

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет робототехніки та приладобудування
Кафедра інформаційно-вимірювальних технологій**

До захисту допущено:

В. о. завідувача кафедри

_____ Наталія ЗАЩЕПКИНА

«___» _____ 2026 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інформаційні вимірювальні
технології»**

спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка»

**на тему: «Інтелектуальна система розумного будинку з адаптивним
налаштуванням параметрів»**

Виконала:

студентка IV курсу, групи ПІ-321

Бойко Ольга Володимирівна

Керівник:

доцент, к.т.н., доцент

Добролюбова Марина Валеріївна

Рецензент:

доцент, к.т.н., доцент

Петрик Валентин Федорович

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Студентка _____

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) робототехніки та приладобудування

Кафедра інформаційно-вимірювальних технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність (спеціалізація) 152 Метрологія та інформаційно-вимірювальна техніка

Освітньо-професійна програма «Інформаційні вимірювальні технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Володимир ЄРЕМЕНКО

«___» квітня 2026 р.

ЗАВДАННЯ

на дипломну роботу студентці

Бойко Ользі Володимирівні

1. Тема роботи Інтелектуальна система розумного будинку з адаптивним налаштуванням параметрів

керівник роботи Добролюбова Марина Валеріївна, к.т.н., доц., затверджені наказом по університету від «25» травня 2026 р. № 1888-с.

2. Термін подання студентом роботи «___» червня 2026.

3. Вихідні дані до роботи. Система повинна забезпечити:

а) збір та збереження результатів вимірювань температури навколишнього середовища в діапазоні від -20 °C до +50 °C;

б) збір та збереження результатів вимірювань відносної вологості повітря- в діапазоні від 0 % до 100 %;

в) автоматичне формування навчальної вибірки на основі пар «бюджетний» сенсор – еталонний сенсор;

г) виконання самокалібрування сенсорних вузлів із застосуванням методів машинного навчання;

д) виявлення аномальних значень у вимірювальних даних в режимі реального часу;

ж) локальне збереження, оброблення та візуалізацію вимірювальної інформації;

з) передачу даних між сенсорними вузлами, шлюзами та серверною частиною із застосуванням безпроводних технологій;

і) оцінювання ефективності алгоритму самокалібрування.

4. Зміст (дипломної роботи) розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Вступ. Аналіз сучасних підходів до калібрування та самокалібрування сенсорних вузлів IBC. Аналіз функціональних обмежень типових платформ «розумного дому» та обґрунтування доцільності використання ML. Розроблення структури та архітектури IBC моніторингу параметрів навколишнього

середовища. Розроблення алгоритму самокалібрування та онлайн-виявлення аномалій. Дослідження ефективності запропонованого алгоритму самокалібрування та оцінювання похибок результатів вимірювань. Висновки.

5. Перелік (ілюстративного) графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів тощо) Плакати: схема структурна; схема інформаційних потоків та взаємодії компонентів системи; архітектура програмного забезпечення системи самокалібрування, алгоритм формування навчальної вибірки та навчання коригувальної моделі, алгоритм виявлення аномалій у вимірювальних даних, результати тестування та експериментального дослідження системи, порівняння характеристик розробленого рішення з існуючими платформами. Презентаційні та роздатні матеріали – не менш ніж 20 слайдів.

6. Консультанти розділів проєкту*

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

7. Дата видачі завдання 20 квітня 2026 року

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз підходів до калібрування та самокалібрування сенсорних вузлів і функціональних обмежень типових платформ «розумного дому»	22.04.2026 р.	виконано
2.	Розроблення структури та архітектури ІВС	28.04.2026 р.	виконано
3.	Розроблення алгоритму самокалібрування сенсорних вузлів та онлайн-виявлення аномалій на основі ML-методів	02.05.2026 р.	виконано
4.	Розроблення програмного забезпечення збору, зберігання, обробки та візуалізації інформації	05.05.2026 р.	виконано
5.	Дослідження ефективності алгоритму та оцінювання похибок результатів вимірювання	14.05.2026 р.	виконано
6.	Оформлення дипломної роботи	01.06.2026 р.	виконано
7.	Оформлення графічних матеріалів	01.06.2026 р.	виконано
8.	Перевірка роботи на ознаки плагіату	01.06 – 05.06.2026	виконано
9.	Відгук керівника	09.06 – 12.06.2026	виконано
11.	Рецензування дипломної роботи	09.06 – 12.06.2026	виконано
13.	Подання роботи на підпис в.о. зав. каф.	червень 2026 р.	виконано
14.	Захист дипломної роботи	червень 2026 р.	виконано

Студентка _____

Ольга БОЙКО

Керівник роботи _____

Марина ДОБРЮЛОВА

АНОТАЦІЯ

Дипломна робота присвячена розробленню інтелектуальної інформаційно-вимірювальної системи розумного будинку з адаптивним налаштуванням параметрів, самокалібруванням сенсорних вузлів та онлайн-виявленням аномальних значень у вимірювальних даних.

Актуальність роботи зумовлена необхідністю підвищення достовірності даних, що надходять від бюджетних сенсорів температури, вологості та якості повітря у побутових IoT-системах. У процесі тривалої експлуатації такі сенсори можуть накопичувати систематичні похибки, зазнавати дрейфу характеристик та формувати недостовірні покази, що негативно впливає на роботу автоматизованих сценаріїв розумного будинку.

У роботі розглянуто сучасні підходи до калібрування та самокалібрування сенсорних вузлів, проаналізовано функціональні обмеження типових платформ «розумного дому» та обґрунтовано доцільність використання методів машинного навчання для корекції вимірювальних даних. Розроблено архітектуру системи, яка передбачає збір даних із сенсорних вузлів, їх передавання через шлюз та подальше оброблення у зовнішньому програмному сервісі без необхідності зміни прошивки кінцевих пристроїв.

Програмну частину реалізовано як сервіс на основі FastAPI, PostgreSQL, MQTT, REST- та WebSocket-інтерфейсів. У системі передбачено збереження історії вимірювань, формування навчальної вибірки на основі пар «бюджетний сенсор – еталонний сенсор», навчання коригувальної моделі, застосування самокалібрування до нових показів, виявлення аномалій за допомогою робастних статистичних методів та візуалізацію результатів у вебінтерфейсі оператора.

Особливу увагу приділено оцінюванню точності запропонованого алгоритму самокалібрування та розрахунку похибок результатів вимірювань. Проведене дослідження показало, що використання коригувальної моделі дає

змогу зменшити вплив дрейфу сенсорів, підвищити достовірність вимірювальної інформації та забезпечити більш надійну роботу системи моніторингу параметрів навколишнього середовища в умовах розумного будинку.

Ключові слова: розумний будинок, інформаційно-вимірювальна система, сенсорний вузол, самокалібрування, машинне навчання, дрейф сенсорів, виявлення аномалій, MQTT, REST, WebSocket, PostgreSQL, IoT.

ANNOTATION

This thesis is devoted to the development of an intelligent information-measuring smart home system with adaptive parameter adjustment, self-calibration of sensor nodes, and online detection of anomalous values in measurement data.

The relevance of the work is determined by the need to improve the reliability of data obtained from low-cost temperature, humidity, and air quality sensors used in household IoT systems. During long-term operation, such sensors may accumulate systematic errors, experience characteristic drift, and generate unreliable readings, which negatively affects the operation of automated smart home scenarios.

The thesis considers modern approaches to calibration and self-calibration of sensor nodes, analyzes the functional limitations of typical smart home platforms, and substantiates the use of machine learning methods for correcting measurement data. The architecture of the system has been developed, which provides data collection from sensor nodes, transmission through a gateway, and further processing in an external software service without the need to modify the firmware of end devices.

The software part is implemented as a service based on FastAPI, PostgreSQL, MQTT, REST, and WebSocket interfaces. The system provides storage of measurement history, formation of a training dataset based on pairs of “low-cost sensor – reference sensor”, training of a correction model, application of self-calibration to new readings, anomaly detection using robust statistical methods, and visualization of results in the operator web interface.

Particular attention is paid to evaluating the accuracy of the proposed self-calibration algorithm and calculating measurement errors. The conducted research showed that the use of the correction model makes it possible to reduce the influence of sensor drift, improve the reliability of measurement information, and ensure more stable operation of the environmental parameter monitoring system in smart home conditions.

Keywords: smart home, information-measuring system, sensor node, self-calibration, machine learning, sensor drift, anomaly detection, MQTT, REST, WebSocket, PostgreSQL, IoT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	4
ВСТУП	5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ	9
1.1 Проблема достовірності сенсорних даних у системах «розумного дому»	9
1.2 Традиційні методи калібрування та самокалібрування сенсорів.....	10
1.3 Використання методів машинного навчання для компенсації дрейфу та аномалій	11
1.4 Огляд наявних платформ і програмних засобів інтеграції	12
1.5 Висновки до розділу 1	13
2 АПАРАТНО-ПРОГРАМНА РЕАЛІЗАЦІЯ ТА СТРУКТУРА ЗАПРОПОНОВАНОЇ СИСТЕМИ	14
2.1 Порівняння обраних пристроїв з поширеними аналогами	14
2.2 Програмна реалізація системи та опис основних модулів коду	18
2.3 Структурна схема системи та програмного додатка	22
2.4 Загальна концепція розробленої системи.....	25
2.5 Логічна структура програмної системи.....	26
2.6 Модель даних і спосіб організації збереження	27
2.7 Алгоритм самокалібрування.....	29
2.8 Онлайн-виявлення аномалій	32
2.9 Принцип роботи фронтенда та сценарій взаємодії з оператором.....	33
2.10 Висновки до розділу 2	34
3 ПОРІВНЯННЯ ЗАПРОПОНОВАНОЇ СИСТЕМИ З ІСНУЮЧИМИ ПІДХОДАМИ	36
3.1 Критерії порівняння	36
3.2 Порівняння з типовими платформами «розумного дому»	37
3.3 Порівняння з Home Assistant як платформою без ML-модуля.....	37
3.4 Порівняння з традиційним ручним калібруванням	38
3.5 Переваги запропонованого рішення	39
3.6 Обмеження та напрями подальшого розвитку.....	39

3.7 Висновки до розділу 3	40
4 ОЦІНЮВАННЯ ПОХИБОК РЕЗУЛЬТАТІВ ВИМІРЮВАНЬ ТА ЕФЕКТИВНОСТІ АЛГОРИТМУ САМОКАЛІБРУВАННЯ.....	41
4.1 Показники оцінювання похибок результатів вимірювань.....	41
4.2 Розрахунок похибок у симульованому сценарії	43
4.3. Оцінювання ефективності алгоритму самокалібрування на архівному сценарії	46
4.4. Обмеження оцінювання та інтерпретація результатів	48
4.5. Висновки до розділу 4	48
ВИСНОВКИ.....	50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	53
ДОДАТКИ.....	55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

TVOC (Total Volatile Organic Compounds) – сумарний вміст летких органічних сполук;

CO₂ – діоксид вуглецю;

RMSE (Root Mean Squared Error) – середньоквадратична похибка;

MAE (Mean Absolute Error) – середня абсолютна похибка;

MAD (Median Absolute Deviation) – медіанне абсолютне відхилення;

WebSocket – протокол двобічної взаємодії клієнта і сервера в реальному часі;

REST (Representational State Transfer) – архітектурний стиль побудови вебсервісів;

MQTT (Message Queuing Telemetry Transport) – легковаговий протокол передавання телеметрії;

API (Application Programming Interface) – програмний інтерфейс застосунку;

ML (Machine Learning) – машинне навчання;

IoT (Internet of Things) – Інтернет речей;

JSON (JavaScript Object Notation) – текстовий формат обміну структурованими даними;

HTTP/HTTPS (HyperText Transfer Protocol / Secure) – протоколи прикладного рівня для обміну вебданими;

SQL (Structured Query Language) – мова структурованих запитів до реляційних баз даних;

CRUD (Create, Read, Update, Delete) – базовий набір операцій над даними;

CORS (Cross-Origin Resource Sharing) – механізм керування доступом до ресурсів між різними джерелами;

Pydantic – бібліотека Python для валідації даних на основі типів.

ВСТУП

У сучасних системах «розумного дому» сенсорні вузли виконують роль первинного джерела даних для всіх автоматизованих рішень: керування мікрокліматом, освітленням, вентиляцією, енергоспоживанням, безпекою та сервісними сценаріями. Практична цінність таких систем безпосередньо залежить від достовірності вимірних параметрів. Якщо сенсор температури, вологості чи концентрації CO₂ накопичує систематичну похибку, то система автоматизації починає ухвалювати рішення на основі викривленого уявлення про стан середовища. Саме тому проблема підтримання точності сенсорних показів упродовж тривалої експлуатації стає одним із ключових чинників якості побутових IoT-рішень [1].

Для бюджетних платформ проблема ускладнюється тим, що більшість масових пристроїв створюються за моделлю «plug-and-play»: користувачеві надається готовий сенсор, який швидко додається до екосистеми, але не має повноцінних інструментів ручного калібрування, метрологічної перевірки або адаптивної компенсації дрейфу. Такі продукти орієнтовані переважно на комфорт та доступність, а не на відтворення характеристик професійних вимірювальних систем. У результаті тривала робота в умовах зміни температури, вологості, забруднення сенсорних елементів, старіння електронних компонентів чи нестабільності живлення може призвести до накопичення відхилень, що безпосередньо впливають на точність автоматизації [1, 2].

Проблема дрейфу сенсорів та автоматичної компенсації похибок активно досліджується в науковій літературі. Сучасні роботи демонструють, що поєднання класичних підходів до корекції із машинним навчанням дозволяє значно підвищити стійкість низьковартісних вимірювальних вузлів. Зокрема, для неконтрольованих середовищ розглядаються методи самокалібрування без постійного доступу до лабораторного еталону, а для низьковартісних IoT-платформ – алгоритми, які можуть виконуватися локально, на межі мережі або у зовнішньому сервісному модулі [2-5].

Актуальність теми зумовлена суперечністю між двома тенденціями. З одного боку, ринок «розумного дому» стрімко розширюється і дедалі частіше включає екологічний моніторинг, контроль комфорту, енергетичну оптимізацію та локальну аналітику. З іншого боку, більшість доступних рішень не забезпечує автоматичного підстроювання вимірювальних каналів під реальні умови експлуатації. Забезпечення достовірності вимірювальної інформації в умовах тривалої експлуатації бюджетних сенсорних вузлів є актуальною метрологічною проблемою, оскільки дрейф характеристик сенсорів, накопичення систематичних відхилень та вплив умов експлуатації безпосередньо погіршують якість вимірювальної інформації, що є основою для прийняття автоматизованих рішень. Це створює потребу в такій архітектурі, яка дозволила б інтегрувати інтелектуальне самокалібрування у вже наявну побутову платформу без перепрошивки шлюзів і без переходу до дорогих промислових систем.

Метою дипломної роботи є розроблення та дослідження інформаційно-вимірювальної системи моніторингу параметрів навколишнього середовища на базі технологій «розумного дому» з алгоритмом самокалібрування на основі методів машинного навчання для зменшення похибки сенсорних вимірювань, автоматичної компенсації дрейфу та виявлення аномалій із забезпеченням інтеграції з бюджетними IoT-платформами без модифікації їхньої базової прошивки [1].

Для досягнення поставленої мети необхідно розв'язати наступні **завдання**:

- проаналізувати існуючі підходи до калібрування й самокалібрування сенсорних вузлів;
- дослідити функціональні обмеження типових платформ «розумного дому» щодо збору, оброблення та передавання вимірювальної інформації;
- обґрунтувати доцільність використання методів машинного навчання для корекції результатів вимірювань та самокалібрування сенсорних вузлів;
- спроектувати архітектуру інформаційно-вимірювальної системи з локальним збором, збереженням і нормуванням даних;

- розробити алгоритм оцінювання відхилень результатів вимірювань, побудови коригувальної моделі самокалібрування та онлайн- виявлення аномалій;
- виконати порівняння розробленого рішення з наявними платформами;
- для підтвердження ефективності запропонованого підходу в роботі передбачено дослідження впливу алгоритму самокалібрування на точність та достовірність вимірювальної інформації. Оцінювання виконується шляхом порівняння похибок результатів вимірювань до та після застосування коригувальної моделі. Такий підхід дає змогу кількісно визначити, наскільки використання алгоритму самокалібрування зменшує вплив дрейфу сенсорів і підвищує якість даних, що використовуються інформаційно-вимірювальною системою розумного будинку;

Об’єктом дослідження є процес моніторингу параметрів навколишнього середовища в системах «розумного будинку» на основі мережі розподілених сенсорних вузлів.

Предметом дослідження є методи самокалібрування та виявлення аномалій сенсорних вузлів на основі машинного навчання і програмна архітектура їх інтеграції в інформаційно-вимірювальну систему моніторингу параметрів навколишнього середовища. У роботі використано методи аналізу часових рядів, робастної статистики, регресійного моделювання, програмної інженерії та системного проєктування IoT-платформ.

Наукова новизна роботи полягає у розвитку архітектури інформаційно-вимірювальної системи моніторингу параметрів навколишнього середовища, в якій модуль самокалібрування сенсорних вузлів реалізовано як зовнішній програмний сервіс, що інтегрується з наявною сенсорною інфраструктурою через стандартні інтерфейси обміну даними та забезпечує автоматичну компенсацію дрейфу похибки вимірювальних каналів і виявлення аномалій без модифікації прошивки вимірювальних вузлів.

Практичне значення полягає в тому, що розроблена інформаційно-вимірювальна система може бути використана для моніторингу параметрів навколишнього середовища із забезпеченням автоматичної компенсації дрейфу похибки сенсорних вузлів різних типів без участі користувача. Реалізований програмний сервіс може бути інтегрований з існуючою сенсорною інфраструктурою через стандартні інтерфейси без модифікації прошивки вимірювальних вузлів, що робить його придатним для впровадження в системах побутового екологічного моніторингу, та може слугувати основою для подальшого розвитку в напрямі багатоканальних інформаційно-вимірювальних систем.

Апробація результатів роботи. За отриманими результатами зроблено доповідь на XXI Науково-практичній конференції студентів, аспірантів та молодих вчених «Ефективність та автоматизація інженерних рішень у приладобудуванні» (10 – 11 грудня 2025 р., м. Київ, Україна) на тему «Система «розумний дім» з алгоритмом самокалібрування на основі методів машинного навчання» з публікацією у збірнику матеріалів конференції та подано рукопис наукової статті у фаховий журнал катеборії Б «Вісник Вінницького політехнічного інституту» (квітень 2026 р., знаходиться у черзі на опублікування) на тему «Система самокалібрування сенсорів розумного будинку на основі робастної регресії та онлайн-детекції аномалій».

Структура роботи. Структура роботи. Загальний обсяг роботи становить 55 сторінок друкованого тексту, робота містить 9 рисунків, 7 таблиць, 5 додатків на 25 сторінках. Список використаних джерел налічує 13 найменувань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ

1.1 Проблема достовірності сенсорних даних у системах «розумного дому»

Система «розумний дім» зазвичай формується як мережа різнотипних сенсорів і виконавчих пристроїв, об'єднаних через шлюз або програмний центр автоматизації. На відміну від лабораторних чи промислових комплексів, така мережа функціонує у змінному, часто неконтрольованому середовищі: сенсори встановлюються поруч із вікнами, батареями, вентиляційними каналами, кухонною технікою, джерелами пилу та вологи. Усе це сприяє накопиченню систематичних і випадкових похибок. Покази сенсора можуть відхилятися як поступово, через старіння, так і стрибкоподібно – через локальні впливи або тимчасові завади [2, 4].

У побутових сценаріях такі відхилення не завжди помітні одразу, але вони мають кумулятивний ефект. Якщо температура в кімнаті систематично завищується на 1 – 2 °C, алгоритм керування опаленням буде зменшувати подачу тепла раніше, ніж це потрібно; якщо сенсор вологості занижує значення, вентиляція може вмикатися із запізненням; якщо датчик CO₂ дрейфує вгору, користувач отримає хибні попередження і втратить довіру до системи. Отже, проблема точності в «розумному домі» стосується не лише вимірювання як такого, а і якості автоматизованих рішень, побудованих на цих вимірюваннях [1, 4].

Особливість масового ринку полягає в тому, що сенсорні вузли часто не мають окремого засобу перевірки або відкритого інтерфейсу корекції. У документації користувачеві пропонуються лише загальні вказівки з розміщення пристрою та базові сценарії діагностики, тоді як складніші питання точності фактично виносяться за межі типового використання. Через це користувач або взагалі не калібрує сенсори, або виконує корекцію на рівні домашньої

автоматизації вручну, що не масштабується при збільшенні кількості пристроїв [1].

1.2 Традиційні методи калібрування та самокалібрування сенсорів

Класичне калібрування передбачає порівняння показів вимірювального каналу з еталонним значенням у контрольованих умовах та подальше визначення поправочного коефіцієнта або калібрувальної характеристики. Для сенсорних мереж це може бути одноразове заводське калібрування, періодична сервісна повірка або ручне коригування користувачем. Перевагою такого підходу є зрозуміла метрологічна інтерпретація, однак його недоліком у побутовому IoT є висока трудомісткість та неможливість оперативно реагувати на зміну умов експлуатації [2, 3].

Самокалібрування розглядається як підхід, за якого система сама оцінює свою похибку та адаптує параметри вимірювального каналу. У роботах із сенсорних мереж для неконтрольованих середовищ підкреслюється, що таке налаштування може спиратися на взаємне порівняння вузлів, використання еталонних каналів, статистичну узгодженість показів або моделі, здатних враховувати часовий дрейф [2]. Ще раніше в дослідженнях інтелектуальних сенсорів було показано, що самокалібрування доцільно розглядати не як окрему операцію, а як складову загального алгоритму адаптивної реакції пристрою на зміну характеристик [3].

У найпростішому випадку самокалібрування реалізується як лінійна компенсація зміщення й масштабу. Проте для побутових низьковартісних сенсорів цього часто недостатньо: похибка може бути нелінійною, залежати від температури середовища, швидкості зміни параметра, завад або деградації чутливого елемента. Саме тому сучасні системи все частіше доповнюються елементами статистичного аналізу та машинного навчання, які дозволяють сформувати більш гнучку функцію корекції, ніж проста таблична поправка.

1.3 Використання методів машинного навчання для компенсації дрейфу та аномалій

У науковій літературі останніх років простежується чітка тенденція до використання ML-моделей для підвищення достовірності сенсорних мереж. У роботах, присвячених калібруванню низьковартісних IoT-сенсорів якості повітря, показано, що моделі машинного навчання здатні істотно покращити точність порівняно з необробленими показами за рахунок урахування залежностей між вимірюваними величинами та контекстними факторами [4]. Для задач компенсації довготривалого дрейфу досліджуються як лінійні та ансамблеві підходи, так і більш складні нейромережеві структури [5].

Окрема група публікацій фокусується на виявленні аномалій у IoT-мережах. У сучасних оглядах аномалій для IoT серед найбільш уживаних алгоритмів називаються Random Forest, Isolation Forest, автоенкодери, рекурентні мережі, градієнтний бустинг, SVM та інші методи, здатні працювати з неоднорідними часовими рядами [6]. Водночас автори підкреслюють, що для ресурсно-обмежених вузлів велике значення має не лише точність, а й вартість виконання алгоритму, затримка, обсяг пам'яті та можливість локального розгортання. Саме тому останні роботи приділяють увагу легковаговому периферійному штучному інтелекту (lightweight edge AI) та комбінації класичного оброблення сигналів з компактними ML-моделями [7].

Для дипломного проєкту це означає, що архітектура не повинна безумовно покладатися на великі глибинні мережі або віддалені хмарні обчислення. Натомість доцільно обрати такий підхід, який поєднає достатню точність з простотою розгортання, можливістю пояснення результатів і здатністю працювати в локальному зовнішньому застосунку. Саме з цієї причини в пропонованому прототипі використовуються робастна регресія для формування калібрувальної функції та робастна статистика для онлайн-виявлення аномалій.

1.4 Огляд наявних платформ і програмних засобів інтеграції

Серед програмних платформ, які найчастіше використовуються для побудови відкритих систем «розумного дому», особливе місце займає Home Assistant. Офіційна документація платформи підтверджує наявність MQTT-інтеграції, REST API та WebSocket API, що робить її зручною точкою інтеграції між фізичною сенсорною мережею та зовнішніми сервісами оброблення даних [8, 9]. Важливо й те, що власний фронтенд Home Assistant також активно використовує WebSocket і REST-підходи, тобто екосистема від початку орієнтована на обмін даними в реальному часі [9].

Для побудови дипломного прототипу також важливо, що Home Assistant підтримує інтеграцію з Xiaomi Home та дозволяє працювати зі шлюзами Xiaomi Gateway. У документації наведено перелік підтримуваних моделей, серед яких зазначено варіант DGNWG02LM з ідентифікатором lumi.gateway.v3, тобто Xiaomi Gateway 3 [11]. Це дає підстави розглядати Gateway 3 як транспортний вузол збору даних, а власний ML-модуль – як зовнішню надбудову, що виконує корекцію вимірів та формує додаткову аналітику без втручання у базову логіку роботи шлюзу.

Як серверну технологію для реалізації прототипу доцільно обрати FastAPI. Офіційна документація позиціонує FastAPI як швидкий Python-фреймворк для побудови API з опорою на типізацію, а також як інструмент, придатний не лише для навчальних задач, а й для продуктивного використання [10]. У контексті теми дипломної роботи це означає, що один стек може забезпечити як API для приймання сенсорних даних, так і вебінтерфейс оператора, а також фонові задачі для періодичного самокалібрування.

Отже, аналіз існуючих платформ показує, що масові екосистеми є достатніми для збору даних і побутової автоматизації, але не надають завершеного механізму інтелектуального самокалібрування. Натомість відкриті платформи на зразок Home Assistant, MQTT та FastAPI створюють сприятливе

програмне середовище для побудови зовнішнього ML-модуля, який компенсує дрейф, виявляє аномалії та працює локально.

1.5 Висновки до розділу 1

Проведений огляд дає підстави сформулювати задачу дипломної роботи як задачу створення програмно-апаратної системи, що поєднує доступну платформу «розумного дому» з окремим модулем самокалібрування на Python. Такий модуль має виконувати приймання сирих значень сенсорів, формування історії вимірів, автоматичне навчання коригувальної моделі за парними значеннями бюджетного та еталонного сенсора, онлайн-корекцію нових показів, виявлення аномалій і надання оператору вебінтерфейсу для перегляду стану системи.

Ключовою вимогою є відсутність необхідності перепрошивки шлюзу чи зміни прошивки сенсорів. Це означає, що архітектура повинна працювати як зовнішній сервіс, який інтегрується через стандартні інтерфейси обміну даними – MQTT, REST або WebSocket. Саме на таку модель орієнтовано наступний розділ, у якому розглянуто структуру запропонованої системи та обґрунтовано вибір окремих програмних компонентів.

2 АПАРАТНО-ПРОГРАМНА РЕАЛІЗАЦІЯ ТА СТРУКТУРА ЗАПРОПОНОВАНОЇ СИСТЕМИ

2.1 Порівняння обраних пристроїв з поширеними аналогами

Для побудови дипломного прототипу обрано чотири основні апаратні компоненти: сенсор температури та вологості Aqara WSDCGQ11LM, монітор якості повітря Aqara TVOC Air Quality Monitor, сенсор руху Aqara RTCGQ11LM та шлюз Xiaomi Gateway 3. Їх вибір зумовлений не максимальною функціональною насиченістю кожного окремого пристрою, а сумісністю з бюджетною локальною IoT-архітектурою, підтримкою безпроводної інтеграції та можливістю передавання даних у зовнішній програмний сервіс без втручання в мікропрограмне забезпечення кінцевих вузлів.

Порівняння доцільно виконувати за такими критеріями: тип безпроводного протоколу, доступність локальної інтеграції, достатність вимірюваних параметрів для задачі самокалібрування, енергетична автономність та відповідність архітектурі з окремим Python-сервісом. Саме з цих позицій у роботі зіставляються обрані компоненти з поширеними аналогами споживчого сегмента, зокрема SONOFF, Xiaomi, Philips Hue, Qingping, Netatmo та Aqara Hub M2.

Як показано на рисунку 2.1 і в таблиці 2.1, для каналу температури і вологості обраний сенсор Aqara WSDCGQ11LM поступається окремим аналогам за паспортною точністю, але виграє з погляду інтеграції в уже сформований Zigbee-контур, компактності та низького енергоспоживання. SONOFF SNZB-02P має кращі заявлені характеристики точності та швидше оновлення даних, однак у межах цього дипломного рішення більш важливою є сумісність з обраним шлюзом та сталість локального маршруту передавання телеметрії. Xiaomi Smart Temperature and Humidity Monitor 3 є зручним пристроєм для побутового моніторингу, але Bluetooth-орієнтована модель менш придатна до безперервного використання у схемі з зовнішнім сервісом самокалібрування.

Для моніторингу якості повітря в прототипі використано Aqara TVOC Air Quality Monitor. З практичної точки зору він доречний тим, що поєднує Zigbee-інтеграцію, локальний індикатор та тривалу автономність. При цьому слід врахувати, що цей пристрій належить до класу TVOC-моніторів якості повітря, а не до повноцінних високоточних NDIR CO₂-аналізаторів. Саме тому у порівнянні з Qingping Air Monitor Lite та Netatmo Healthy Home Coach він виглядає скромніше за кількістю каналів або стаціонарними можливостями, проте краще відповідає задачі бюджетного локального прототипу, в якому першорядне значення має інтегрованість у Zigbee-мережу та придатність до зовнішньої ML-обробки.

Сенсор руху Aqara RTCGQ11LM обрано як допоміжний контекстний пристрій системи. Його роль у поточному прототипі не пов'язана безпосередньо з регресійним самокалібруванням, однак він демонструє, що запропонована архітектура здатна працювати не лише з безперервними вимірювальними каналами, а й з подієвими джерелами. Порівняно з SONOFF SNZB-03P і Philips Hue Motion Sensor він забезпечує достатню дальність виявлення, широке поле огляду та просту інтеграцію зі шлюзом Xiaomi Gateway 3 без потреби в окремій екосистемі на зразок Hue Bridge.

Окремо слід порівняти транспортний вузол системи. Xiaomi Gateway 3 не є найбагатшим за функціональністю варіантом на ринку, однак у межах дипломної роботи саме він найбільш логічно узгоджується з уже обраними сенсорами та забезпечує збір Zigbee-подій, базову локальну взаємодію і можливість передавання даних у зовнішній сервіс. Aqara Hub M2 має розширені можливості, зокрема Ethernet та IR-керування, а SONOFF ZBBridge-P краще орієнтований на власну eWeLink-екосистему. Проте для розробленого прототипу першочерговим є не максимальна кількість функцій хабу, а надійне функціонування каналу «сенсор – шлюз – зовнішній сервіс самокалібрування».

У запропонованій архітектурі для реалізації процедури самокалібрування використовується концепція еталонного сенсора того самого фізичного типу, що і бюджетний вимірювальний вузол. В рамках даного дослідження працездатність

алгоритму перевірено на основі симульованих та архівних даних, де еталонний канал моделюється як сенсор без систематичного дрейфу. У практичному застосуванні роль еталонного каналу може виконувати сенсор вищого класу точності того самого фізичного типу, наприклад, Sensirion SHT40 для каналів температури та вологості, або інший попередньо відкалібрований чи сертифікований засіб вимірювальної техніки як джерело еталонних показів. Запропонована система не залежить від конкретної моделі еталонного сенсора, що забезпечує гнучкість її практичного впровадження.

Отже, вибір апаратної основи в роботі є інженерно вмотивованим. Обрані пристрої не завжди є найточнішими чи найфункціональнішими у своїх класах, однак вони найкраще відповідають цільовій постановці дипломного проєкту: побудові доступної системи «розумного дому» з локальним сервісом самокалібрування, який працює поверх уже наявної побутової інфраструктури без перепрошивки кінцевих вузлів.

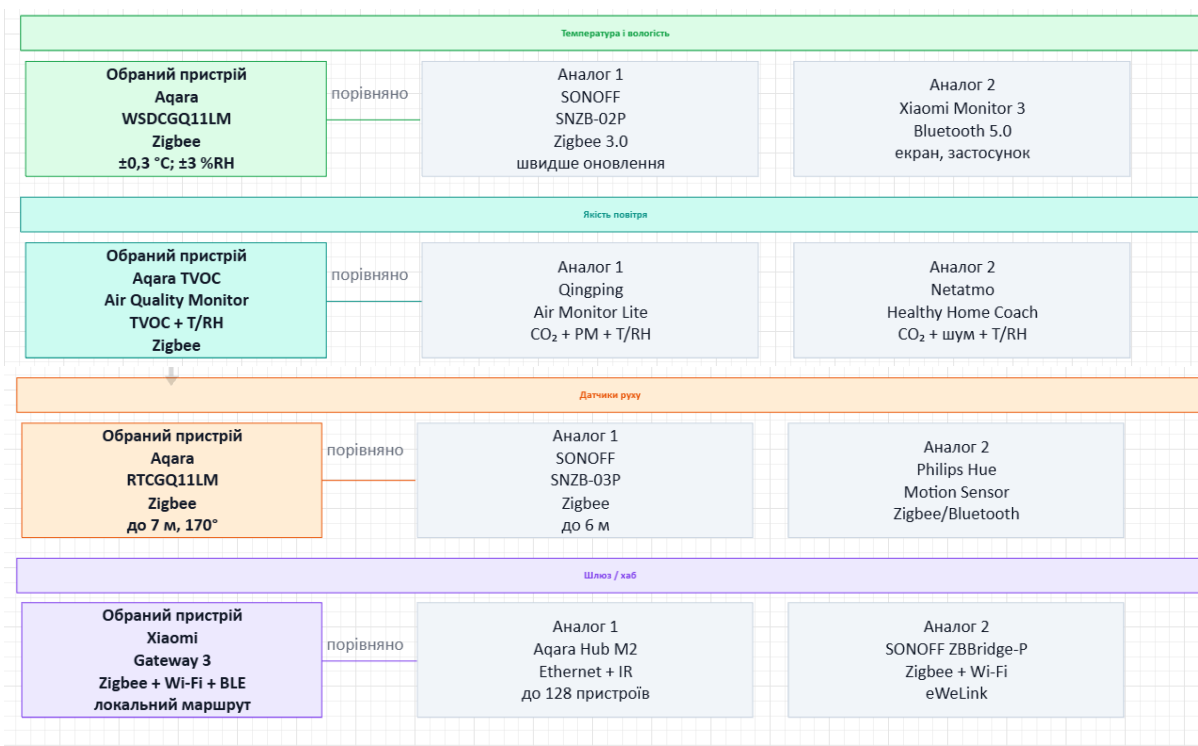


Рисунок 2.1 – Порівняння пристроїв, використаних у дипломному прототипі, з поширеними аналогами

Таблиця 2.1 – Порівняння обраних пристроїв з поширеними аналогами

Категорія	Обраний пристрій	Порівнювані аналоги	Узагальнений висновок
Температура і вологість	Aqara WSDCGQ11LM	SONOFF SNZB-02P; Xiaomi Smart Temperature and Humidity Monitor 3	Обраний сенсор забезпечує достатню функціональність і кращу сумісність із Zigbee-контуром прототипу, хоча окремі аналоги мають вищу заявлену точність або іншу екосистемну орієнтацію.
Якість повітря	Aqara TVOC Air Quality Monitor	Qingping Air Monitor Lite; Netatmo Healthy Home Coach	Пристрій краще відповідає бюджетній Zigbee-архітектурі, але за класом вимірювання належить до TVOC-моніторів, тому порівняння з CO ₂ -орієнтованими рішеннями слід трактувати як інженерне.
Датчик руху	Aqara RTCGQ11LM	SONOFF SNZB-03P; Philips Hue Motion Sensor	У межах прототипу вирішальними є простота інтеграції та локальна сумісність. Преміальні аналоги дають додаткові можливості, але не є критично необхідними для демонстраційної системи.
Шлюз / хаб	Xiaomi Gateway 3	Aqara Hub M2; SONOFF ZBBridge-P	Gateway 3 не є найфункціональнішим хабом у класі, проте саме він найкраще узгоджується з уже обраними вузлами та транспортною моделлю зовнішнього сервісу самокалібрування.

Таким чином, наведене порівняння слід інтерпретувати не як пошук формально «найкращого» окремого пристрою, а як вибір компонентів, що є оптимальними саме для заданої архітектури локального IoT-прототипу з зовнішнім модулем самокалібрування.

2.2 Програмна реалізація системи та опис основних модулів коду

Програмний додаток системи самокалібрування реалізовано як зовнішній сервіс на Python 3.11, який працює поверх побутової IoT-інфраструктури та не потребує змін у прошивці сенсорів або шлюзу. Такий підхід відповідає загальній концепції дипломної роботи, оскільки дозволяє відокремити алгоритмічну частину від транспортного рівня і водночас зберегти сумісність з масовими пристроями «розумного дому».

Технологічний стек прототипу охоплює FastAPI та Uvicorn для побудови серверної частини, PostgreSQL для збереження історії вимірювань і моделей, scikit-learn для навчання робастної регресії HuberRegressor, paho-mqtt для приймання MQTT-повідомлень, Jinja2 для шаблонів вебінтерфейсу, а також HTML, CSS і JavaScript для клієнтської панелі. Додатково в проєкті передбачено docker-compose-конфігурацію для швидкого розгортання інфраструктурних компонентів.

Центральним керівним модулем є файл main.py. Основні модулі програмного додатка та їх функції відображено в таблиці 2.2. Він ініціалізує FastAPI-застосунок, налаштовує життєвий цикл сервісу, CORS, статичні ресурси та шаблони, створює екземпляри репозиторію і сервісних класів, запускає фоновий цикл перенавчання, вмикає MQTT-підписку та відкриває REST- і WebSocket-інтерфейси. У коді реалізовано кінцеві точки /api/health, /api/summary, /api/sensors, /api/readings, /api/calibration/run, /api/models та /ws, що забезпечує як взаємодію з клієнтами, так і демонстраційне керування системою.

Таблиця 2.2 – Основні модулі програмного додатка та їх функції

Модуль	Призначення	Ключові дії та взаємодія
main.py	Точка входу FastAPI-застосунку	Ініціалізація сервісів і БД, налаштування CORS, статичних файлів, REST API, WebSocket, запуск retrain loop та MQTT-сервісу.

Продовження таблиці 2.2

Модуль	Призначення	Ключові дії та взаємодія
config.py	Централізована конфігурація	Зчитування APP_NAME, DATABASE_URL, MQTT_*, SIMULATOR_* і RETRAIN_* зі змінних середовища; формування об'єкта Settings.
database.py	Опис і ініціалізація схеми БД	Створення таблиць sensors, readings, calibration_models та SQL-представлень readings_enriched, sensor_dashboard_latest, dashboard_summary.
repository.py	Шар доступу до даних	CRUD-операції для сенсорів, запис показів, отримання history, snapshot, summary, збереження та читання моделей.
schemas.py	Опис структур вхідних даних	Pydantic-моделі ReadingCreate, SensorCreate та допоміжна валідація даних, що надходять через API і MQTT.
services/mqtt_service.py	Приймання MQTT-повідомлень	Підписка на брокер, декодування topic/payload, перетворення у ReadingCreate, передавання даних до IngestService.
services/ingest_service.py	Основний конвеєр обробки	Ідентифікація сенсора, застосування калібрувальної моделі, MAD-перевірка аномалій, запис reading у БД, підготовка результату для клієнта.
services/calibration_service.py	Самокалібрування каналів	Вирівнювання часових пар, навчання HuberRegressor, оцінювання MAE before/after, формування статусів calibrated, attention, stable, застосування моделі.
services/simulator_service.py	Генерація тестового середовища	Моделювання 4 кімнат, 20 сенсорів, дрейфу, шуму, сплесків і warm-up/online-сценаріїв для експериментальної перевірки системи.
services/websocket_manager.py	Розсилка оновлень у реальному часі	Підтримка активних WebSocket-з'єднань і трансляція подій про нові readings, моделі та стан сервісу.

Модуль	Призначення	Ключові дії та взаємодія
templates/index.html, static/app.js, style.css	Операторський вебінтерфейс	Відображення summary-карток, таблиці сенсорів, історії показів, моделей, ручного введення та посилань на Grafana.

Після надходження вимірювання конвеєр обробки реалізується у файлі `ingest_service.py`. Спочатку система або знаходить уже відомий сенсор, або створює його опис за вхідними атрибутами. Далі викликається метод `apply()` із `calibration_service.py`, який використовує останню прийняту модель для обчислення `calibrated_value` та `residual`. Після цього виконується MAD-оцінка аномальності на основі останніх 60 точок, а остаточний запис зберігається у таблиці `readings`.

Модуль `calibration_service.py` реалізує саме ядро самокалібрування. Він формує часово узгоджені пари «бюджетний сенсор – еталон», обмежує навчальну вибірку останніми 300 парами, вимагає не менше 12 узгоджених спостережень, навчає `HuberRegressor` та приймає модель лише тоді, коли похибка після корекції зменшується щонайменше на 2 %. Така логіка робить систему не лише адаптивною, а й безпечною з експлуатаційного погляду, оскільки в разі недостатньої якості моделі вона повертається до нейтральної поведінки.

Важливу роль у проєкті відіграє `simulator_service.py`. Саме він відтворює чотири кімнати з різними профілями температури, вологості та CO_2 /якості повітря, моделює дрейф, шум і короткочасні збурення, а отже забезпечує керований експериментальний полігон для перевірки алгоритму. На рівні інтерфейсу JavaScript-код формує summary-картки, таблицю сенсорів, графік `history` та відображення параметрів останніх моделей, що дає змогу поєднати аналітичну логіку з наочним режимом демонстрації.

Отже, програмна реалізація системи має чітку сервісну структуру. Вона відокремлює конфігурацію, логіку приймання даних, алгоритм самокалібрування, схему збереження, емуляційне середовище та візуалізацію.

Саме така декомпозиція є доречною для дипломного проекту, оскільки полегшує подальший розвиток системи, її тестування та інтеграцію з іншими елементами домашньої автоматизації.

Для підсилення зв'язку між описом архітектури та програмною реалізацією в роботі доцільно окремо показати, як саме проходить один вимір від зовнішнього повідомлення до збереженого запису. Така деталізація демонструє, що програмний код не є набором ізольованих файлів, а утворює єдиний конвеєр обробки даних.

На рисунку 2.2 показано, що вхідний payload проходить валідацію, нормалізацію, застосування актуальної калібрувальної моделі, перевірку на аномальність і лише після цього записується до бази даних. У результаті операторська панель отримує не просто сире значення, а вже підготовлений аналітичний результат з ознаками якості даних.

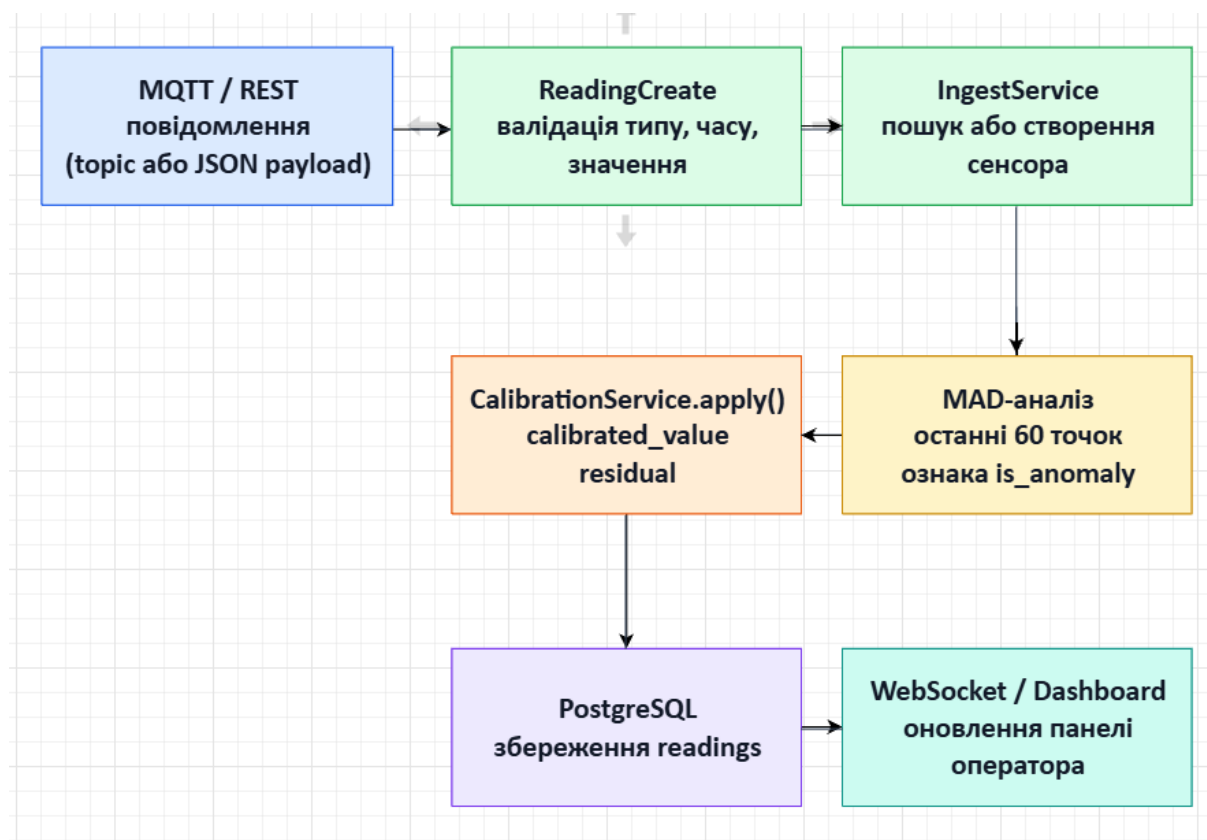


Рисунок 2.2 – Логіка обробки одного вимірювання у програмному сервісі

Фрагмент у лістингу 2.1 відображає ключовий принцип побудови серверної частини: API-метод не містить складної бізнес-логіки, а передає дані до сервісного шару. Це спрощує тестування, дозволяє повторно використовувати IngestService для MQTT-повідомлень і зменшує зв'язність між транспортним рівнем та алгоритмами обробки.

Лістинг 2.1 – Узагальнений фрагмент REST-прийому нового вимірювання

```
app = FastAPI(title=settings.app_name)
repository = Repository(settings.database_url)
calibration = CalibrationService(repository)
ingest = IngestService(repository, calibration, websocket_manager)

@app.post("/api/readings")
async def create_reading(payload: ReadingCreate):
    result = await ingest.process(payload)
    await websocket_manager.broadcast({
        "type": "reading.created",
        "data": result,
    })
    return result
```

2.3 Структурна схема системи та програмного додатка

З погляду системного проєктування запропоноване рішення доцільно розглядати у двох взаємопов'язаних площинах: як загальну структурну схему всієї апаратно-програмної системи та як внутрішню схему програмного додатка. Перша відображає шлях даних від фізичних сенсорів до оператора, а друга – внутрішню організацію коду, через яку реалізуються приймання даних, самокалібрування, виявлення аномалій і збереження результатів.

На рисунку 2.3 наведено структурну схему системи самокалібрування сенсорів розумного будинку. Вона об'єднує фізичний рівень сенсорів і шлюзу, канал передавання даних, серверний рівень обробки та рівень відображення. Такий поділ є важливим, оскільки показує, що алгоритм самокалібрування не вбудовується в окремий сенсор, а функціонує як зовнішній програмний шар між транспортною інфраструктурою та сервісами візуалізації.

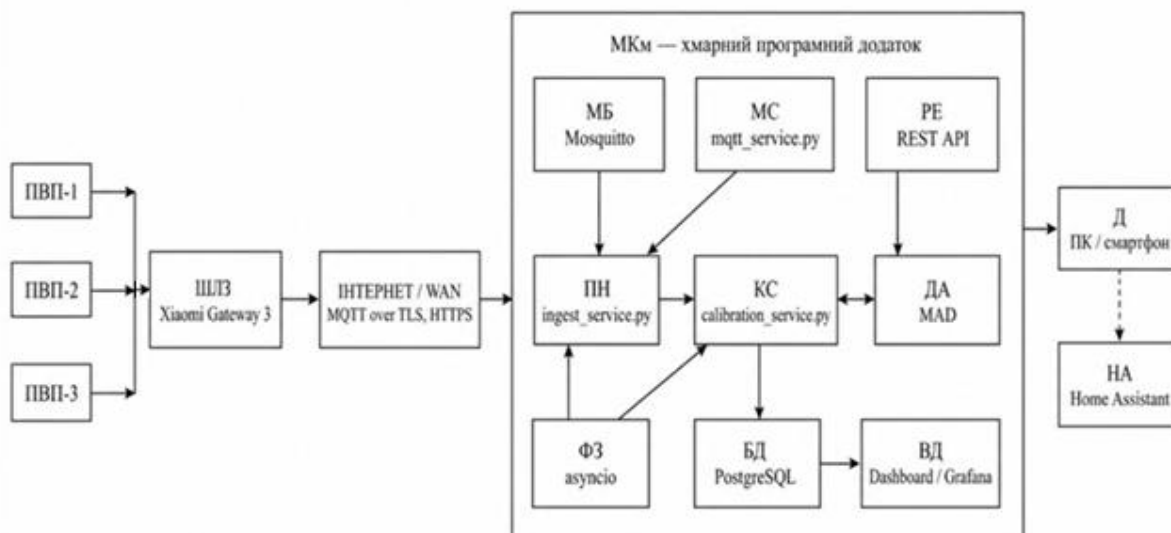


Рисунок 2.3 – Структурна схема системи самокалібрування сенсорів розумного будинку

На структурній схемі позначені такі елементи:

ПВП-1 – сенсор температури та вологості Aqara WSDCGQ11LM;

ПВП-2 – сенсор CO₂ / якості повітря Aqara TVOC-Z01;

ПВП-3 – сенсор руху / присутності Aqara RTCGQ11LM;

ШЛЗ – шлюз збору даних Xiaomi Gateway 3;

ІНТЕРНЕТ / WAN – мережа передавання даних між локальним шлюзом і хмарним сервером за протоколами MQTT over TLS та HTTPS;

МКм – модуль керування та моніторингу, реалізований як хмарний програмний додаток;

МБ – MQTT-брокер Mosquitto;

МС – модуль mqtt_service.py, який приймає дані з брокера та перетворює topic/payload у формат ReadingCreate;

РЕ – REST-інтерфейс POST /api/readings для альтернативного прийому даних через HTTPS;

ПН – модуль ingest_service.py для прийому, нормалізації, первинної обробки та запису даних;

КС – модуль calibration_service.py для навчання та застосування калібрувальних моделей;

ДА – модуль виявлення аномалій на основі MAD;

ФЗ – фонові задачі asuncіo, що виконують перенавчання моделей і сервісні цикли;

БД – база даних PostgreSQL;

ВД – засоби відображення даних у веб-дашборді та Grafana;

Д – користувацький пристрій відображення даних (ПК, смартфон або планшет);

НА – Home Assistant, перспективний модуль інтеграції системи домашньої автоматизації.

На фізичному рівні система включає сенсор температури та вологості Aqara WSDCGQ11LM, монітор якості повітря Aqara TVOC Air Quality Monitor, сенсор руху Aqara RTCGQ11LM та шлюз Xiaomi Gateway 3. Передавання подій від шлюзу до зовнішнього сервісу здійснюється через MQTT over TLS, HTTPS або REST API. На серверному рівні функціонують модулі MQTTIngestService, IngestService, CalibrationService, блок детекції аномалій, фонові задачі та база даних PostgreSQL, а на рівні відображення – веб-дашборд, Grafana, мобільний доступ і перспективний контур інтеграції з Home Assistant.

Окрім загальної системної схеми, для дипломної роботи важливо зафіксувати й внутрішні зв'язки між головними програмними модулями. Це дає можливість не лише показати архітектуру на рівні компонентів, а й аргументувати обраний спосіб поділу коду на окремі файли та сервіси. Відповідну структуру програмного додатка подано на рисунку 2.4.

Як видно з рисунка 2.4, верхній рівень програмної структури утворюють модулі входу та конфігурації: frontend-частина, main.py, config.py і допоміжні файли запуску. Середній рівень становлять сервісні модулі mqtt_service.py, ingest_service.py, calibration_service.py та simulator_service.py, які забезпечують обробку потоку даних. Нижній рівень охоплює repository.py, database.py з PostgreSQL та websocket_manager.py. Саме така організація створює прозорий маршрут проходження даних від payload або HTTP-запиту до збереженого результату і подальшого WebSocket-оновлення клієнта.

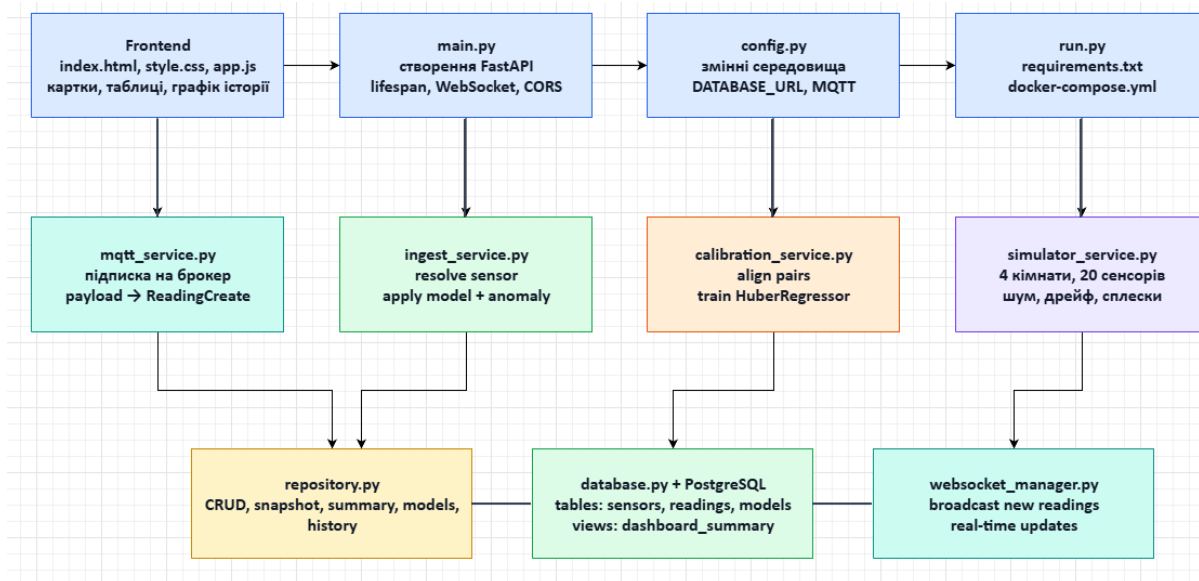


Рисунок 2.4 – Структурна схема програмного додатка та взаємодії основних модулів коду

Перевагою наведеної структурної декомпозиції є те, що окремі блоки можна розвивати незалежно. Наприклад, модуль симуляції може бути замінений на реальне джерело даних, модуль калібрування – на іншу модель машинного навчання, а фронтенд – на інтеграцію з Home Assistant або іншою операторською панеллю. Отже, структурна схема не лише описує поточний стан прототипу, а й задає основу для подальшої масштабованої еволюції дипломного рішення.

2.4 Загальна концепція розробленої системи

У цій дипломній роботі запропоновано архітектуру системи «розумний дім», у якій модуль самокалібрування реалізується не всередині прошивки шлюзу або окремого сенсора, а як зовнішній програмний сервіс на Python. Такий підхід дозволяє зберегти сумісність із побутовими IoT-пристроями та використовувати стандартні засоби інтеграції для збору телеметрії, не змінюючи базової логіки роботи сенсорної мережі. Як транспортне та інтеграційне середовище доцільно використовувати Home Assistant, оскільки офіційна документація платформи підтверджує наявність MQTT-інтеграції, REST API та

WebSocket API, а також підтримку Xiaomi Gateway моделі DGNWG02LM (lumi.gateway.v3) через інтеграцію Xiaomi Home.

Функціонально система складається з чотирьох основних контурів: збору даних, збереження та нормалізації, машинного навчання для самокалібрування та контуру візуалізації. У першому контурі дані надходять від сенсорів через MQTT, API або локальний симулятор. У другому контурі вимірювання накопичуються в базі даних PostgreSQL разом із часовими мітками, параметрами сенсора та оцінкою аномальності. У третьому контурі здійснюється побудова коригувальної моделі за парою «бюджетний сенсор – еталонний сенсор». У четвертому контурі результати подаються оператору через вебпанель, де доступні зведення, історія вимірів, статус моделей і ручний запуск самокалібрування.

Запропонована архітектура орієнтована на локальне розгортання. Це означає, що весь цикл – від приймання сирих значень до побудови скоригованих показів – виконується в межах локальної мережі або домашнього сервера. Такий підхід є важливим для систем «розумного дому», оскільки зменшує залежність від зовнішніх хмарних сервісів, знижує затримки оброблення і підвищує контроль користувача над власними даними.

2.5 Логічна структура програмної системи

Розроблена система реалізована за клієнт-серверною моделлю. Серверна частина побудована на FastAPI та забезпечує REST-інтерфейс для приймання нових вимірів, отримання списку сенсорів, доступу до історії показів, запуску пакетного самокалібрування та виведення статистичних зведень. Вибір FastAPI обґрунтований тим, що офіційна документація позиціонує його як сучасний високопродуктивний Python-фреймворк з опорою на типізацію, автоматичну генерацію API-документації та практичну придатність до production-сценаріїв.

На логічному рівні backend поділено на кілька незалежних модулів. Модуль repository відповідає за доступ до бази даних і зосереджує всю SQL-

логіку в одному місці. Модуль `ingest_service` приймає нові виміри, визначає сенсор, застосовує наявну калібрувальну модель і виконує первинну перевірку на аномальність. Модуль `calibration_service` формує вибірки для навчання, обчислює параметри коригувальної моделі та зберігає результати в базі. Модуль `mqtt_service` реалізує підключення до брокера повідомлень і перетворює зовнішні MQTT-повідомлення на внутрішній формат вимірів. Модуль `simulator_service` генерує тестові дані для демонстрації роботи системи без фізичного стенда. Окремо виділено `websocket_manager`, який забезпечує оновлення фронтенда в реальному часі.

Фронтенд реалізований у вигляді односторінкової панелі керування на HTML, CSS та JavaScript без використання важких зовнішніх фреймворків. Це зроблено свідомо, оскільки для дипломного прототипу пріоритетом є прозорість логіки, простота розгортання та мінімальна залежність від додаткових бібліотек. Через REST-запити панель періодично отримує зведення та стан сенсорів, а через WebSocket – реагує на оновлення системи в реальному часі.

2.6 Модель даних і спосіб організації збереження

Для збереження стану системи використано реляційну модель з трьома основними сутностями: `sensors`, `readings` і `calibration_models`. Таблиця `sensors` містить опис сенсора: його назву, тип, локацію, протокол, ознаку еталонного вузла та посилання на референсний сенсор. Таблиця `readings` фіксує кожне отримане вимірювання, включаючи сире значення, скориговане значення, залишок після корекції, часову мітку та ознаку аномалії. Таблиця `calibration_models` зберігає параметри побудованої моделі, якість до та після корекції, інтегральний показник дрейфу й додаткові службові характеристики навчання.

Такий підхід дозволяє вирішити одразу кілька задач. По-перше, система отримує повну історію вимірів для побудови часових графіків і повторного навчання моделей. По-друге, кожна нова версія калібрувальної моделі не затирає

попередню, а зберігається окремим записом, що дає змогу аналізувати еволюцію сенсора в часі. По-третє, логічне розділення на сенсори, покази та моделі спрощує подальше масштабування системи, наприклад додавання нових типів сенсорів, нових алгоритмів корекції або розширення фронтенда.

На рівні бази даних особливо важливо зберігати не лише останні значення сенсорів, а й повний історичний контекст. Саме історія потрібна для побудови навчальних пар, повторного розрахунку метрик, пояснення причин появи аномалій та перевірки того, як змінюється поведінка сенсора після чергового перенавчання. Як видно з рисунка 2.5, таблиця `sensors` задає контекст вимірювального каналу, `readings` накопичує часовий ряд сирих і скоригованих значень, а `calibration_models` зберігає параметри та якість кожної прийнятої моделі. SQL-представлення використовуються для підготовки агрегованих даних до вебпанелі й Grafana без дублювання логіки у фронтенді.

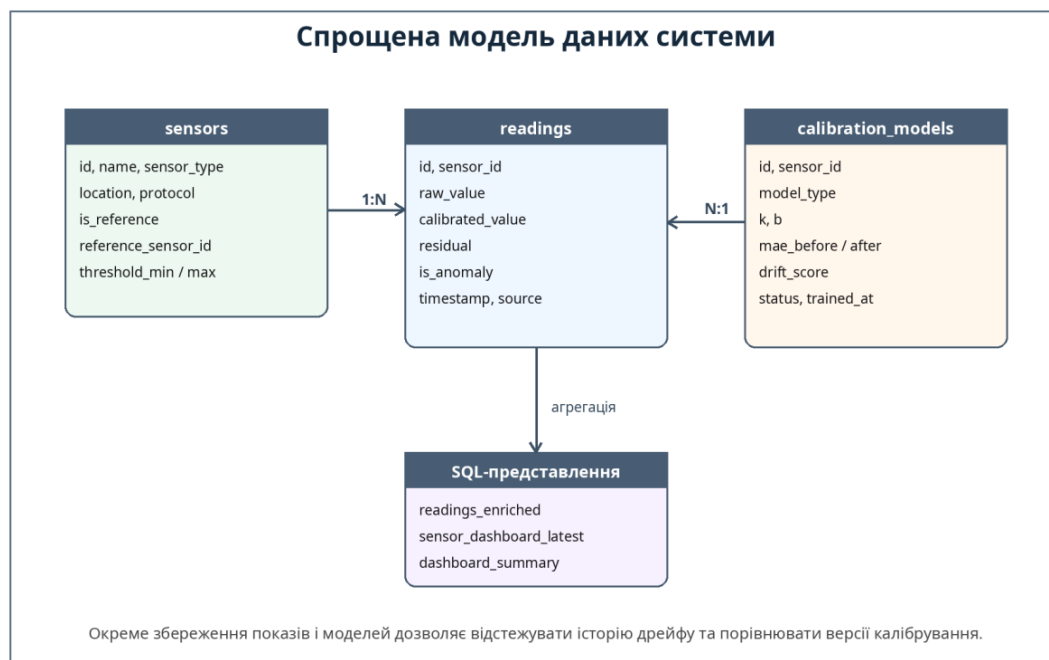


Рисунок 2.5 – Спрощена ER-схема збереження даних та моделей

Завдяки такій структурі база даних виконує роль не тільки сховища, а й джерела пояснюваності: для кожного скоригованого значення можна визначити,

яка модель була використана, наскільки вона покращувала похибку на момент навчання та чи були поруч аномальні події.

Практичну роль полів у моделі даних та місце їх використання відображено у таблиці 2.3.

Таблиця 2.3 – Практична роль основних полів у моделі даних

Поле або група полів	Де використовується	Практичне значення
sensor_type, location, protocol	sensors	Дають змогу фільтрувати канали за фізичним типом, кімнатою та способом надходження даних.
raw_value, calibrated_value, residual	readings	Дозволяють порівнювати вихідні й скориговані покази, а також оцінювати залишкову похибку.
k, b, mae_before, mae_after, status	calibration_models	Фіксують параметри перетворення та критерій, за яким модель була прийнята або відхилена.

2.7 Алгоритм самокалібрування

Основна ідея самокалібрування полягає в тому, що для кожного бюджетного сенсора система намагається знайти пов'язаний з ним еталонний сенсор того самого фізичного типу. Далі з історії вимірів формується вибірка пар значень, синхронізованих за часовими мітками з певною допустимою похибкою часу. На основі цих пар будується функція перетворення сирого значення бюджетного сенсора в оцінку, максимально наближену до показів еталонного каналу.

У прототипі для цього використано HuberRegressor. Вибір саме цієї моделі обґрунтований тим, що документація scikit-learn визначає HuberRegressor як робастну до викидів лінійну регресію, а в прикладах порівняння показано, що її прогнози менше спотворюються сильними викидами, ніж звичайна ridge-регресія. Для задачі побутових сенсорів це є важливою перевагою, оскільки історія вимірів часто містить шум, короточасні стрибки та разові помилки передачі.

Додатковим аргументом на користь такого підходу є сучасні дослідження низьковартісних сенсорів, у яких підкреслюється, що машинне навчання здатне істотно підвищувати надійність показів порівняно з необробленими даними, особливо коли сенсор піддається впливу середовища або тривало працює в польових умовах. Огляди самокалібрування для неконтрольованих середовищ та новіші роботи з ML-калібрування низьковартісних сенсорів підтверджують доцільність переходу від одноразового ручного налаштування до адаптивних алгоритмів корекції.

Практично навчання моделі виконується так. Спочатку обчислюється середня абсолютна похибка між сирими значеннями бюджетного сенсора та значеннями еталонного каналу. Після цього навчається робастна регресійна модель. Далі порівнюється якість до та після корекції. Якщо після застосування моделі середня абсолютна похибка помітно зменшується, модель приймається та зберігається в базі даних. Якщо ж покращення не досягнуто, система залишає ідентичне перетворення й позначає стан сенсора як стабільний або такий, що потребує уваги. Такий механізм запобігає ситуації, коли невдала модель погіршує якість даних. Поріг покращення у 2 % обрано з урахуванням того, що він перевищує приведену похибку сенсора відносно діапазону вимірювань – зокрема, для температурного каналу Aquira WSDCQ11LM з паспортною характеристикою $\pm 0,3$ °C та діапазоном 70 °C приведена похибка становить 0,43 %, що є меншою за встановлений поріг. Це забезпечує прийняття практично значущих коригувальних моделей і виключає випадкове покращення як підставу для оновлення моделі.

Для наочного опису цього процесу алгоритм самокалібрування можна подати як послідовність рішень (рис. 2.6), у якій навчання моделі саме по собі не є достатньою умовою її використання. Модель вважається робочою лише після порівняння метрик до та після корекції.

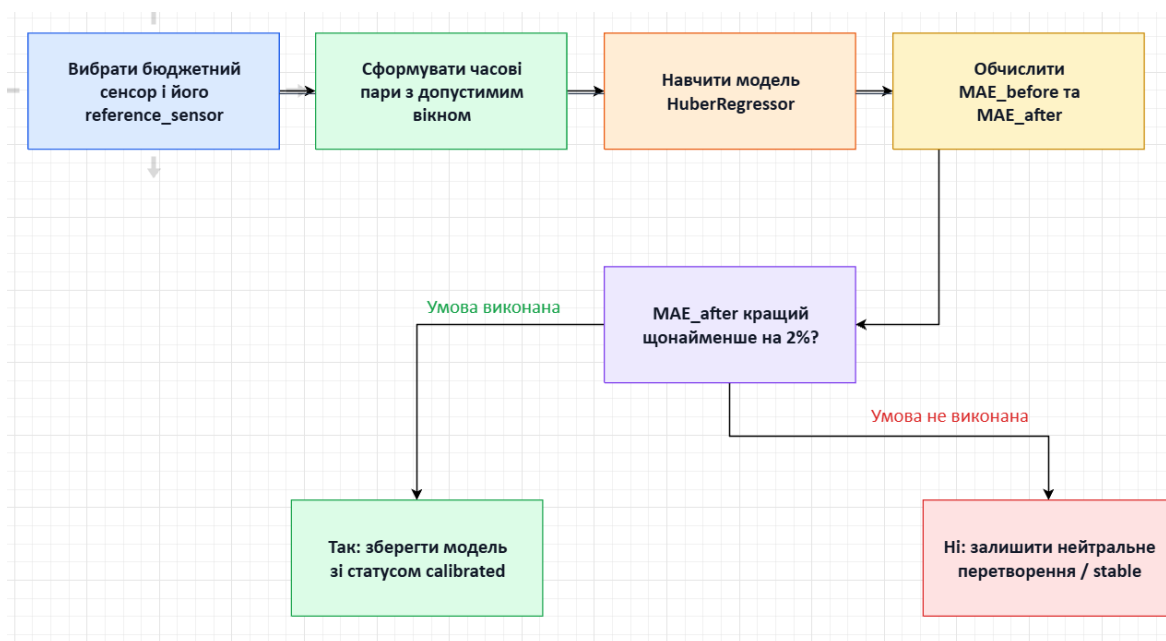


Рисунок 2.6 – Алгоритм навчання та прийняття калібрувальної моделі

У лістингу 2.2 відображено два захисні механізми: мінімальну кількість парних спостережень і поріг покращення за MAE. Завдяки цьому система не зберігає випадкову модель, побудовану на замалій або нерепрезентативній вибірці, а також не застосовує корекцію, яка не дає помітного практичного виграшу.

Лістинг 2.2 – Узагальнений фрагмент навчання та перевірки моделі

```

pairs = await repository.get_aligned_pairs(
    sensor.id,
    sensor.reference_sensor_id,
    limit=300,
)
if len(pairs) < 12:
    return neutral_model(sensor.id)

x, y = build_training_arrays(pairs)
mae_before = mean_absolute_error(y, x.ravel())

model = HuberRegressor().fit(x, y)
mae_after = mean_absolute_error(y, model.predict(x))

if mae_after <= mae_before * 0.98:
    return save_model(sensor.id, model, status="calibrated")
return neutral_model(sensor.id, status="stable")
  
```

2.8 Онлайн-виявлення аномалій

Окрім калібрування, система виконує онлайн-виявлення аномалій у потоці вимірів. Для цього використовується робастна статистична схема на основі медіани та MAD (median absolute deviation). Вибір саме цього підходу зумовлений тим, що сенсорні дані в системах «розумного дому» часто містять короткочасні сплески, шум, разові помилки передавання або різкі локальні зміни середовища. Методи, що спираються на середнє значення та стандартне відхилення, у таких умовах можуть давати нестійку оцінку, оскільки поодинокі викиди суттєво впливають на середнє. Натомість медіана та MAD є менш чутливими до окремих аномальних точок, тому краще відображають типовий стан сенсора за останній проміжок часу. Кожне нове скориговане значення порівнюється з недавньою історією того самого сенсора. Якщо відхилення перевищує встановлений поріг, значення позначається як аномальне.

Такий підхід обрано з двох причин. По-перше, він добре узгоджується з робастною природою основної калібрувальної моделі. По-друге, він є обчислювально легким і придатним для edge-сценаріїв. У сучасних роботах з edge anomaly detection для IoT підкреслюється, що практичне значення мають не лише точність, а й низькі витрати пам'яті, затримка та можливість локального запуску без постійної передачі даних у хмару. Саме тому для дипломного прототипу було обрано не важкі глибинні мережі, а ресурсозаощадний підхід до виявлення аномалій на периферії мережі (lightweight-підхід), достатній для домашньої сенсорної мережі.

Важливо підкреслити, що аномалія в цій системі не ототожнюється автоматично з поломкою сенсора. Вона інтерпретується як сигнал для додаткового аналізу. Причиною аномального виміру може бути як дефект або дрейф сенсора, так і реальна подія у приміщенні: різке провітрювання, локальне нагрівання, поява людей, короткий сплеск CO₂ або інший дійсний фізичний вплив. Тому результат аномалійного аналізу повинен розглядатися у зв'язку з контекстом системи.

2.9 Принцип роботи фронтенда та сценарій взаємодії з оператором

Розроблений вебінтерфейс виконує роль операторської панелі. На ньому користувач бачить загальну кількість сенсорів, обсяг накопичених вимірів, число аномалій за останню добу та кількість сенсорів із виявленим дрейфом. Окремо відображається таблиця сенсорів з останніми сирими та скоригованими значеннями, статусом моделі й ознакою аномалії. Для кожного сенсора можна побудувати графік історії, де порівнюються сирі(raw) і відкалібровані (calibrated) значення.

Через ту саму панель оператор може вручну додати новий показ, що є зручним для тестування та демонстрації системи на захисті. Також передбачено ручний запуск пакетного самокалібрування. Це дозволяє показати два режими роботи: автоматичний періодичний режим і ініційований користувачем режим. У результаті фронтенд не лише візуалізує дані, а й є засобом керування життєвим циклом моделі.

Операторський інтерфейс у роботі виконує не декоративну, а діагностичну функцію. Він має давати відповідь на три практичні питання: які сенсори зараз активні, чи є помітні відхилення, і чи спрацювала модель самокалібрування для конкретного каналу.

На рисунку 2.7 показано логіку компонування вебпанелі: у верхній частині розміщуються агреговані показники, у центральній – таблиця останніх станів сенсорів, а нижче – зона історії для вибраного каналу. Такий формат зручний для демонстрації на захисті, оскільки дає змогу одночасно показати і результат роботи алгоритму, і стан програмної системи.

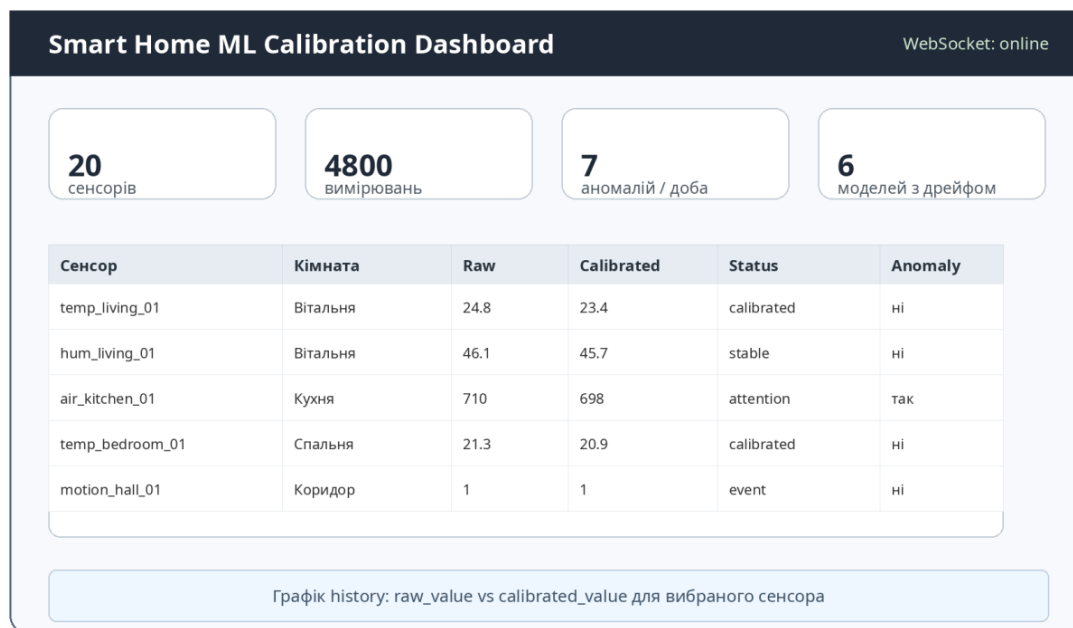


Рисунок 2.7 – Узагальнений вигляд операторської вебпанелі та панелі адміністратора

2.10 Висновки до розділу 2

У цьому розділі було запропоновано архітектуру програмної системи для самокалібрування сенсорів «розумного дому», яка поєднує локальний збір даних, їх збереження, побудову робастної ML-моделі та вебвізуалізацію результатів. Вибрана архітектура є достатньо гнучкою для подальшого

розширення. Вона не потребує перепрошивки шлюзу, може бути інтегрована з Home Assistant через стандартні інтерфейси обміну та забезпечує підвищення якості сенсорних даних локальними засобами.

3 ПОРІВНЯННЯ ЗАПРОПОНОВАНОЇ СИСТЕМИ З ІСНУЮЧИМИ ПІДХОДАМИ

3.1 Критерії порівняння

Для оцінювання запропонованого рішення доцільно використовувати такі критерії: наявність відкритих інтерфейсів інтеграції, можливість локального розгортання, наявність механізмів самокалібрування, наявність засобів виявлення аномалій, складність впровадження та придатність до роботи з бюджетними сенсорами. Саме ці характеристики визначають практичну цінність системи в умовах реального «розумного дому», де важливі не лише теоретична точність, а й простота інтеграції та експлуатації.

У межах цього порівняння доцільно розглядати три умовні класи рішень: типову платформу «розумного дому» без вбудованої ML-корекції, відкриту інтеграційну платформу Home Assistant без окремого модуля самокалібрування та запропоновану в дипломній роботі систему з зовнішнім Python-сервісом.

Для стислого узагальнення відмінностей між підходами сформовано порівняльну таблицю (табл. 3.1), у якій акцент зроблено не на маркетингових характеристиках платформ, а на їхній придатності саме до задачі підтримання достовірності сенсорних даних.

Таблиця 3.1 – Порівняння підходів до роботи з сенсорними даними

Підхід	Сильні сторони	Обмеження в контексті задачі
Типова закрита smart-home платформа	Швидке підключення пристроїв, готовий мобільний застосунок, прості сценарії автоматизації.	Зазвичай не має відкритого механізму ML-корекції, не пояснює якість даних.
Home Assistant без зовнішнього ML-модуля	Відкриті інтеграції, MQTT, REST, WebSocket, підтримка багатьох пристроїв.	Добре збирає стани, але потребує окремої надбудови для самокалібрування та оцінювання похибок.
Традиційне ручне калібрування	Метрологічно зрозуміле, може виконуватися відносно еталонного приладу.	Погано масштабується, потребує участі користувача та не компенсує дрейф автоматично.

Продовження таблиці 3.1

Підхід	Сильні сторони	Обмеження в контексті задачі
Запропонована система	Локальна ML-корекція, журналювання моделей, онлайн-детекція аномалій, робота без перепрошивки шлюзу.	Потребує історії даних і бажаної наявності reference-сенсора для якісного навчання.

З таблиці 3.1 видно, що дипломне рішення не конкурує напряму з Home Assistant або типовими мобільними екосистемами, а доповнює їх функцією контролю якості даних. Саме це робить запропонований підхід практично корисним для бюджетного «розумного дому».

3.2 Порівняння з типовими платформами «розумного дому»

Більшість побутових платформ орієнтується насамперед на підключення пристроїв, відображення станів і виконання автоматизацій. У такому класі систем сенсор розглядається як джерело поточного значення, а не як вимірювальний канал, що потребує динамічної оцінки похибки. Через це проблема дрейфу, накопичення систематичної помилки та статистичного контролю якості даних зазвичай не розв'язується на рівні базової екосистеми.

Порівняно з таким підходом запропонована система вводить окремий шар контролю достовірності даних. Вона не замінює базову платформу автоматизації, а доповнює її. Саме в цьому полягає одна з головних переваг: користувач зберігає звичну інфраструктуру «розумного дому», але отримує поверх неї інструмент самокалібрування та аналітики.

3.3 Порівняння з Home Assistant як інтеграційною платформою без ML-модуля

Home Assistant є значно гнучкішою платформою, ніж типові закриті побутові екосистеми. Документація Home Assistant підтверджує наявність MQTT-інтеграції, REST API, WebSocket API та підтримку Xiaomi Gateway, що

робить платформу сильною основою для збирання телеметрії, побудови сценаріїв автоматизації й інтеграції різнорідних пристроїв. Однак у доступній документації основний акцент робиться саме на інтеграції, отриманні станів і керуванні пристроями; окремий вбудований механізм ML-самокалібрування сенсорів у межах цих інтерфейсів не описаний. Це дозволяє зробити обґрунтований висновок, що Home Assistant є чудовим інтеграційним середовищем, але шар інтелектуальної корекції даних доцільно реалізовувати зовні.

Порівняно з «чистим» Home Assistant, запропоноване рішення додає три нові можливості. По-перше, воно автоматично оцінює ступінь розходження між бюджетним та еталонним сенсором. По-друге, воно навчає й застосовує коригувальну модель без участі користувача. По-третє, воно виконує онлайн-виявлення аномалій та відображає результати у спеціалізованій вебпанелі. Таким чином, Home Assistant у цій архітектурі доцільно розглядати як інтеграційний фундамент, а розроблений сервіс – як аналітичну надбудову над ним.

3.4 Порівняння з традиційним ручним калібруванням

Традиційне ручне калібрування має важливу перевагу - високу метрологічну прозорість. Користувач або сервісний інженер працює з контрольованим еталоном, явно визначає поправку та може документально зафіксувати процедуру. Проте для побутової сенсорної мережі цей підхід має суттєві обмеження: він трудомісткий, не масштабується при великій кількості пристроїв і не враховує поступову зміну характеристик сенсора вже після завершення процедури калібрування. Огляди self-calibration для неконтрольованих середовищ і сучасні роботи з низьковартісними сенсорами підтверджують, що саме польове або адаптивне перекалібрування є ключем до тривалої підтримки якості даних.

У запропонованій системі ручне калібрування не скасовується як ідея, але його роль змінюється. Замість того щоб бути єдиним інструментом підтримання

точності, воно може використовуватися як початковий етап або як контрольна процедура, тоді як повсякденна корекція виконується автоматично за даними референсного сенсора та накопиченої історії.

3.5 Переваги запропонованого рішення

Головною перевагою розробленої системи є її збалансованість між складністю та практичністю. Вона не вимагає важкої обчислювальної інфраструктури, але все ж використовує машинне навчання там, де воно реально покращує результат. Робастна лінійна модель є достатньо інтерпретованою, щоб пояснити користувачеві, як саме перетворюється вимір, і водночас достатньо стійкою до разових викидів у навчальній вибірці. Це добре узгоджується з рекомендаціями використовувати ML-калібрування для low-cost sensor networks та робастні або lightweight-підходи для edge-сценаріїв.

Другою перевагою є відсутність необхідності змінювати прошивку сенсорів чи шлюзу. Для дипломного проєкту це критично, оскільки значно зменшує ризики під час впровадження та робить архітектуру універсальнішою. Якщо в майбутньому зміниться конкретний набір датчиків, логіка самокалібрування може залишитися майже незмінною – достатньо переналаштувати транспортний шар і мапування сенсорів.

Третьою перевагою є поєднання калібрування та виявлення аномалій в одному циклі. У багатьох практичних системах ці задачі розділяються, через що оператор змушений користуватися кількома інструментами одночасно. У запропонованому рішенні вебпанель одразу показує і скориговані значення, і ознаку аномалії, і статус моделі, що спрощує аналіз стану системи.

3.6 Обмеження та напрями подальшого розвитку

Запропонована модель також має обмеження. По-перше, вона передбачає наявність еталонного або умовно еталонного сенсора, з яким можна порівнювати

бюджетний канал. Якщо такого сенсора немає, можливості корекції стають обмеженішими, а система переходить до більш слабких статистичних механізмів оцінювання. По-друге, базова реалізація використовує лінійну робастну модель, тоді як для деяких типів сенсорів залежність між сирими та коректними значеннями може бути нелінійною. По-третє, якість моделі залежить від накопиченої історії та від правильності синхронізації часових рядів.

Подальший розвиток системи може рухатися в кількох напрямках. Перший напрям – розширення набору моделей, наприклад додавання градієнтного бустингу, рекурентних моделей або окремих моделей для різних режимів середовища. Другий напрям - автоматичне врахування зовнішніх факторів, таких як час доби, сезонність або стан вентиляції. Третій напрям - тісніша інтеграція з Home Assistant через готові сутності та автогенерацію MQTT discovery-повідомлень. Література з ML-калібрування низьковартісних сенсорів показує, що включення часового контексту та додаткових зовнішніх ознак може покращувати якість корекції.

3.7 Висновки до розділу 3

Порівняльний аналіз показує, що запропонована система займає проміжне, але дуже практичне положення між простими платформами домашньої автоматизації та складними спеціалізованими вимірювальними комплексами. Від типових smart-home платформ вона успадковує простоту інтеграції та локального використання. Від сучасних підходів до сенсорних мереж вона бере ідею адаптивного самокалібрування й контролю якості даних. У результаті формується рішення, яке є достатньо простим для дипломного проекту і водночас достатньо змістовним з науково-прикладної точки зору.

4 ОЦІНЮВАННЯ ПОХИБОК РЕЗУЛЬТАТІВ ВИМІРЮВАНЬ ТА ЕФЕКТИВНОСТІ АЛГОРИТМУ САМОКАЛІБРУВАННЯ

4.1 Показники оцінювання похибок результатів вимірювань

Для перевірки ефективності запропонованої системи самокалібрування доцільно використовувати сукупність показників, які характеризують як відхилення окремих вимірювань від еталона, так і узагальнений ефект корекції на вибірці. У роботі як базові метрики застосовано абсолютну похибку, відносну похибку, середню абсолютну похибку MAE, середньоквадратичну похибку RMSE та коефіцієнт покращення після самокалібрування. Такий вибір є виправданим для задачі порівняння «сирих» і скоригованих значень відносно еталонного каналу, оскільки MAE добре інтерпретується в одиницях вимірюваної величини, а RMSE додатково підкреслює вплив великих відхилень.

Абсолютна похибка окремого вимірювання визначається як різниця між еталонним і оціненим значенням:

$$\Delta_i = y_i - \hat{y}_i. \quad (4.1)$$

Модуль абсолютної похибки записується у вигляді:

$$|\Delta_i| = |y_i - \hat{y}_i|. \quad (4.2)$$

Для інтерпретації похибки у відсотках використовується відносна похибка:

$$\delta_i = \frac{|y_i - \hat{y}_i|}{|y_i|} \cdot 100 \%. \quad (4.3)$$

Середня абсолютна похибка на вибірці з n спостережень обчислюється за формулою:

$$MAE = \frac{1}{n} \cdot \sum |y_i - \hat{y}_i|. \quad (4.4)$$

Середньоквадратична похибка визначається як:

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum (y_i - \hat{y}_i)^2}. \quad (4.5)$$

Для кількісної оцінки ефективності алгоритму самокалібрування використовується відносне покращення за MAE:

$$I = \frac{(MAE_before - MAE_after)}{MAE_before} \cdot 100 \%. \quad (4.6)$$

Коефіцієнт I є кількісним показником ефективності алгоритму самокалібрування і відображає відносне зменшення похибки результатів вимірювань після застосування коригувальної моделі.

Для оцінювання відхилення результатів вимірювань відносно діапазону вимірювань сенсора використовується приведена похибка, що розраховується за формулою

$$\gamma = \frac{|\Delta_i|}{X_{\text{dian}}} \cdot 100 \%, \quad (4.7)$$

де X_{dian} – діапазон вимірювань відповідного сенсора. Приведена похибка дозволяє коректно порівнювати відхилення між вимірювальними каналами різних типів, оскільки приводить похибку до єдиної відносної шкали.

Такий підхід дає змогу уникнути прийняття моделі, яка не забезпечує практично значущого зменшення похибки результатів вимірювань, що узгоджується з метрологічними вимогами до засобів вимірювальної техніки.

4.2 Розрахунок похибок у симульованому сценарії

Основна кількісна перевірка виконувалася на вбудованому симуляторі, який моделює чотири кімнати, 20 сенсорів і 4800 вимірювань. На етапі накопичення історії система формує 180 початкових пакетів даних, після чого виконується первинне batch-калібрування. Далі робота продовжується в online-режимі з періодичним перенавчанням моделей. Така методика дає змогу окремо оцінити ефект початкового налаштування і подальшу стійкість самокалібрування в потоці нових вимірювань. Основні показники похибки до і після самокалібрування у симульованому сценарії наведені у таблиці 4.1.

Таблиця 4.1 – Основні показники похибки до і після самокалібрування у симульованому сценарії

Сценарій	Група каналів	MAE до	MAE після	Покращення, %
Первинне калібрування	Температура	1,56 °C	0,31 °C	80,1
Первинне калібрування	Вологість	1,44 %RH	1,34 %RH	6,9
Первинне калібрування	Усі парні канали	1,50	0,83	44,7
Завершальне online-вікно	Температура	1,87 °C	1,08 °C	42,3
Завершальне online-вікно	Вологість	1,39 %RH	1,19у %RH	14,4

Для оцінювання відхилення результату вимірювань відносно діапазону вимірювань використано приведену похибку γ .

Для температурного каналу сенсора Aqara WSDCGQ11LM з діапазоном вимірювань від $-20\text{ }^{\circ}\text{C}$ до $+50\text{ }^{\circ}\text{C}$ ($X_{\text{dian}} = 70\text{ }^{\circ}\text{C}$):

$$\gamma_{\text{до}} = 1,56/70 \cdot 100\% = 2,23\%,$$

$$\gamma_{\text{після}} = 0,31/70 \cdot 100\% = 0,44\%.$$

Для вологісного каналу з діапазоном вимірювань від $0\text{ \%RH}$ до $+100\text{ \%RH}$ ($X_{\text{dian}} = 100\text{ \%RH}$):

$$\gamma_{\text{до}} = 1,44/100 \cdot 100\% = 1,44\%,$$

$$\gamma_{\text{після}} = 1,34/100 \cdot 100\% = 1,34\%.$$

У таблиці 4.2 наведені показники похибки результатів вимірювань та ефективності алгоритму самокалібрування для симульованого сценарію.

Таблиця 4.2 – Показники похибки результатів вимірювань та ефективності алгоритму самокалібрування для симульованого сценарію

Сценарій	Група каналів	MAE до	MAE після	Покращення, %	$\gamma_{\text{до}}, \%$	$\gamma_{\text{після}}, \%$
Первинне калібрування	Температура	1,56 $^{\circ}\text{C}$	0,31 $^{\circ}\text{C}$	80,1	2,23	0,44
Первинне калібрування	Вологість	1,44 $\text{\%RH}$	1,34 $\text{\%RH}$	6,9	1,44	1,34
Первинне калібрування	Усі парні канали	1,50	0,83	44,7	—	—
Завершальне online-вікно	Температура	1,87 $^{\circ}\text{C}$	1,08 $^{\circ}\text{C}$	42,3	2,67	1,54
Завершальне online-вікно	Вологість	1,39 $\text{\%RH}$	1,19 $\text{\%RH}$	14,4	1,39	1,19

Наведені результати показують, що найбільша ефективність самокалібрування досягається для температурних каналів, де дрейф у симуляторі має переважно афінний характер і добре описується робастною лінійною моделлю. Для каналів вологості покращення є помітно меншим, що свідчить про більшу роль випадкових збурень і неафінних компонент похибки. У середньому в завершальному online-вікні зниження MAE по всіх парних каналах становить 30,4 %, тобто алгоритм самокалібрування зберігає практичну ефективність і після серії перенавчань.

З інженерної точки зору це означає, що запропонований підхід особливо придатний для тих вимірювальних каналів, у яких основну частку похибки формують зсув нуля та зміна масштабу. Саме такі умови часто виникають у бюджетних температурних сенсорах, встановлених у житлових приміщеннях.

Для наочного порівняння результатів таблиці 4.1 доцільно подати їх також у графічній формі. На рисунку 4.1 видно не лише абсолютне зменшення MAE, а й відмінність ефекту для температурних та вологісних каналів.

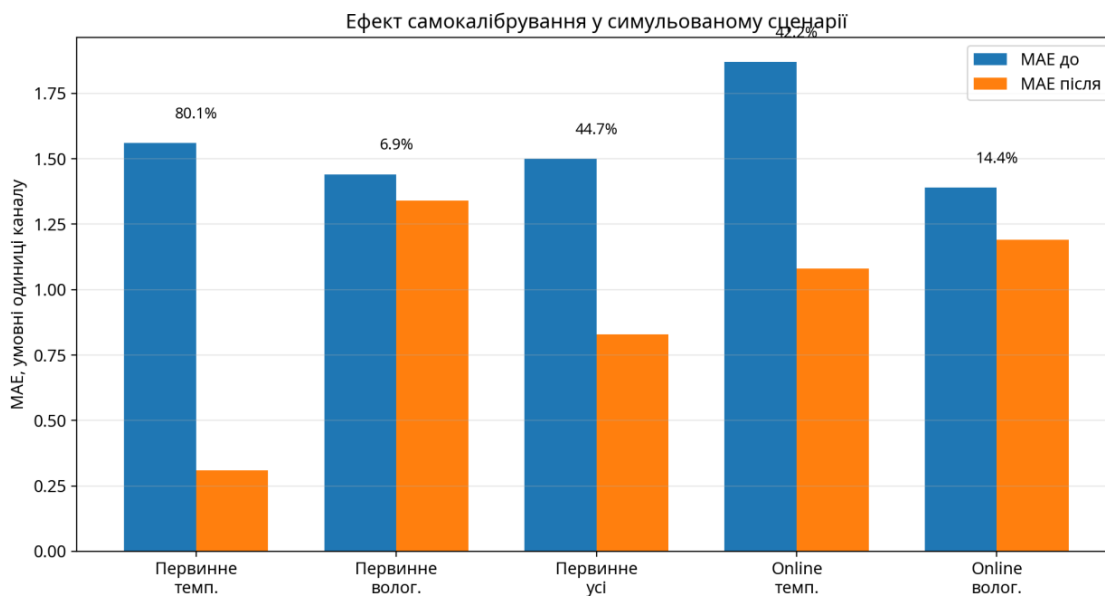


Рисунок 4.1 – Порівняння MAE до і після самокалібрування у симульованому сценарії

Рисунок 4.1 підтверджує, що найбільше відносне покращення припадає на температурні канали, де модель фактично компенсує систематичний зсув і масштабування. Для вологості зменшення похибки є менш різким, що узгоджується з більшою випадковою складовою таких вимірювань у симуляторі.

4.3. Оцінювання ефективності алгоритму самокалібрування на архівному сценарії

Для додаткової перевірки працездатності алгоритму було проаналізовано архівний набір даних проєкту, у якому збережено історію вимірювань та версій калібрувальних моделей. Така перевірка є важливою, оскільки доповнює синтетичний симулятор сценарієм, сформованим на основі реальної програмної експлуатації сервісу. Порівняння останніх моделей показує, що температурний канал у вітальні демонструє значний потенціал корекції, тоді як для каналу вологості ефект виявляється помірним.

Для температурного каналу вітальні з діапазоном вимірювань від $-20\text{ }^{\circ}\text{C}$ до $+50\text{ }^{\circ}\text{C}$ ($X_{\text{dian}} = 70\text{ }^{\circ}\text{C}$):

$$\gamma_{\text{до}} = 1,789/70 \cdot 100\% = 2,56\%,$$

$$\gamma_{\text{після}} = 0,162/70 \cdot 100\% = 0,23\%.$$

Для вологісного каналу вітальні з діапазоном вимірювань від $0\text{ \%RH}$ до $+100\text{ \%RH}$ ($X_{\text{dian}} = 100\text{ \%RH}$):

$$\gamma_{\text{до}} = 0,611/100 \cdot 100\% = 0,61\%,$$

$$\gamma_{\text{після}} = 0,588/100 \cdot 100\% = 0,59\%.$$

У таблиці 4.3 наведені показники похибки результатів вимірювань та ефективності алгоритму самокалібрування для архівного сценарію.

Таблиця 4.3 – Показники похибки результатів вимірювань та ефективності алгоритму самокалібрування для архівного сценарію

Канал	MAE до	MAE після	Покращення, %
Вітальня / температура	1,789	0,162	90,9
Вітальня / вологість	0,611	0,588	3,8

Отримані значення узгоджуються з результатами симульованого експерименту: для температурного датчика корекція істотно зменшує середню похибку, тоді як для вологості канал уже на початку ближчий до стабільного режиму і тому простір для покращення менший. Таким чином, архівний сценарій підтверджує, що самокалібрування доцільно інтерпретувати не як універсальне «посилення» всіх каналів, а як адаптивний механізм, який дає найбільшу користь саме там, де систематичний дрейф є достатньо вираженим.

Графічне подання архівного сценарію додатково підкреслює різницю між каналами: температурний канал має великий запас для корекції, тоді як канал вологості уже перебуває ближче до стабільного режиму.

На рисунку 4.2 видно, що для температури архівний набір демонструє суттєве зменшення похибки після застосування моделі, тоді як для вологості ефект є помірним.

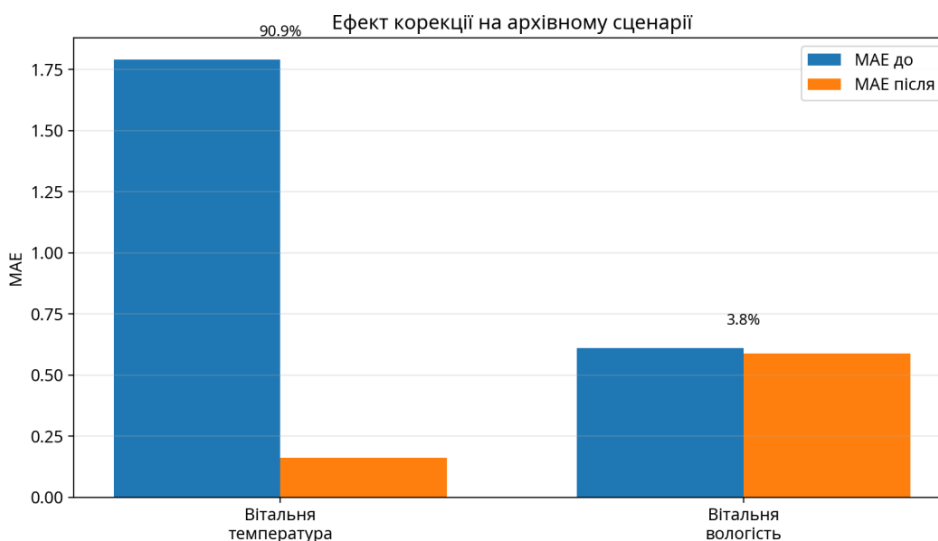


Рисунок 4.2 – Порівняння MAE до і після корекції для архівного сценарію

Це не суперечить загальній логіці роботи системи: алгоритм не повинен штучно «покращувати» стабільний канал, якщо статистично значущого виграшу майже немає.

4.4. Обмеження оцінювання та інтерпретація результатів

Подані розрахунки похибок характеризують насамперед ефективність програмної корекції в межах побудованого прототипу. Вони не є повноцінною метрологічною атестацією приладу у стандартизованому значенні, оскільки основний масив даних отримано на симуляторі та в архівному сценарії програмної системи, а не на сертифікованому калібрувальному стенді. Тому наведені значення MAE та RMSE слід трактувати як показники прикладної достовірності телеметрії для задач автоматизації, а не як формально затверджений клас точності.

Разом з тим саме такий спосіб оцінювання є достатнім для дипломного проєкту, метою якого є обґрунтування архітектури, алгоритму та прикладної корисності системи. Подальше поглиблення дослідження доцільно пов'язати з довготривалими польовими експериментами у різних приміщеннях, використанням сертифікованих еталонних сенсорів, а також побудовою повнішого бюджету невизначеності з урахуванням часової нестабільності, монтажних умов та впливу навколишнього середовища.

4.5. Висновки до розділу 4

У розділі сформовано систему показників для кількісного оцінювання похибок результатів вимірювань, що включає абсолютну похибку, MAE, RMSE, коефіцієнт покращення I та приведену похибку γ відносно діапазону вимірювань сенсора. Розрахунки виконано для симульованого та архівного сценаріїв. Встановлено, що алгоритм самокалібрування є найбільш дієвим для каналів з вираженою систематичною складовою похибки – зокрема для

температурних каналів приведена похибка γ після корекції не перевищує 0,44 % від діапазону вимірювань. Для вологісних каналів з переважно випадковою складовою похибки простір для корекції є більш обмеженим, що підтверджується меншим значенням коефіцієнта покращення I .

ВИСНОВКИ

У дипломній роботі розглянуто проблему накопичення похибки результатів вимірювань внаслідок дрейфу характеристик сенсорних вузлів в інформаційно-вимірювальних системах моніторингу параметрів навколишнього середовища на базі технологій «розумного дому» і показано, що для бюджетних сенсорних вузлів, які не мають вбудованих засобів калібрування, це явище в умовах тривалої експлуатації безпосередньо впливає на якість вимірювальної інформації, що є основою для прийняття автоматизованих рішень, якість автоматизації, енергетичну ефективність і зручність користування домашньою інфраструктурою.

На основі аналізу існуючих підходів до калібрування сенсорних вузлів та функціональних обмежень наявних платформ «розумного дому» щодо збору, оброблення та передавання вимірювальної інформації запропоновано архітектуру інформаційно-вимірювальної системи моніторингу параметрів навколишнього середовища на базі сенсорів Aqara, шлюзу Xiaomi Gateway 3 та MQTT-протоколу, в якій модуль самокалібрування реалізовано як зовнішній програмний сервіс на основі FastAPI з використанням бази даних PostgreSQL, що інтегрується з наявною сенсорною інфраструктурою через стандартні інтерфейси обміну даними без модифікації прошивки вимірювальних вузлів.

У роботі обґрунтовано доцільність застосування методів машинного навчання, зокрема робастної регресії, для корекції результатів вимірювань та самокалібрування сенсорних вузлів. Показано, що на відміну від традиційного ручного калібрування, яке не масштабується при збільшенні кількості вимірювальних каналів і не компенсує дрейф автоматично, запропонований підхід забезпечує автоматичну компенсацію дрейфу похибки вимірювальних каналів шляхом періодичного оновлення коригувальної моделі без участі користувача.

Розроблено алгоритм самокалібрування сенсорних вузлів, що формує часово узгоджені пари показів «бюджетного» та еталонного сенсорів, навчає коригувальну модель на основі HuberRegressor та приймає її лише за умови

зменшення похибки результатів вимірювань не менше ніж на 2 % відносно паспортної приведеної похибки сенсора, що запобігає збільшенню похибки результатів вимірювань внаслідок застосування моделі, яка не пройшла перевірку за критерієм MAE. Додатково реалізовано онлайн-виявлення аномалій у показах вимірювальних каналів на основі медіани та MAD, що забезпечує виявлення аномальних значень без чутливості до поодиноких викидів у потоці вимірювальної інформації.

Виконаний розрахунок похибок показав, що самокалібрування є особливо ефективним для температурних каналів: у симульованому сценарії та на архівному наборі даних спостерігається істотне зменшення MAE після корекції. Для каналів вологості ефект є помірнішим, але все одно підтверджує доцільність використання адаптивної корекції замість повної довіри до «сирих» показів сенсорів.

Оцінено ефективність запропонованого алгоритму самокалібрування шляхом аналізу зміни похибки результатів вимірювань у симульованому та архівному сценаріях. Для температурних каналів коефіцієнт покращення I становить 80,1 % – у симульованому сценарії та 90,9 % в архівному, а приведена похибка γ зменшується з 2,23 % до 0,44 % від діапазону вимірювань. Для каналів вологості ефект є помірнішим – коефіцієнт покращення становить 6,9 % та 3,8 % відповідно, що пояснюється більшою випадковою складовою похибки цього типу сенсорів. Отримані результати підтверджують доцільність застосування запропонованого алгоритму самокалібрування для зменшення похибки результатів вимірювань у каналах з вираженою систематичною складовою похибки.

Порівняльний аналіз розробленої системи з існуючими платформами показав, що типові платформи «розумного дому» та Home Assistant забезпечують збір і передавання вимірювальної інформації, але не мають вбудованого механізму автоматичної компенсації дрейфу похибки сенсорних вузлів. Традиційне ручне калібрування, хоча і є метрологічно прозорим, не масштабується при збільшенні кількості вимірювальних каналів і не забезпечує

автоматичної компенсації дрейфу в умовах тривалої експлуатації. Розроблена система займає проміжне положення між цими підходами, доповнюючи існуючу сенсорну інфраструктуру модулем самокалібрування без необхідності її заміни.

Практичне значення роботи полягає в тому, що запропоновану систему можна розгорнути без перепрошивки сенсорів або шлюзу, використовуючи вже наявні побутові пристрої та локальний програмний сервіс. Це робить рішення доступним для недорогих домашніх систем автоматизації та водночас створює основу для подальшого переходу до складніших багатовимірних і контекстно-адаптивних моделей.

Результати роботи можуть бути використані при розробці інформаційно-вимірювальних систем моніторингу параметрів навколишнього середовища з автоматичною компенсацією дрейфу похибки сенсорних вузлів. Відсутність необхідності модифікації прошивки вимірювальних вузлів та інтеграція через стандартні інтерфейси обміну даними забезпечують можливість впровадження розробленого програмного сервісу в системах побутового екологічного моніторингу, а також створюють основу для подальшого розвитку в напрямі багатоканальних інформаційно-вимірювальних систем.

Окремі результати дослідження апробовано в авторських наукових публікаціях [12, 13], що підтверджує наукову та прикладну цінність отриманих результатів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Yamamoto K., Togami T., Yamaguchi N., Ninomiya S. Machine Learning-Based Calibration of Low-Cost Air Temperature Sensors Using Environmental Data. *Sensors*. 2017. Vol. 17, No. 6, Article 1290. DOI: <https://doi.org/10.3390/s17061290>.
2. Cecchi S., Tinarelli A., Avvenuti A., Baldini S., Bianchi F. Self-calibration methods for uncontrolled environments in sensor networks: A reference survey. *Ad Hoc Networks*. 2019. Vol. 88. P. 142–159. DOI: <https://doi.org/10.1016/j.adhoc.2019.01.008>.
3. Rivera J., Carrillo M., Chacón M., Herrera G., Bojorquez G. Self-Calibration and Optimal Response in Intelligent Sensors Design Based on Artificial Neural Networks. *Sensors*. 2007. Vol. 7, No. 8. P. 1509–1529. DOI: <https://doi.org/10.3390/s7081509>.
4. Dubey R., Telles A., Nikkel J. et al. Low-Cost CO₂ NDIR Sensors: Performance Evaluation and Calibration Using Machine Learning Techniques. *Sensors*. 2024. Vol. 24, No. 17, Article 5675. DOI: <https://doi.org/10.3390/s24175675>.
5. Taştan M. Machine Learning-Based Calibration and Performance Evaluation of Low-Cost Internet of Things Air Quality Sensors. *Sensors*. 2025. Vol. 25, No. 10, Article 3183. DOI: <https://doi.org/10.3390/s25103183>.
6. DeMedeiros K., Hendawi A., Alvarez M. A Survey of AI-Based Anomaly Detection in IoT and Sensor Networks. *Sensors*. 2023. Vol. 23, No. 3, Article 1352. DOI: <https://doi.org/10.3390/s23031352>.
7. Liu B., Zhou Y., Fu H., Fu P., Feng L. Lightweight Self-Detection and Self-Calibration Strategy for MEMS Gas Sensor Arrays. *Sensors*. 2022. Vol. 22, No. 12, Article 4315. DOI: <https://doi.org/10.3390/s22124315>.
8. Home Assistant. MQTT integration documentation. URL: <https://www.home-assistant.io/integrations/mqtt/> (Accessed 14.02.2026).
9. Home Assistant Developer Docs. WebSocket API. URL: <https://developers.home-assistant.io/docs/api/websocket/> (Accessed 20.02.2026).

10. FastAPI. Official documentation. URL: <https://fastapi.tiangolo.com/> (Accessed 17.03.2026).
11. Home Assistant. Xiaomi Home integration documentation. URL: https://www.home-assistant.io/integrations/xiaomi_miio/ (Accessed 25.03.2026).
12. Бойко О. В., Добролюбова М. В. Система самокалібрування сенсорів розумного будинку на основі робастної регресії та онлайн-детекції аномалій. Рукопис наукової статті, поданий у *Вісник Вінницького політехнічного інституту*, 2026 (знаходиться у черзі на опублікування).
13. Бойко О. В., Добролюбова М. В. Система «розумний дім» з алгоритмом самокалібрування на основі методів машинного навчання // Ефективність та автоматизація інженерних рішень у приладобудуванні: збірник доповідей XXI Науково-практичної конференції студентів, аспірантів та молодих вчених, м. Київ, Україна, 10-11 грудня 2025 р. Київ, 2025. С 192-195. URL: https://drive.google.com/file/d/17dY6RNMIbE_t8P1qiUHV5lbfRilbQZPD/view?usp=drive_link, <https://ela.kpi.ua/server/api/core/bitstreams/a737ec0e-fcbf-4d50-8776-29e8e7ec0e72/content>