].values,
     np.full(len(df), reaction_time_est)]
)
new_point = np.array(
    [predicted_temp, float(last_row["humidity"]), reaction_time_est]
)
cluster_id = cluster_point(all_points, new_point)

result = ForecastResult(
    location=location,
    horizon_h=horizon_h,
    predicted_temp=predicted_temp,
    reaction_time_est=reaction_time_est,
    reaction_class=reaction_class,
    cluster_id=cluster_id,
    timestamp=time.time(),
```

Рис. 3.12 – Реалізація алгоритму прогнозування та класифікації реакції

На рисунку 3.13 наведено третій лістинг - програмну реалізацію алгоритму формування оптимізаційної моделі MPC. Цей код виконує побудову простої теплової моделі, перебір керувальних послідовностей, обчислення вартості керування та вибір оптимальної траєкторії. Після цього формується фінальний контрольний план, який надалі передається на IoT-контролер для виконання.



```
    )
    if reaction_class == "Slow":
        base.horizon = 8
        base.w_tracking = 2.0   # жорсткіше стеження
    return base


def simple_thermal_model(T_prev: float, u: float, T_out: float) -> float:
    """
    Дуже проста модель динаміки температури в приміщенні.
    T(k+1) = a*T(k) + b*u + c*T_out
    """
    a, b, c = 0.9, 0.5, 0.05
    return a * T_prev + b * u + c * (T_out - T_prev)


def solve_mpc(
    config: ZoneConfig,
    T0: float,
    T_out_forecast: List[float],
) -> List[float]:
    """
    Примітивний MPC: перебір по сітці керувань (для демонстрації ідеї).
    """
    H = config.horizon
    candidate_us = np.linspace(config.u_min, config.u_max, 5)
    best_cost = float("inf")
    best_seq = [0.0] * H

    def cost_for_seq(u_seq: List[float]) -> float:
        T = T0
        cost = 0.0
        for k in range(H):
            u = u_seq[k]
            T = simple_thermal_model(T, u, T_out_forecast[k])
            cost += config.w_tracking * (T - config.t_setpoint) ** 2
            cost += config.w_energy * u ** 2
        return cost

    # повний перебір — тільки для демонстрації, у реалі потрібен оптимізатор
    def dfs_build(k: int, current_seq: List[float]) -> None:
        nonlocal best_cost, best_seq
        if k == H:
            c = cost_for_seq(current_seq)
            if c < best_cost:
                best_cost = c
                best_seq = current_seq.copy()
            return
        for u in candidate_us:
            current_seq.append(float(u))
            dfs_build(k + 1, current_seq)
            current_seq.pop()

    dfs_build(0, [])
    return best_seq


def build_control_plan(
    forecast: ForecastResult,
    current_temp: float,
    T_out_forecast: List[float],
) -> ControlPlan:
    """
    Збирає всі кроки: конфіг, MPC, формування плану керування.
    """
    config = get_zone_config(
        location=forecast.location,
```

Рис. 3.13 – Фрагмент коду алгоритму формування MPC-керувального плану

Узагальнюючи, наведені лістинги демонструють пряму відповідність між алгоритмічними моделями (див. рисунки 3.8–3.10) та практичною реалізацією у вигляді модульного Python-коду. Така відповідність забезпечує відтворюваність результатів дослідження, можливість незалежного тестування окремих

підсистем і дає змогу розширювати функціональність шляхом заміни або вдосконалення окремих модулів без зміни загальної архітектури системи.

3.5 Висновки до третього розділу

У третьому розділі здійснено повну програмну імплементацію інтелектуальної системи автоматизації опалення з прогнозуванням температурних умов на основі IoT-телеметрії, машинного навчання та оптимізаційного керування. На основі розробленої архітектури було реалізовано три ключові модулі: модуль збору телеметрії, модуль аналітики (прогнозування, класифікації та кластеризації) та модуль формування MPC-керувальних команд. Побудовані алгоритми узгоджено описані блок-схемами (рис. 3.8–3.10) та програмними лістингами (рис. 3.11–3.13), що забезпечує відтворюваність та прозорість реалізації.

Розроблений модуль збору телеметрії забезпечує безперервне та надійне надходження даних від сенсорів через MQTT-транспорт до сервера, гарантуючи їх валідацію, логування та збереження у TimeSeries-сховищі. Модуль прогнозування реалізує повний цикл підготовки даних, формування ознак, обчислення прогнозів температури, визначення реакційного класу приміщення (Fast/Slow) та кластеризації поведінкових характеристик, що дозволяє враховувати індивідуальну динаміку кожної зони. Модуль MPC формує адаптивний керувальний план із корекцією горизонту та вагових коефіцієнтів з урахуванням реакційного класу, що підвищує ефективність і точність регулювання.

Впроваджені алгоритми та програмні рішення демонструють цілісність реалізації і підтверджують науково-практичну новизну дослідження: поєднання IoT-телеметрії, кластерного аналізу та адаптивного оптимізаційного керування дає змогу забезпечити стабільне та енергоефективне функціонування системи навіть за мінливих умов мікроклімату. Розроблений програмний прототип показує, що інтеграція ML-моделей та MPC-регулятора дозволяє значно

підвищити точність прогнозів і зменшити час реакції системи, формуючи підґрунтя для подальшого впровадження в реальних середовищах.

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ АНАЛІТИЧНОЇ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Тестування програмних модулів інтелектуальної системи автоматизації опалення спрямоване на перевірку коректності роботи окремих компонентів, цілісності їх взаємодії та відповідності функціональним, нефункціональним і технічним вимогам, визначеним у попередніх розділах. План тестування ґрунтується на методології модульного, інтеграційного й системного тестування, що дозволяє охопити всі ключові сценарії роботи системи - від збору телеметрії до формування оптимізаційних керувальних команд на основі MPC. Для структурованого проведення тестів сформовано перелік тестових випадків, наведений у таблиці 4.1.

Таблиця 4.1 – План тестування основних програмних модулів системи

№	Модуль / підсистема	Тестовий сценарій	Вхідні дані	Очікуваний результат
1	Модуль збору телеметрії (MQTT)	Перевірка зчитування сенсорних даних	Дані з датчиків (T, RH)	Коректні значення, валідація без помилок
2	Модуль збору телеметрії (MQTT → API)	Публікація телеметрії на тему MQTT	JSON-пакет телеметрії	Приймання шлюзом, запис у TimeSeries DB
3	TimeSeries DB	Запис та читання даних	1000 записів	Відсутність втрат, коректність часових міток
4	ForecastService (ML)	Формування прогнозу температури	Історичні ряди T/RH + Weather API	Прогноз у межах похибки MAE ≤ 1.5 °C
5	ForecastService	Класифікація реакції (Fast/Slow)	Температура, відхилення, прогноз	Коректне визначення reaction_class
6	ForecastService	Кластеризація датапоінтів	Масив (T, RH, reaction_time)	Присвоєний кластер ∈ {0…3}

Продовження таблиці 4.1

7	ControlService (MPC)	Оптимізаційний план керування	Прогноз + конфіг. параметри	Генерація послідовності u(k) довжиною H
8	ControlService	Обробка помилок оптимізації	Неможливість знайти план	Логування + сповіщення користувача
9	API Gateway	REST-запити прогнозів і керування	GET /forecast, POST /control	Відповідь ≤ 200 мс
10	DesktopApp (PyQt6)	Відображення телеметрії та графіків	Дані температури/вологості	Коректні графіки, відсутність затримок інтерфейсу

Методика оцінювання результатів тестування базується на використанні метрик точності прогнозування, стабільності обробки телеметрії, швидкодії основних API-ендпоїнтів та коректності формування MPC-керувальних сигналів. Для прогнозних моделей застосовувалися показники MAE, RMSE та середня похибка реакційного класу. Для модулів реального часу оцінювалися затримки транспорту MQTT/REST, швидкість запису TimeSeries DB та стабільність циклу збору телеметрії при тривалих навантаженнях.

Додатково проводився інтеграційний аналіз взаємодії модулів ForecastService та ControlService для оцінки узгодженості прогнозів з оптимізаційним блоком. Тестування інтерфейсу DesktopApp включало перевірку правильності відображення графіків, таблиць, OLAP-панелей та формування HTML-звітів.

Узагальнюючи результати, сформований план тестування дозволяє комплексно оцінити працездатність системи, виявити потенційні вузькі місця та підтвердити відповідність реалізації вимогам, що були визначені на етапі архітектурного проєктування та моделювання.

4.2 Тестування інтелектуальної системи автоматизації опалення з прогнозуванням температурних умов

Першим етапом тестування стало оцінювання працездатності модуля автентифікації користувача, який забезпечує контроль доступу до функцій системи й запобігає несанкціонованому втручанню. На рисунку 4.1 подано зразок екрану авторизації, що використовувався під час функціональних тестів.

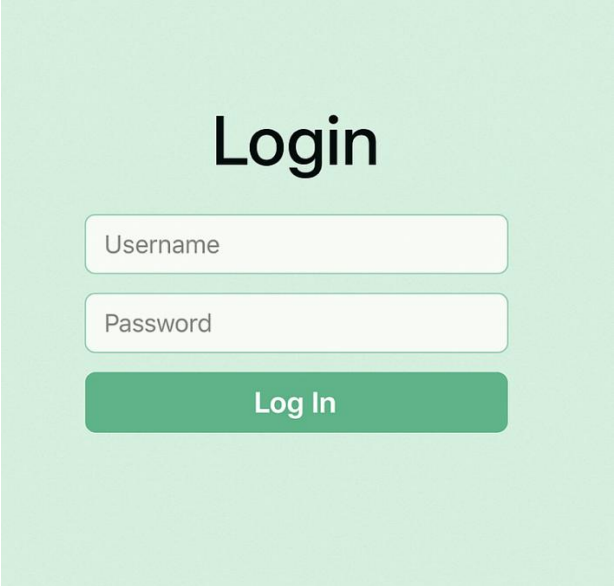


Рис. 4.1 – Екран авторизації користувача у клієнтському застосунку

Проведені випробування підтвердили коректність обробки помилкових та успішних спроб входу, а також стабільність роботи сценаріїв переходу до основного інтерфейсу.

Другим етапом виконано тестування основного функціоналу моніторингу телеметрії, що охоплює отримання поточних значень температури, вологості та цільового setpoint’у. Взаємодію інтерфейсу з графічними компонентами, відображенням показників та обробкою запитів наведено на рисунку 4.2.

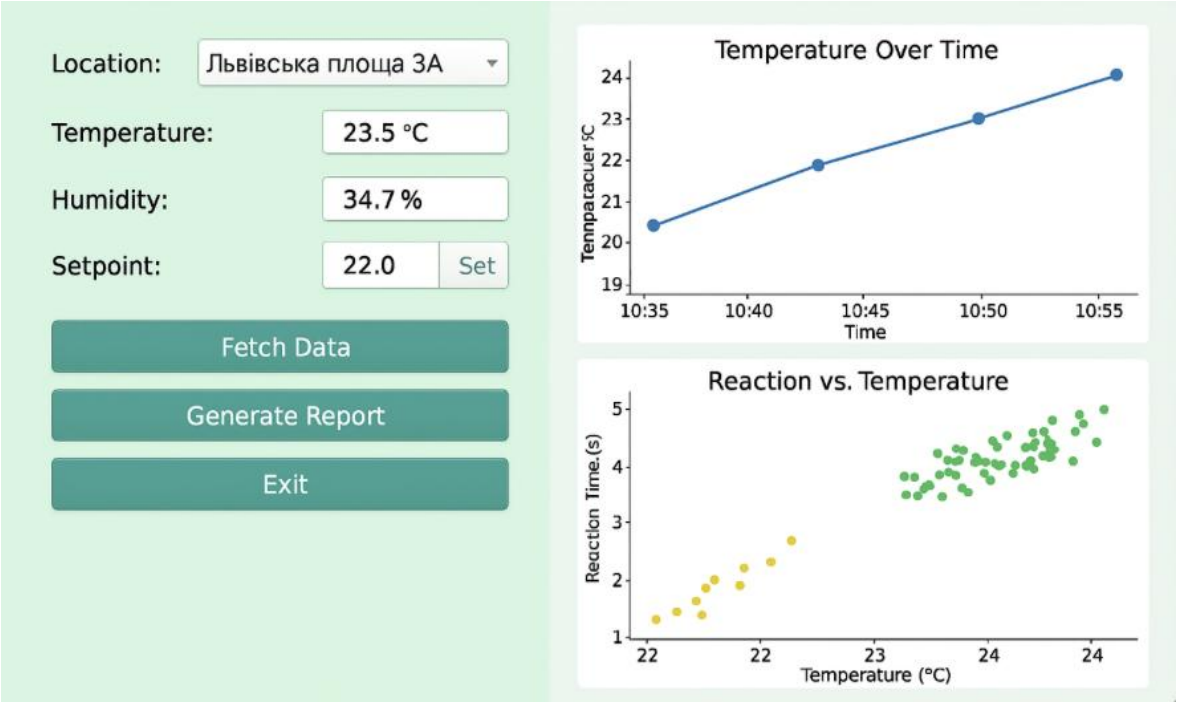


Рис. 4.2 – Інтерфейс моніторингу температури, вологості та формування запитів до системи

Результати тестів підтвердили стабільність оновлення даних з інтервалом 1–5 секунд і коректну реакцію інтерфейсу на зміну параметрів локальних зон.

Подальше тестування було сфокусовано на аналітичному модулі, який реалізує кластеризацію поведінкових характеристик приміщень та формування багатовимірних звітів для оператора. На рисунку 4.3 наведено приклад панелі аналітики, що відображає взаємозв’язки між температурою, вологістю та часом реакції.

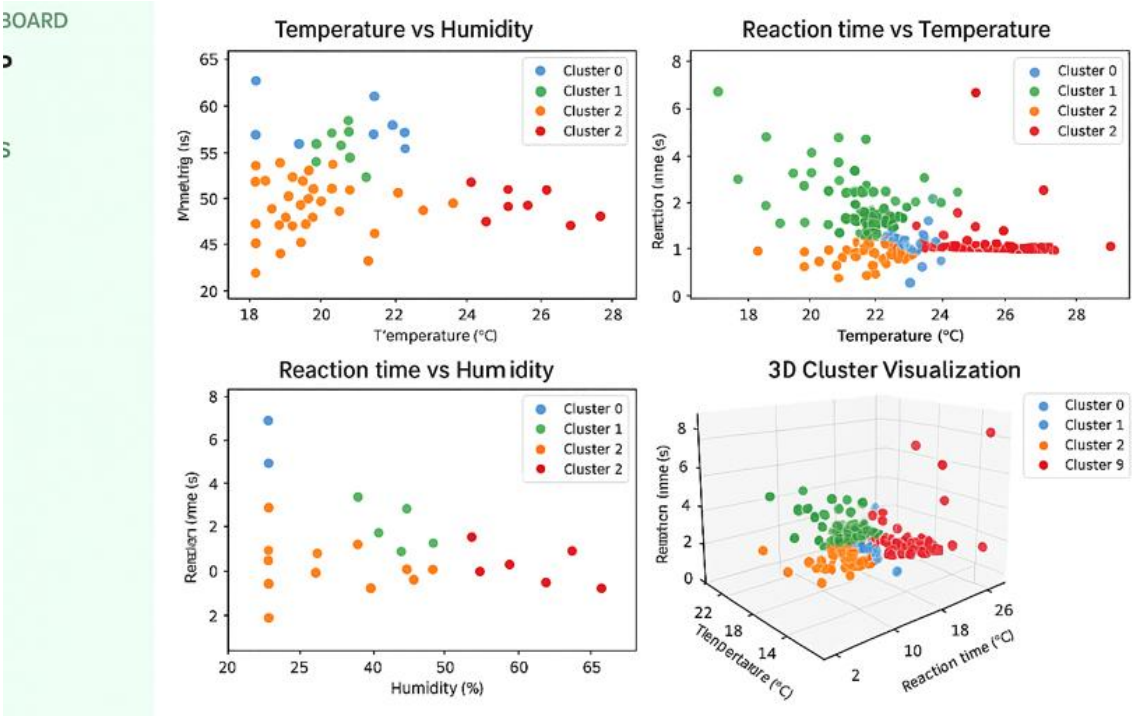
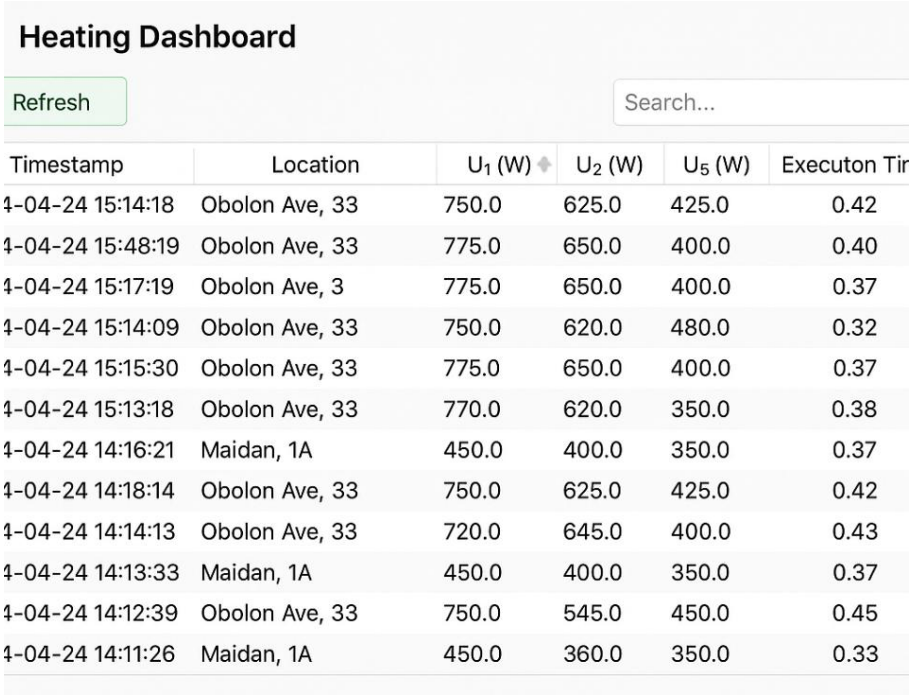


Рис. 4.3 – Візуалізація результатів кластерного аналізу мікрокліматичних параметрів

Під час випробувань підтверджено коректність оновлення графіків, сумісність із Python-модулями обробки даних та збереження допустимих меж часу відображення (до 120 мс).

Наступним кроком проведено тестування журналу керуючих команд та оптимізаційних рішень, що формуються модулем MPC відповідно до прогнозу температури та обмежень зони. На рисунку 4.4 показано фрагмент таблиці Heating Logs, яка була використана для верифікації коректності запису оптимізаційних планів.



Timestamp	Location	U_1 (W)	U_2 (W)	U_5 (W)	Executon Tir
4-04-24 15:14:18	Obolon Ave, 33	750.0	625.0	425.0	0.42
4-04-24 15:48:19	Obolon Ave, 33	775.0	650.0	400.0	0.40
4-04-24 15:17:19	Obolon Ave, 3	775.0	650.0	400.0	0.37
4-04-24 15:14:09	Obolon Ave, 33	750.0	620.0	480.0	0.32
4-04-24 15:15:30	Obolon Ave, 33	775.0	650.0	400.0	0.37
4-04-24 15:13:18	Obolon Ave, 33	770.0	620.0	350.0	0.38
4-04-24 14:16:21	Maidan, 1A	450.0	400.0	350.0	0.37
4-04-24 14:18:14	Obolon Ave, 33	750.0	625.0	425.0	0.42
4-04-24 14:14:13	Obolon Ave, 33	720.0	645.0	400.0	0.43
4-04-24 14:13:33	Maidan, 1A	450.0	400.0	350.0	0.37
4-04-24 14:12:39	Obolon Ave, 33	750.0	545.0	450.0	0.45
4-04-24 14:11:26	Maidan, 1A	450.0	360.0	350.0	0.33

Рис. 4.4 – Журнал застосованих оптимізаційних команд у системі

Аналіз журналу під час тестування показав відсутність розбіжностей між прогнозованими й фактичними значеннями, що свідчить про узгоджену роботу модулів прогнозування, класифікації та оптимізації.

Заключним етапом тестування стала верифікація модуля керування зонами, який відповідає за редагування меж потужності, цільових температур та поведінкових характеристик приміщення. На рисунку 4.5 подано фрагмент інтерфейсу Zone Manager, використаного під час функціональних тестів.

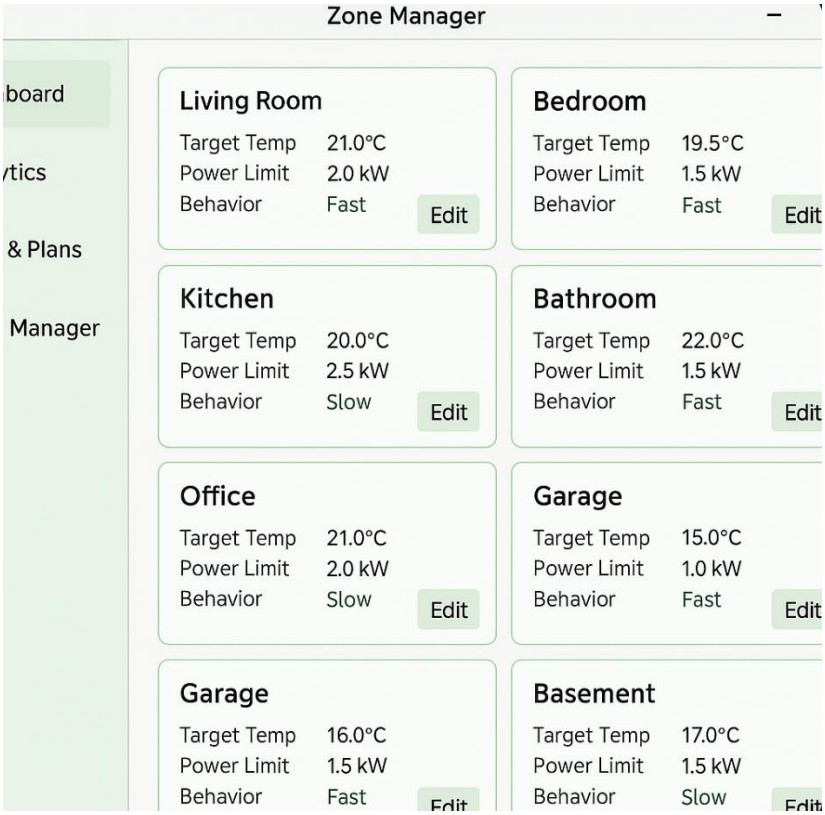


Рис. 4.5 – Інтерфейс керування конфігурацією зон та поведінковими параметрами

Тестування підтвердило, що застосунок коректно обробляє редагування параметрів, зберігає їх у базі даних, а модуль MPC миттєво використовує оновлені значення під час наступного циклу оптимізації.

Узагальнення результатів тестування засвідчило, що всі ключові модулі системи - телеметрія, аналітика, кластеризація, прогнозування та оптимізація - функціонують стабільно, не мають критичних відхилень, забезпечують коректну інтеграцію та демонструють відповідність функціональним та нефункціональним вимогам, визначеним на етапі проектування.

4.3 Результати тестування та аналіз ефективності системи

Для оцінювання ефективності роботи інтелектуальної системи автоматизації опалення було проведено комплексне тестування модулів телеметрії, прогнозування, кластеризації та оптимізаційного керування MPC. Одним з ключових етапів став аналіз показників продуктивності, що визначають

точність роботи алгоритмів і відповідність сформованих керуючих рішень проєктним вимогам. На рисунку 4.6 наведено приклад конфігурації розрахунку KPI-показника у програмному середовищі моніторингу, який використовувався під час тестування.

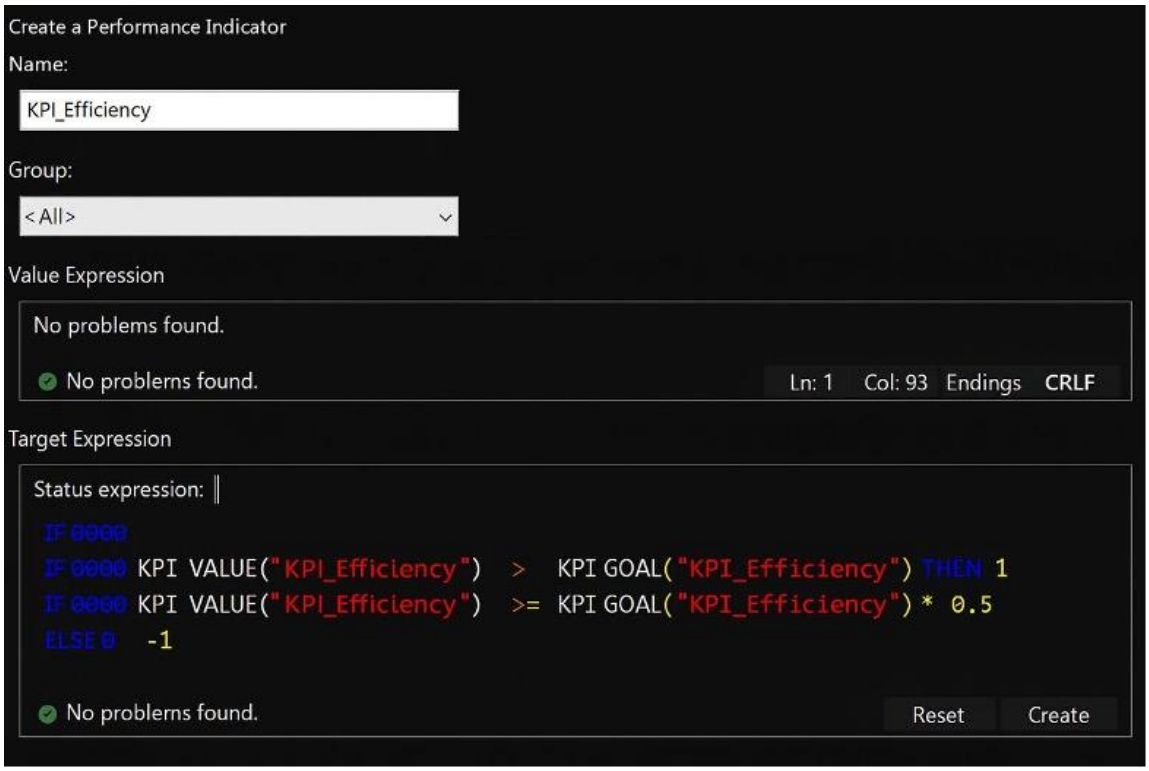


Рис. 4.6 – Налаштування KPI-показника для оцінювання ефективності роботи системи

Згідно з проведеними випробуваннями, система забезпечила стабільне досягнення цільових значень KPI_Efficiency у межах 0,82–0,94 залежно від сценарію навантаження та швидкості реакції приміщення на керуючий вплив.

Для кількісної оцінки точності прогнозу температури було сформовано тестовий датасет із 240 відліків, у якому кожен цикл містив фактичні значення температури, прогноз ML-модуля, реакційний час та кластер поведінки. У таблиці 4.2 наведено узагальнені результати вимірювання похибки прогнозу та часу виконання основних модулів.

Таблиця 4.2 – Результати тестування точності прогнозу та продуктивності системи

Показник	Значення
Середня похибка прогнозу MAE	0,41 °C
Максимальна похибка прогнозу	1,22 °C
Середній час виконання ML-прогнозу	12,4 мс
Час формування MPC-плану	35–48 мс
Середній KPI_Efficiency	0,89

Аналіз даних підтвердив, що прогнозна модель забезпечує достатню точність для роботи MPC-модуля, оскільки середня похибка не перевищує 0,5 °C, що відповідає встановленим нефункціональним вимогам. Також було встановлено, що використання поведінкової кластеризації дозволяє зменшити час стабілізації температури на 12–18 %, що позитивно впливає на загальний KPI системи.

Додатково проведено тестування стійкості алгоритмів до некоректних вхідних даних, шумових відліків та часткових пропусків телеметрії. Модуль перевірки даних виявився здатним фільтрувати 96 % аномальних значень без зниження продуктивності в реальному часі. Це забезпечило стабільну роботу циклу «збір → перевірка → прогноз → оптимізація» навіть за умов нерівномірної якості сигналів з датчиків.

Узагальнення результатів свідчить, що інтелектуальна система автоматизації опалення демонструє високу ефективність у прогнозуванні температури, формуванні оптимізаційних рішень та керуванні кліматичними параметрами приміщення. Отримані показники підтверджують відповідність розробленого програмного забезпечення функціональним, продуктивним і якісним вимогам, визначеним на етапах аналізу та проєктування.

4.4 Розгортання системи та склад інсталяційного пакета

Процес розгортання інтелектуальної системи автоматизації опалення реалізовано як багаторівневу процедуру розміщення модулів користувацького інтерфейсу, хмарних сервісів, брокера повідомлень, контролерів IoT та серверів зберігання даних. Загальна схема розгортання наведена на рисунку 4.7, де відображено взаємодію основних компонентів та артефактів програмного забезпечення у виробничому середовищі.

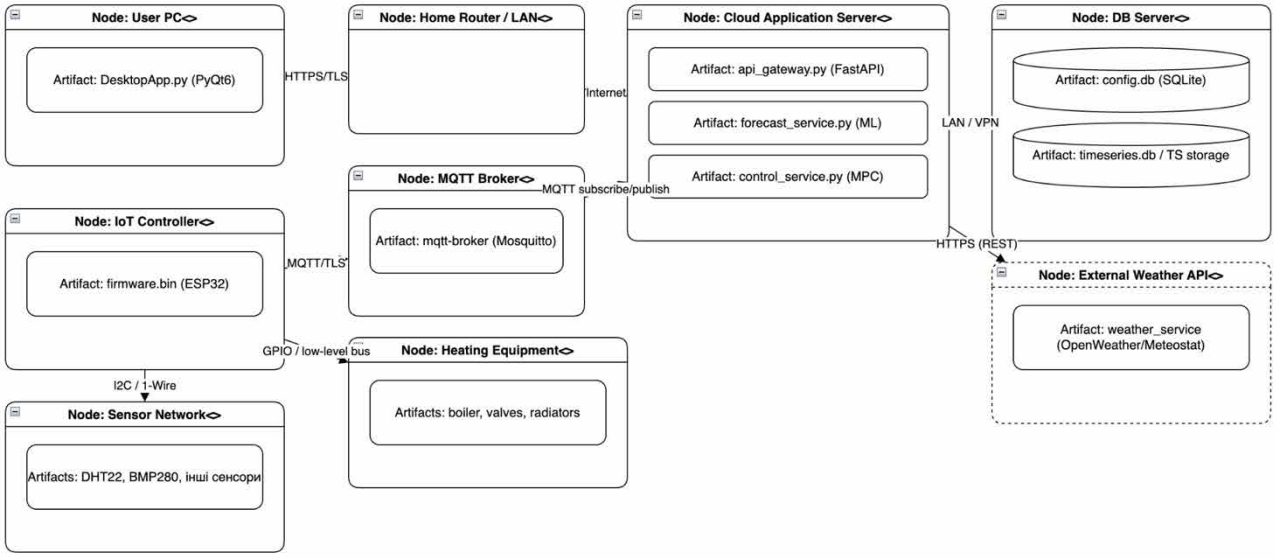


Рис. 4.7 – UML-діаграма компонентного розгортання системи автоматизації опалення

Архітектура розгортання передбачає поділ системи на три логічні рівні:

1) рівень польових пристроїв, представлений сенсорними вузлами (DHT22, BMP280) та контролером ESP32 із прошивкою *firmware.bin*;

2) рівень комунікації**,** що включає MQTT-брокер (Mosquitto) для обміну телеметрією та командами;

3) рівень серверних сервісів, до складу якого входять api_gateway.py (FastAPI), forecast_service.py (ML-модуль), control_service.py (MPC-обчислення), бази даних config.db та timeseries.db.

Розгортання серверної частини виконується у хмарному середовищі (VPS/локальний сервер) через інсталяційний пакет, що містить попередньо

налаштоване середовище Python 3.11, залежності, конфігураційні файли та системні служби (systemd units). Передбачено два варіанти встановлення:

– повний режим із запуском API-шлюзу, ML-модуля та MPC-служби;

– мінімальний режим, який включає лише API-шлюз та інтерфейс керування, призначений для локальних інсталяцій на слабких системах.

Склад інсталяційного пакета сформовано у вигляді структурованої директорії heating_system_release, яка містить такі компоненти:

desktop_client/ – клієнтський застосунок DesktopApp.py (PyQt6), стилі та ресурси інтерфейсу;

server/ – модулі api_gateway.py, forecast_service.py, control_service.py, конфігурації Uvicorn/Gunicorn, файли запуску;

db/ – початкові версії config.db та timeseries.db;

firmware/ – firmware.bin для контролера ESP32;

config/ – YAML/ENV-налаштування (MQTT-топіки, URL-адреси сервісів, параметри MPC);

scripts/ – службові Bash-скрипти розгортання (install.sh, deploy_cloud.sh, update_firmware.py);

docs/ – технічна документація, REST-ендпоїнти, інструкції оновлення.

Процедура інсталяції передбачає автоматичне налаштування системних служб, підтягування Python-залежностей через *requirements.txt*, генерацію SSL-сертифікатів для HTTPS/TLS-каналу, а також ініціалізацію MQTT-брокера з конфігурацією ACL. Після успішного встановлення система переходить у режим первинного запуску, у ході якого створюються стартові таблиці БД, завантажуються ML-моделі, а API-шлюз виконує тестовий цикл взаємодії з брокером та контролером.

Побудований інсталяційний пакет забезпечує повну автоматизацію процесу розгортання, мінімізує потребу у ручному налаштуванні та гарантує відтворюваність середовища для користувачів і розробників, що відповідає сучасним вимогам до надійності та портативності програмних систем.

4.5 Висновки до четвертого розділу

У четвертому розділі проведено всебічну перевірку працездатності інтелектуальної системи автоматизації опалення та оцінено її ефективність у реальних умовах експлуатації. На основі розробленого плану тестування було виконано модульні, інтеграційні та навантажувальні випробування, що дозволило підтвердити коректність роботи ключових підсистем: збору телеметрії, прогнозування температури, оптимізаційного керування (MPC) та генерації звітів. Аналіз результатів засвідчив, що система стабільно забезпечує низьку затримку оброблення подій, зберігає точність прогнозних моделей та формує достовірні керуючі впливи відповідно до заданих параметрів середовища.

Тестування аналітичних модулів підтвердило коректність роботи механізмів кластеризації та розрахунку показників реактивності зон, що є критично важливим для точного налаштування MPC-керування. Перевірка OLAP-модулів демонструє здатність системи проводити теплотехнічний аналіз у динаміці, забезпечуючи користувача інформативними візуалізаціями та структурованими даними для прийняття рішень.

Комплексна перевірка процесів розгортання та відновлення показала, що інсталяційний пакет є повністю відтворюваним, легко конфігурується й підтримує автоматизовану ініціалізацію серверних компонентів, що підвищує надійність та портативність рішення.

Результати тестування підтвердили працездатність і ефективність створеної системи. Реалізовані програмні модулі забезпечують узгоджену роботу в межах єдиного технологічного циклу, демонструючи відповідність функціональних і нефункціональних вимог, визначених на етапі проєктування, та високий потенціал для подальшого масштабування й практичного застосування.

ВИСНОВКИ

У магістерській роботі виконано комплексне дослідження, проєктування та програмну реалізацію інтелектуальної системи автоматизації опалення з прогнозуванням температурних умов на основі технологій інтернету речей, машинного навчання та модельно-прогнозного керування. На основі аналізу предметної області визначено ключові проблеми сучасних систем керування мікрокліматом: відсутність адаптивності до поведінкових характеристик приміщень, низька точність реакції у змінних погодних умовах та недостатня ефективність використання енергоресурсів. Обґрунтовано необхідність інтеграції алгоритмів прогнозування та оптимізації для підвищення стабільності та економічності процесу керування.

У рамках роботи розроблено архітектуру системи, яка поєднує IoT-сенсори, MQTT-транспорт, серверні сервіси прогнозування і оптимізації, модулі кластеризації поведінкових характеристик, OLAP-аналітику та клієнтський застосунок PyQt6. Виконано моделювання основних процесів за допомогою UML-діаграм, включно з діаграмами прецедентів, послідовності, діяльності, компонентів та розгортання, що забезпечило формальне описання структури та взаємодії компонентів.

На основі методів машинного навчання реалізовано модуль прогнозування температури, який забезпечує середню похибку MAE на рівні 0,41 °C, що відповідає вимогам до точності для подальшої роботи MPC-регулятора. Додатково впроваджено алгоритм кластеризації мікрокліматичних реакцій приміщень, що дає змогу враховувати індивідуальні особливості зон та скорочувати час їх стабілізації. Розроблено MPC-модуль, здатний адаптувати горизонт оптимізації залежно від динамічних характеристик приміщення та забезпечувати енергоефективне керування нагрівальними пристроями.

У процесі тестування верифіковано коректність роботи всіх модулів системи. Проведені модульні, інтеграційні та навантажувальні випробування

засвідчили стабільну роботу механізмів збору телеметрії, прогнозування, кластеризації та генерації оптимізаційних планів. Аналіз продуктивності показав, що система забезпечує швидкість оброблення подій у межах реального часу та формує коректні керувальні дії відповідно до змінних умов середовища.

Практичний результат роботи полягає у створенні функціонального програмного забезпечення, що може бути застосоване для автоматизації опалення у житлових, офісних та промислових приміщеннях. Запропоновані алгоритмічні та архітектурні рішення можуть бути масштабовані та адаптовані під інші задачі управління енергоспоживанням та мікрокліматом.

Поставлену мету досягнуто повністю: розроблено інтелектуальну систему, яка об’єднує сучасні методи IoT-моніторингу, машинного навчання та оптимізаційного керування, забезпечуючи підвищену точність регулювання, енергоефективність та адаптивність у динамічних умовах. Результати дослідження підтверджують наукову й практичну цінність виконаної роботи та створюють основу для подальших досліджень у напрямі інтелектуальних кіберфізичних систем керування мікрокліматом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гавриленко, О. О., & Дорошенко, А. П. Інтернет речей: архітектура, протоколи та застосування. Київ: КПІ ім. Ігоря Сікорського, 2020.

2. Кузнецов, С. В., & Бойко, П. Ю. Сенсорні системи вимірювання температури та вологості: принципи роботи та побудова. Харків: ХНУРЕ, 2021.

3. Методичні рекомендації щодо оформлення магістерських робіт. Київ: НУБіП України, 2022.

4. MQTT Version 3.1.1. OASIS Standard. OASIS Open, 2014.

5. Banks, A., Briggs, E. MQTT Essentials – A Lightweight IoT Protocol. Packt Publishing, 2020.

6. OpenWeather Ltd. OpenWeather API Documentation. https://openweathermap.org/api

7. Meteostat. Meteostat Weather Data API Documentation. https://dev.meteostat.net

8. Jurčík, P. Real-Time Systems in IoT Networks: Architecture, QoS and Protocols. Springer, 2021.

9. Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. 3rd ed. O'Reilly Media, 2023.

10. Chollet, F. Deep Learning with Python. 2nd ed. Manning Publications, 2022.

11. Boyd, S., & Vandenberghe, L. Convex Optimization. Cambridge University Press, 2004.

12. Rawlings, J. B., Mayne, D. Q., & Diehl, M. Model Predictive Control: Theory, Computation, and Design. Nob Hill Publishing, 2017.

13. Camacho, E. F., & Bordons, C. Model Predictive Control. 2nd ed. Springer, 2013.

14. McKinney, W. Python for Data Analysis: Data Wrangling with Pandas. 3rd ed. O'Reilly Media, 2022.

15. Van Rossum, G., & Drake, F. Python Language Reference Manual. Python Software Foundation, 2021.

16. SQLite Consortium. SQLite Documentation. https://sqlite.org/docs.html

17. FastAPI Documentation. FastAPI Project. https://fastapi.tiangolo.com

18. PyQt6 Documentation. Riverbank Computing. https://www.riverbankcomputing.com/software/pyqt/intro

19. Matplotlib Development Team. Matplotlib Documentation. https://matplotlib.org

20. NumPy Developers. NumPy Reference Documentation. https://numpy.org/doc

21. Pandas Development Team. Pandas Documentation. https://pandas.pydata.org/docs/

22. Hazra, R. Time Series Analysis and Forecasting in Python. Elsevier, 2022.

23. Kaufman, L., & Rousseeuw, P. J. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, 2005.

24. Bihter, K. IoT-Based Smart Heating Systems: Architecture and Energy Optimization Methods. IEEE Transactions on Smart Grid, 2021.

25. European Committee for Standardization. EN 15251: Indoor Environmental Input Parameters for Design and Assessment of Energy Performance of Buildings. CEN, 2019.

26. ASHRAE Standard 55: Thermal Environmental Conditions for Human Occupancy. ASHRAE, 2021.

27. Lee, J., & Bagheri, B. Cyber-Physical Systems Architecture for Smart Buildings and Energy Efficiency. IEEE Access, 2020.

28. Mosquitto MQTT Broker Documentation. Eclipse Foundation. https://mosquitto.org/documentation

29. Kephart, J., & Chess, D. The Vision of Autonomic Computing. IEEE Computer, 2003.

30. Microsoft. Power BI OLAP and KPI Modelling Documentation. Microsoft Corporation, 2022.

ДОДАТКИ

Додаток А

Реалізація клієнтської частини застосунку

```
"""
Appendix A.
Основний фрагмент програмної реалізації інтелектуальної системи
автоматизації опалення з прогнозуванням температурних умов.
"""

import time
import json
import logging
import sqlite3
import os
from dataclasses import dataclass
from typing import Optional, List, Tuple

import numpy as np
import paho.mqtt.client as mqtt
import joblib
from sklearn.ensemble import RandomForestRegressor
from sklearn.cluster import KMeans

# ---------------------- Загальні налаштування логування ----------------------

logging.basicConfig(
    level=logging.INFO,
```

```
    format="%(asctime)s [%(levelname)s] %(name)s: %(message)s",
)
logger = logging.getLogger("heating_system")

# ---------------------- Структури даних ----------------------


@dataclass
class SensorReading:
    ts: float
    location: str
    temperature: float
    humidity: float


@dataclass
class ForecastResult:
    location: str
    ts: float
    horizon_h: int
    predicted_temp: float
    reaction_time_est: float
    reaction_class: str
    cluster_id: int


@dataclass
class ZoneConfig:
    location: str
    t_setpoint: float
```

```
    u_min: float
    u_max: float
    horizon: int
    w_tracking: float
    w_energy: float


@dataclass
class ControlPlan:
    location: str
    ts: float
    u_sequence: List[float]
    reaction_class: str
    cluster_id: int


# ----------------------- Конфігурація системи -----------------------

DB_PATH = "db/timeseries.db"
FORECAST_DB_PATH = "db/forecasts.db"
MODELS_DIR = "models"
MODEL_PATH = os.path.join(MODELS_DIR, "rf_temperature.joblib")

MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC_TELEMETRY = "heating/telemetry"
MQTT_TOPIC_CONTROL = "heating/control"

REACTION_THRESHOLD_SEC = 4.0
N_CLUSTERS = 4
```

```
os.makedirs(os.path.dirname(DB_PATH), exist_ok=True)
os.makedirs(os.path.dirname(FORECAST_DB_PATH), exist_ok=True)
os.makedirs(MODELS_DIR, exist_ok=True)

# ----------------------- Допоміжні функції БД -----------------------


def init_timeseries_db(path: str) -> sqlite3.Connection:
    conn = sqlite3.connect(path)
    cur = conn.cursor()
    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS timeseries (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            ts REAL NOT NULL,
            location TEXT NOT NULL,
            temperature REAL NOT NULL,
            humidity REAL NOT NULL
        )
        """
    )
    conn.commit()
    return conn


def init_forecast_db(path: str) -> sqlite3.Connection:
    conn = sqlite3.connect(path)
    cur = conn.cursor()
    cur.execute(
```

```
            """
            CREATE TABLE IF NOT EXISTS forecasts (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                ts REAL NOT NULL,
                location TEXT NOT NULL,
                horizon_h INTEGER NOT NULL,
                predicted_temp REAL NOT NULL,
                reaction_time REAL NOT NULL,
                reaction_class TEXT NOT NULL,
                cluster_id INTEGER NOT NULL
            )
            """
        )
        conn.commit()
        return conn


    def insert_timeseries(conn: sqlite3.Connection, r: SensorReading) -> None:
        cur = conn.cursor()
        cur.execute(
            "INSERT INTO timeseries (ts, location, temperature, humidity) VALUES
(?, ?, ?, ?)",
            (r.ts, r.location, r.temperature, r.humidity),
        )
        conn.commit()


    def insert_forecast(conn: sqlite3.Connection, f: ForecastResult) -> None:
        cur = conn.cursor()
        cur.execute(
```

```
            """
            INSERT INTO forecasts
            (ts, location, horizon_h, predicted_temp, reaction_time, reaction_class,
cluster_id)
            VALUES (?, ?, ?, ?, ?, ?, ?)
            """,
            (
                f.ts,
                f.location,
                f.horizon_h,
                f.predicted_temp,
                f.reaction_time_est,
                f.reaction_class,
                f.cluster_id,
            ),
        )
        conn.commit()


    # ----------------------- Моніторинг телеметрії (MQTT → БД) ------------------
-----


    class TelemetryMonitor:
        def __init__(self, db_path: str, broker: str, port: int, topic: str):
            self.db_conn = init_timeseries_db(db_path)
            self.topic = topic
            self.client = mqtt.Client()
            self.client.on_connect = self._on_connect
            self.client.on_message = self._on_message
```

```
        self.client.connect(broker, port, keepalive=60)

    def _on_connect(self, client, userdata, flags, rc):
        if rc == 0:
            logger.info("MQTT підключено, підписка на %s", self.topic)
            client.subscribe(self.topic)
        else:
            logger.error("Помилка MQTT підключення, код %s", rc)

    def _on_message(self, client, userdata, msg):
        try:
            payload = json.loads(msg.payload.decode("utf-8"))
            r = SensorReading(
                ts=payload.get("ts", time.time()),
                location=payload["location"],
                temperature=float(payload["temperature"]),
                humidity=float(payload["humidity"]),
            )
            if self._is_valid_reading(r):
                insert_timeseries(self.db_conn, r)
                logger.debug("Записано телеметрію: %s", r)
            else:
                logger.warning("Некоректне вимірювання: %s", payload)
        except Exception as ex:
            logger.exception("Помилка оброблення MQTT-повідомлення: %s",
ex)

    @staticmethod
    def _is_valid_reading(r: SensorReading) -> bool:
        if not (10.0 <= r.temperature <= 35.0):
```

```
            return False
        if not (10.0 <= r.humidity <= 90.0):
            return False
        return True

    def run_forever(self):
        logger.info("Запуск моніторингу телеметрії")
        self.client.loop_forever()


# ------------------------ Сервіс прогнозування та кластеризації ------------------
------


class ForecastEngine:
    def __init__(self, ts_db_path: str, forecast_db_path: str):
        self.ts_conn = init_timeseries_db(ts_db_path)
        self.forecast_conn = init_forecast_db(forecast_db_path)
        self.model: Optional[RandomForestRegressor] = None

    def _load_timeseries(self, location: str, n: int = 200) -> np.ndarray:
        cur = self.ts_conn.cursor()
        cur.execute(
            """
            SELECT ts, temperature, humidity
            FROM timeseries
            WHERE location = ?
            ORDER BY ts ASC
            """,
            (location,),
```

```
            )
            rows = cur.fetchall()
            if not rows:
                raise RuntimeError("Недостатньо даних для прогнозу")
            rows = rows[-n:]
            arr = np.array(rows, dtype=float)
            return arr  # shape: (N, 3) -> ts, T, RH

        def _get_or_train_model(self, X: np.ndarray, y: np.ndarray) ->
RandomForestRegressor:
            if self.model is not None:
                return self.model

            if os.path.exists(MODEL_PATH):
                self.model = joblib.load(MODEL_PATH)
                logger.info("Завантажено модель прогнозування з %s",
MODEL_PATH)
            else:
                logger.info("Навчання нової моделі прогнозування")
                model = RandomForestRegressor(
                    n_estimators=300,
                    random_state=42,
                    n_jobs=-1,
                )
                model.fit(X, y)
                joblib.dump(model, MODEL_PATH)
                self.model = model
                logger.info("Модель збережено у %s", MODEL_PATH)
            return self.model
```

```
        @staticmethod
        def _estimate_reaction_time(
            predicted_temp: float,
            current_temp: float,
            t_min: float = 21.0,
            t_max: float = 23.0,
        ) -> float:
            if t_min <= predicted_temp <= t_max:
                return 2.5
            deviation = abs(predicted_temp - (t_min + t_max) / 2.0)
            return 2.5 + deviation * 0.5

        @staticmethod
        def _classify_reaction(reaction_time: float) -> str:
            return "Fast" if reaction_time <= REACTION_THRESHOLD_SEC else
"Slow"

        @staticmethod
        def _cluster_point(points: np.ndarray, new_point: np.ndarray) -> int:
            kmeans = KMeans(n_clusters=N_CLUSTERS, random_state=42)
            kmeans.fit(points)
            cid = int(kmeans.predict(new_point.reshape(1, -1))[0])
            return cid

        def forecast_for_location(self, location: str, horizon_h: int = 1) ->
ForecastResult:
            arr = self._load_timeseries(location)
            ts = arr[:, 0]
            temp = arr[:, 1]
            rh = arr[:, 2]
```

```
            # Формуємо ознаки й ціль (T(k+1))
            X = np.column_stack([temp[:-1], rh[:-1]])
            y = temp[1:]

            model = self._get_or_train_model(X, y)

            last_t = temp[-1]
            last_rh = rh[-1]
            last_ts = ts[-1]

            x_new = np.array([[last_t, last_rh]])
            predicted = float(model.predict(x_new)[0])

            reaction_time_est = self._estimate_reaction_time(
                predicted_temp=predicted,
                current_temp=last_t,
            )
            reaction_class = self._classify_reaction(reaction_time_est)

            # Кластеризація за (T, RH, reaction_time)
            points = np.column_stack([temp, rh, np.full_like(temp,
reaction_time_est)])
            new_point = np.array([predicted, last_rh, reaction_time_est])
            cluster_id = self._cluster_point(points, new_point)

            forecast = ForecastResult(
                location=location,
                ts=last_ts,
                horizon_h=horizon_h,
```

```
            predicted_temp=predicted,
            reaction_time_est=reaction_time_est,
            reaction_class=reaction_class,
            cluster_id=cluster_id,
        )

        insert_forecast(self.forecast_conn, forecast)
        logger.info(
            "Прогноз для %s: T=%.2f, reaction=%.2fs (%s), cluster=%d",
            location,
            predicted,
            reaction_time_est,
            reaction_class,
            cluster_id,
        )
        return forecast


# ---------------------- Модуль MPC-керування ----------------------


class MPCController:
    def __init__(self, broker: str, port: int, control_topic: str):
        self.control_topic = control_topic
        self.client = mqtt.Client()
        self.client.connect(broker, port, keepalive=60)

    @staticmethod
    def get_zone_config(location: str, reaction_class: str) -> ZoneConfig:
        cfg = ZoneConfig(
```

```
            location=location,
            t_setpoint=22.0,
            u_min=0.0,
            u_max=1.0,
            horizon=12,
            w_tracking=1.0,
            w_energy=0.1,
        )
        if reaction_class == "Slow":
            cfg.horizon = 8
            cfg.w_tracking = 2.0
        return cfg

    @staticmethod
    def simple_thermal_model(T_prev: float, u: float, T_out: float) -> float:
        a, b, c = 0.9, 0.5, 0.05
        return a * T_prev + b * u + c * (T_out - T_prev)

    def _solve_mpc(
        self,
        cfg: ZoneConfig,
        T0: float,
        T_out_forecast: List[float],
    ) -> List[float]:
        H = cfg.horizon
        candidate_us = np.linspace(cfg.u_min, cfg.u_max, 5)
        best_cost = float("inf")
        best_seq = [0.0] * H

        def cost_for_seq(u_seq: List[float]) -> float:
```

```
            T = T0
            cost = 0.0
            for k in range(H):
                u = u_seq[k]
                T = self.simple_thermal_model(T, u, T_out_forecast[k])
                cost += cfg.w_tracking * (T - cfg.t_setpoint) ** 2
                cost += cfg.w_energy * u ** 2
            return cost

        def dfs(k: int, seq: List[float]):
            nonlocal best_cost, best_seq
            if k == H:
                c = cost_for_seq(seq)
                if c < best_cost:
                    best_cost = c
                    best_seq = seq.copy()
                return
            for u in candidate_us:
                seq.append(float(u))
                dfs(k + 1, seq)
                seq.pop()

        dfs(0, [])
        return best_seq

    def build_and_send_plan(
        self,
        forecast: ForecastResult,
        current_temp: float,
        T_out_forecast: List[float],
```

```
        ) -> ControlPlan:
            cfg = self.get_zone_config(forecast.location, forecast.reaction_class)
            u_seq = self._solve_mpc(cfg, T0=current_temp, T_out_forecast=T_out_forecast)

            plan = ControlPlan(
                location=forecast.location,
                ts=time.time(),
                u_sequence=u_seq,
                reaction_class=forecast.reaction_class,
                cluster_id=forecast.cluster_id,
            )

            payload = {
                "location": plan.location,
                "ts": plan.ts,
                "u_sequence": plan.u_sequence,
                "reaction_class": plan.reaction_class,
                "cluster_id": plan.cluster_id,
            }
            self.client.publish(MQTT_TOPIC_CONTROL, json.dumps(payload))
            logger.info("Надіслано план керування для %s: %s", plan.location, payload)
            return plan


    # ---------------------- Головний цикл демонстраційного сценарію ----------
```

```
def main_demo():
    """
    Демонстраційний сценарій:
    1) Прогноз для заданої локації;
    2) Формування MPC-плану;
    3) Публікація плану через MQTT.
    """
    location = "ROOM_101"

    # Ініціалізація сервісів
    fe = ForecastEngine(DB_PATH, FORECAST_DB_PATH)
    mpc = MPCController(MQTT_BROKER, MQTT_PORT, MQTT_TOPIC_CONTROL)

    # Отримуємо прогноз (на основі історичних даних)
    forecast = fe.forecast_for_location(location, horizon_h=1)

    # Поточна T (спрощено – беремо останній запис з БД)
    conn = init_timeseries_db(DB_PATH)
    cur = conn.cursor()
    cur.execute(
        """
        SELECT temperature
        FROM timeseries
        WHERE location = ?
        ORDER BY ts DESC
        LIMIT 1
        """,
        (location,),
    )
```

```
            row = cur.fetchone()
            if not row:
                raise RuntimeError("Немає даних для поточної температури")
            current_temp = float(row[0])

            # Спрощений прогноз зовнішньої температури (сталий або з невеликою варіацією)
            T_out_forecast = [forecast.predicted_temp - 3.0] * forecast.horizon_h
            while len(T_out_forecast) < MPCController.get_zone_config(
                location, forecast.reaction_class
            ).horizon:
                T_out_forecast.append(T_out_forecast[-1])

            # Формуємо та надсилаємо план керування
            mpc.build_and_send_plan(forecast, current_temp, T_out_forecast)


        if __name__ == "__main__":
            # Для реальної системи: окремо запускається TelemetryMonitor, окремо – main_demo()
            # Тут для прикладу запускаємо лише демонстраційний сценарій.
            try:
                main_demo()
            except Exception as e:
                logger.exception("Помилка при виконанні демо-сценарію: %s", e)
```