БАКАЛАВРСЬКА РОБОТА

КРБ.СІ-05.00.00.000 ПЗ

Група СІ-22-1

Данило ЛАБАЙ

2026

Івано-Франківський національний технічний університет нафти і газу
Факультет Інформаційних Технологій
Кафедра інформаційно-телекомунікаційних технологій та систем

Лабай Данило Назарійович
(прізвище, ім’я, по батькові)

УДК 681.58+004.43
(індекс)

БАКАЛАВРСЬКА РОБОТА

Розроблення апаратно-програмних засобів розподіленої IoT-системи з надлишковими вимірюваннями для виявлення відмов сенсорів та відновлення їх даних
(назва роботи)

Системна інженерія - Інтернет речей
(назва освітньої програми)

151 - Автоматизація та комп'ютерно-інтегровані технології
(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня _____ Лабай Д. Н.
(підпис, ініціали та прізвище здобувача)

Науковий Керівник _____ Волинський Орест Ігорович, доц.,к.т.н.
(підпис, прізвище, ім’я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
В. о. завідувача кафедри
_____ ***Заміховська О. Л.***
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ - 2026

Івано-Франківський національний технічний університет нафти і газу

(повне найменування закладу вищої освіти)

Факультет *Інформаційних технологій*

Кафедра *Інформаційно-телекомунікаційних технологій і систем*

Освітній рівень бакалавр

Спеціальність 151 - Автоматизація та комп'ютерно-інтегровані технології

(шифр і назва)

ЗАТВЕРДЖУЮ

В. о. Завідувач кафедри ІТТС д.т.н., проф.

_____ О.Л.Заміховська

«_____» _____ 2026 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Лабаю Данилу Назарійовичу

(прізвище, ім’я, по-батькові)

1. Тема роботи Розроблення апаратно-програмних засобів розподіленої IoT-системи з надлишковими вимірюваннями для виявлення відмов сенсорів та відновлення їх даних

Керівник роботи Волинський Орест Ігорович, доц., к.т.н.

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом закладу вищої освіти від ‘‘__’’ _____ 2026 року № ____

2. Строк подання студентом роботи «11» червня 2026 р.

3. Вхідні дані до роботи Мікроконтролер ESP32, датчики температури та параметрів мікроклімату, АЦП ADS1115, бездротовий зв'язок Wi-Fi, протокол MQTT, серверний застосунок на Python, база даних SQLite.

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

1. Аналіз відмов у розподілених IoT-системах.
2. Аналіз методів виявлення аномалій та відновлення даних.
3. Розроблення структури та апаратної частини системи.
4. Розроблення програмного забезпечення вимірювального вузла.
5. Реалізація серверного застосунку та бази даних.
6. Розроблення алгоритмів діагностики відмов і реконструкції даних.
7. Тестування та аналіз результатів роботи системи.

Перелік графічного матеріалу (з точним значенням обов’язкових креслень)

Структурна схема IoT-системи; Структурна схема вимірювального вузла; Принципова електрична схема вузла; Схема мережевої взаємодії MQTT; Алгоритм роботи ESP32; Структура бази даних; Алгоритм виявлення відмов та відновлення даних; Вигляд веб-дашборда системи.

6 .Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	Волинський О. І.		

7. Дата видачі завдання *19.01.2026р.*

Керівник_____

Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Термін виконання етапів роботи	Примітка
1	Аналіз типових апаратних та програмних відмов у розподілених системах моніторингу мікроклімату	01.04.2026-10.04.2026	Виконано
2	Обґрунтування вибору протоколу зв'язку (Wi-Fi/MQTT) та концепції аналітичної надлишковості вимірювань	11.04.2026-20.04.2026	Виконано
3	Розробка структурної схеми мультипараметричного вимірювального вузла на базі ESP32 та АЦП ADS1115	21.04.2026-30.04.2026	Виконано
4	Практична реалізація апаратної частини: підключення датчиків (BMP390, DHT22, BH1750FVI, MH-Z19C) та вимірювального тракту температури	01.05.2026-12.05.2026	Виконано
5	Розроблення firmware для ESP32 (двоядерна архітектура FreeRTOS, NTP-синхронізація, MQTT-публікація)	13.05.2026-22.05.2026	Виконано
6	Створення серверного застосунку на Python (прийом даних, SQLite, трирівнева діагностика відмов)	23.05.2026-05.06.2026	Виконано
7	Розробка модулів реконструкції даних (кригінг, медіанна фільтрація) та веб-дашборду (Plotly Dash)	06.06.2026-10.06.2026	Виконано
8	Тестування системи, аналіз точності відновлення даних та оформлення роботи	11.06.2026	Виконано

Студент_____ Лабай Д. Н.
(підпис) (прізвище та ініціали)

Керівник роботи_____Волинський О.І.
(підпис) (прізвище та ініціали)-

РЕФЕРАТ

Бакалаврська робота складається з 3 розділів, 13 рисунків, 9 таблиць, загальний обсяг роботи 97 сторінки, використано 18 літературних джерел.

Тема бакалаврської роботи: Розроблення апаратно-програмних засобів розподіленої IoT-системи з надлишковими вимірюваннями для виявлення відмов сенсорів та відновлення їх даних.

Мета роботи: розроблення апаратно-програмних засобів розподіленої IoT-системи, яка за рахунок організації надлишкових вимірювань забезпечує автоматичне виявлення відмов фізичних сенсорів у режимі реального часу та математичне відновлення їхніх значень для збереження цілісності часових рядів телеметрії.

Об'єкт дослідження: процеси збору, бездротової передачі, цифрової обробки телеметричної інформації та забезпечення її цілісності у просторово розподілених сенсорних IoT-мережах.

Предмет дослідження: апаратно-програмні, протокольні, архітектурні та алгоритмічні рішення для побудови розподіленої IoT-системи з надлишковими вимірюваннями, що реалізують функції автоматичної діагностики аномалій і аналітичного відновлення втрачених даних.

Результати бакалаврської роботи: Проведено аналіз природи та механізмів виникнення типових апаратних і програмних відмов у розподілених системах моніторингу мікроклімату. Обґрунтовано використання концепції аналітичної (структурної) надлишковості , а також застосування бездротового зв'язку Wi-Fi і протоколу MQTT для гнучкого просторового позиціонування вузлів. Реалізовано апаратні модулі мультипараметричного вимірювального вузла на базі мікроконтролера ESP32, прецизійного 16-розрядного АЦП ADS1115 для чотирьох температурних каналів та набору цифрових датчиків (BMP390, DHT22, BH1750FVI, MH-Z19C). Створено оптимізоване вбудоване програмне забезпечення (firmware) з використанням двоядерної архітектури FreeRTOS. Розроблено серверний застосунок на Python із використанням бази даних SQLite.

Створено інтерактивний веб-дашборд на Plotly Dash. Проведене тестування підтвердило стабільність наскрізного потоку даних, рівномірність часових міток із похибкою менше 10 мс та успішне функціонування алгоритмів автоматичного виявлення і відновлення пошкоджених відліків у режимі реального часу.

Ключові слова: РОЗПОДІЛЕНА IOT-СИСТЕМА, АНАЛІТИЧНА НАДЛИШКОВІСТЬ, ДІАГНОСТИКА ВІДМОВ, ВІДНОВЛЕННЯ ДАНИХ, ESP32, ADS1115, MQTT, СЕНСОРНА МЕРЕЖА, PYTHON, ВЕБ-ДАШБОРД.

ABSTRACT

The bachelor's thesis consists of 3 chapters, 13 figures, 9 tables, the total volume of the work is 97 pages, 18 literary sources are used.

Topic of the bachelor's thesis: Development of hardware and software tools for a distributed IoT system with redundant measurements for detecting sensor failures and recovering their data.

Purpose of the work: development of hardware and software tools for a distributed IoT system, which, due to the organization of redundant measurements, provides automatic detection of physical sensor failures in real time and mathematical recovery of their values to preserve the integrity of telemetry time series.

Object of research: processes of collection, wireless transmission, digital processing of telemetry information and ensuring its integrity in spatially distributed sensor IoT networks.

Subject of research: hardware, software, protocol, architectural and algorithmic solutions for building a distributed IoT system with redundant measurements that implement the functions of automatic anomaly diagnosis and analytical recovery of lost data.

Results of the bachelor's thesis: An analysis of the nature and mechanisms of typical hardware and software failures in distributed microclimate monitoring systems is carried out. The use of the concept of analytical (structural) redundancy, as well as the use of Wi-Fi wireless communication and MQTT protocol for flexible spatial positioning of nodes are substantiated. Hardware modules of a multiparameter measuring node based on the ESP32 microcontroller, a precision 16-bit ADC ADS1115 for four temperature channels and a set of digital sensors (BMP390, DHT22, BH1750FVI, MH-Z19C) are implemented. Optimized firmware using the dual-core FreeRTOS architecture is created. A server application in Python using the SQLite database is developed.

An interactive web dashboard on Plotly Dash is created. The testing confirmed the stability of the end-to-end data flow, the uniformity of timestamps with an error of less than 10 ms, and the successful operation of algorithms for automatic detection and recovery of damaged samples in real time. Keywords: DISTRIBUTED IOT SYSTEM, ANALYTICAL REDUNDANCY, FAULT DIAGNOSIS, DATA RECOVERY, ESP32, ADS1115, MQTT, SENSOR NETWORK, PYTHON, WEB DASHBOARD.

ЗМІСТ

ВСТУП 8
1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ДІАГНОСТИКИ ВІДМОВ ТА ЗАБЕЗПЕЧЕННЯ НАДІЙНОСТІ ДАНИХ У СЕНСОРНИХ МЕРЕЖАХ ... 11
1.1 Аналіз типових апаратних та програмних відмов у розподілених системах моніторингу мікроклімату 11
1.2 Концепція апаратної та аналітичної (структурної) надлишковості вимірювань в IoT-мережах 16
1.3 Огляд методів виявлення аномалій у багатовимірних часових рядах сенсорних даних 20
1.4 Математичні моделі та алгоритмічні підходи до відновлення спотворених сигналів 26
2 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ МУЛЬТИПАРАМЕТРИЧНОГО ВИМІРЮВАЛЬНОГО ВУЗЛА 32
2.1 Постановка задачі та концепція побудови вимірювальної системи 32
2.2 Обґрунтування вибору протоколу зв'язку 34
2.3 Апаратна платформа мікроконтролерного вузла 37
2.4 Вимірювальний тракт температури на базі АЦП ADS1115 41
2.4.1 Принцип дії та характеристики ADS1115 41
2.4.2 Схемотехнічна реалізація підключення ADS1115 45
2.5 Первинні перетворювачі температури 47
2.5.1 NTC-термістори 47
2.5.2 Кремнієвий p-n-діод як датчик температури 48
2.6 Датчики додаткових параметрів мікроклімату 51
2.6.1 Датчик атмосферного тиску BMP390 51
2.6.2 Датчик відносної вологості DHT22 51
2.6.3 Датчик концентрації CO_2 MH-Z19C 52

					КРБ.СІ-05.00.00.000 ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розробив		Лабай Д. Н.			Розроблення апаратно-програмних засобів розподіленої IoT-системи з надлишковими вимірюваннями для виявлення відмов сенсорів та відновлення їх даних	Літ.	Арк. 6 / Акрушів 93
Перевірив		Волинський О. І.					
Н.контр		Возний А. В.				ІФНТУНГ	
Затверд		Заміховська О. Л.					

2.6.4 Датчик освітленості BH1750FVI 53
2.7 Принципова електрична схема вимірювального вузла.....55
2.8 Просторова конфігурація мережі вимірювальних вузлів.....57
2.8.1 Принципи формування просторової сітки 57
2.8.2 Конфігурація вузлів у досліджуваному приміщенні 59
2.8.3 Фізичне кріплення та теплова ізоляція вузлів 60
2.9 Надлишковість вимірювань та стратегія реконструкції даних.....61
3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА АЛГОРИТМИ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНОЇ СИСТЕМИ.....64
3.1 Структурна організація програмного забезпечення.....64
3.2 Апаратна платформа вузла та підключені датчики.....65
3.3 Структура та алгоритм firmware ESP32 68
3.3.1 Загальна організація програми на Arduino (ESP32) 68
3.3.2 Цикл вимірювань та формування пакету 69
3.3.3 Прийом команд та управління сигналізацією 72
3.4 Розгортання MQTT-брокера Mosquitto на сервері 75
3.5 Серверний застосунок на Python: архітектура та модулі 76
3.6 Прийом, парсинг і синхронізація пакетів.....77
3.7 Структура бази даних SQLite та стратегія запису.....79
3.8 Модуль діагностики відмов 83
3.9 Модуль реконструкції даних 84
3.10 REST API та веб-дашборд.....85
3.11 Узагальнення архітектурних рішень.....86
ВИСНОВКИ 88
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 90
Додатки 93

ВСТУП

Актуальність теми. Тематика кваліфікаційної роботи передбачає розроблення апаратно-програмних засобів розподіленої IoT-системи з надлишковими вимірюваннями для виявлення відмов сенсорів та відновлення їх даних. Сучасні умови автоматизації технологічних процесів, розгортання систем «розумних» будівель та моніторингу критичної інфраструктури вимагають високої надійності й безперервності отримання телеметричної інформації. Проте у реальних умовах експлуатації первинні перетворювачі постійно піддаються деградації та збоям, серед яких найпоширенішими є адитивний дрейф нуля, імпульсні завади (стрибкові відмови), а також повна втрата або «замороження» сигналу через програмні помилки чи апаратні несправності.

Традиційні підходи до підвищення надійності за рахунок прямого апаратного дублювання вимірювальних каналів (наприклад, триплексних конфігурацій) мають суттєві обмеження через значне зростання вартості, габаритів та енергоспоживання системи, що критично для масштабних бездротових IoT-мереж. Ефективним альтернативним рішенням є використання аналітичної (структурної) надлишковості, яка базується на математичних, часових та просторових кореляціях між вимірюваними параметрами середовища. Оскільки фізичні процеси у замкненому просторі взаємопов'язані законами термодинаміки, формування надлишкової матриці спостережень дозволяє виявляти аномалії та реконструювати втрачені дані на аналітичному рівні без додаткових апаратних витрат. З огляду на це, проектування спеціалізованих апаратно-програмних засобів, які поєднують мікроконтролерні вузли, надійні протоколи передачі та серверні алгоритми

самодіагностики й компенсації відмов, є актуальною науково-практичною задачею.

Метою дослідження є розроблення апаратно-програмних засобів розподіленої IoT-системи, яка за рахунок організації надлишкових вимірювань забезпечує автоматичне виявлення відмов фізичних сенсорів у режимі реального часу та математичне відновлення їхніх значень для збереження цілісності часових рядів телеметрії.

Завданнями дослідження є:

- аналіз типових апаратних і програмних відмов у розподілених системах моніторингу та існуючих методів виявлення аномалій і відновлення спотворених сигналів у багатовимірних часових рядах;
- обґрунтування архітектурних рішень системи, вибору бездротового стандарту зв'язку Wi-Fi та прикладного протоколу MQTT для гнучкого просторового позиціонування вузлів;
- апаратна реалізація мультипараметричного вимірювального вузла на базі мікроконтролера ESP32, зовнішнього 16-розрядного сигма-дельта АЦП ADS1115 для чотирьох температурних каналів та цифрових датчиків додаткових параметрів (BMP390, DHT22, BH1750FVI, MH-Z19C);

Об’єктом дослідження є процеси збору, бездротової передачі, цифрової обробки телеметричної інформації та забезпечення її цілісності у просторово розподілених сенсорних IoT-мережах.

Предметом дослідження є апаратно-програмні, протокольні, архітектурні та алгоритмічні рішення для побудови розподіленої IoT-системи з надлишковими вимірюваннями, що реалізують функції автоматичної діагностики аномалій і аналітичного відновлення втрачених даних.

Практичне значення одержаних результатів полягає в розробці діючого апаратно-програмного прототипу розподіленої IoT-системи (у складі 12 вимірювальних вузлів та центрального сервера), який забезпечує неперервність наповнення баз даних навіть у разі повної відмови або деградації окремих фізичних сенсорів. Створені інженерні рішення щодо побудови прецизійного вимірювального тракту, двоядерної організації firmware під FreeRTOS, трирівневої системи фільтрації аномалій та батч-транзакцій у базі даних SQLite (режим WAL) мають універсальний характер і можуть бути безпосередньо використані при проектуванні промислових та комерційних IoT-систем із підвищеними вимогами до живучості та цілісності даних.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ДІАГНОСТИКИ ВІДМОВ ТА ЗАБЕЗПЕЧЕННЯ НАДІЙНОСТІ ДАНИХ У СЕНСОРНИХ МЕРЕЖАХ

1.1 Аналіз типових апаратних та програмних відмов у розподілених системах моніторингу мікроклімату

Розподілені системи моніторингу мікроклімату є невід'ємною складовою сучасної промислової автоматизації, агропромислового комплексу, систем управління розумними будівлями та критичної інфраструктури. Водночас надійність таких систем визначається не лише якістю окремих компонентів, а й архітектурними рішеннями щодо опрацювання відмов, що виникають на різних рівнях — від первинних перетворювачів до програмного забезпечення збору та аналізу даних. Розуміння природи, класифікації та механізмів розвитку відмов є необхідною передумовою для проектування ефективних методів їх виявлення та компенсації.

Відмови у сенсорних системах прийнято класифікувати за декількома критеріями: характером прояву (раптові та поступові), природою походження (апаратні та програмні), просторовою локалізацією (локальні та системні), а також ступенем впливу на вихідний сигнал (повна втрата сигналу, часткове спотворення, нестаціонарний зсув). Для систем моніторингу мікроклімату особливо актуальними є три групи відмов: дрейф нуля, шумові впливи та повна втрата сигналу.

Дрейф нуля (zero drift) є одним із найпоширеніших і водночас найбільш небезпечних видів деградації первинних перетворювачів. Він проявляється як повільна, монотонна зміна вихідного сигналу сенсора в умовах незмінного вимірюваного параметра і може бути спричинений

декількома факторами. По-перше, це старіння матеріалів чутливого елемента — в терморезисторах типу NTC зміна опору при старінні може досягати 0,5–2,0% за рік навіть в нормальних умовах експлуатації. По-друге, механічний та термічний гістерезис конструктивних елементів корпусу датчика: при циклічних температурних навантаженнях виникають залишкові деформації, що змінюють геометрію пружного елемента або умови теплообміну між чутливим елементом та довкіллям. По-третє, хімічна деградація під дією вологи, окисних газів або UV-випромінювання може змінити поверхневі властивості напівпровідникового p-n переходу діода, що використовується як температурний сенсор.

З точки зору метрологічної моделі дрейф нуля описується адитивною похибкою з повільно мінливою амплітудою. Якщо позначити істинне значення температури як T(t), а вихідний сигнал деградованого сенсора — як S(t), то модель відмови типу «дрейф» матиме вигляд [2]:

$$S(t) = T(t) + \delta(t), \qquad (1.1)$$

де δ(t) — функція дрейфу, яка в першому наближенні може бути апроксимована лінійною залежністю $\delta(t) = \alpha \cdot t$ або нелінійною типу

$$\delta(t) = \delta max \cdot (1 - \exp(-t/\tau d)), \qquad (1.2)$$

де α — швидкість дрейфу, δmax — асимптотичне значення зсуву, τd — характеристична стала часу процесу деградації. Небезпека дрейфу полягає в тому, що при відсутності еталонного сигналу або перехресної перевірки він залишається невиявленим протягом тривалого часу, поступово накопичуючи систематичну похибку в базі даних системи моніторингу.

Шумові впливи є другою за поширеністю групою відмов та деградацій сенсорного тракту. Природа шумів у вимірювальних системах є надзвичайно різноманітною. Теплові (джонсонівські) шуми, обумовлені хаотичним

тепловим рухом носіїв заряду в провідниках та напівпровідниках, мають рівномірний спектральний розподіл і є фундаментальним фізичним обмеженням точності будь-якого вимірювального пристрою. Для терморезистора з опором R при абсолютній температурі T та смузі пропускання B спектральна щільність теплового шуму визначається формулою Найквіста: Sn = 4kBTR, де kB — стала Больцмана. Окрім теплових шумів, у реальних системах присутні 1/f-шуми (флікер-шуми), що особливо значущі на низьких частотах і пов'язані з флуктуаціями рухомості носіїв заряду в напівпровідниках та дефектами кристалічної структури.

Електромагнітні завади є джерелом зовнішніх шумів, що набувають особливого значення в умовах насиченого промислового середовища або в приміщеннях із значною кількістю бездротових пристроїв. Для бездротових IoT-систем на основі Wi-Fi, що працюють у смузі 2,4 ГГц, це є особливо актуальним, оскільки ця смуга характеризується надзвичайно високою щільністю пристроїв і значним рівнем інтерференції. Вплив ЕМЗ на аналоговий вимірювальний тракт проявляється у вигляді адитивного шуму з характеристиками, що залежать від спектральних властивостей джерела завади та параметрів вимірювального кола. Статистично такі шуми часто описуються негаусівськими розподілами з важкими хвостами або моделлю суміші гаусіанів.

Серед різновидів шумових відмов окремо слід виділити т. зв. «стрибкові» відмови (spike faults) — короткочасні імпульсні спотворення сигналу, що виникають внаслідок мікровідключень з'єднань у ланцюзі живлення або вимірювальному ланцюзі, розряду електростатики, або неповного встановлення АЦП. Для 16-розрядного АЦП ADS1115 такі стрибки можуть сягати значень, що відповідають кільком десяткам градусів у

перерахунку на температуру, і тривають протягом одного-кількох тактів вимірювання. Виявлення таких відмов вимагає застосування порогової фільтрації на основі різницевих критеріїв або медіанної обробки ковзного вікна даних.

Повна втрата сигналу є найбільш очевидним, але водночас і найнебезпечнішим з точки зору цілісності системи видом відмови. Вона може виникати внаслідок механічного пошкодження сенсора або з'єднувальних провідників, відмови джерела живлення вузла, замикання або обриву в ланцюзі вимірювального мосту, заморожування прошивки мікроконтролера, а також відмови бездротового зв'язку (втрата Wi-Fi з'єднання, вичерпання буфера TCP/IP стека). Характерною особливістю цього виду відмов є те, що система може реагувати на них двома принципово різними способами: або видавати незмінне «замороженое» значення (stuck-at fault), або повністю припиняти передачу пакетів даних (data loss fault).

У першому випадку, коли сенсор видає постійне значення, що відповідає останньому виміру або деякому опорному рівню, алгоритм виявлення відмов повинен аналізувати дисперсію сигналу в ковзному часовому вікні. В нормальних умовах температурний сигнал завжди містить певний рівень природних флуктуацій, обумовлених мікроконвекцією повітря та турбулентними процесами теплообміну, тому дисперсія σ^2 у часовому вікні завжди перевищує певний мінімальний поріг. Якщо σ^2 стає близькою до нуля або нижчою за поріг теплового шуму ADS1115 (що відповідає приблизно 0,03–0,05°C для типових режимів роботи), це є надійним індикатором відмови типу «замороження».

У другому випадку — повної відсутності пакетів — сервер реєструє відмову зв'язку, яка може бути як тимчасовою (флуктуації Wi-Fi-каналу), так

і постійною (апаратна відмова вузла). Розрізнення цих двох ситуацій є важливою задачею, оскільки алгоритм відновлення даних повинен активуватися лише при стійкій відмові, уникаючи зайвих обчислень при короткочасних розривах зв'язку. Для цього доцільно використовувати адаптивний таймаут, що враховує статистику попередніх затримок передачі від конкретного вузла.

Програмні відмови в розподілених IoT-системах формують окрему, часто недооцінювану категорію загроз цілісності даних. На рівні вбудованого програмного забезпечення (firmware) ESP-01 найбільш характерними є: переповнення стека при рекурсивних викликах або виділенні великих масивів у локальній пам'яті, нескінченне очікування відповіді від I2C-пристрою при зависанні шини (I2C bus freeze), некоректне відновлення після перезавантаження за відсутності збереження стану, а також помилки скидання сторожового таймера (watchdog timer) при довготривалих операціях DNS-запиту або встановлення TLS-з'єднання.

На рівні бібліотек обробки даних існує специфічний клас програмних відмов, пов'язаних із числовими помилками: переповнення цілочисельних типів при накопиченні сум для обчислення середнього, втрата точності при конвертації між типами float та int, а також похибки округлення при масштабуванні АЦП-кодів до фізичних одиниць. Для ADS1115 з розрядністю 16 біт та внутрішнім підсилювачем програмного підсилення (PGA) некоректне налаштування діапазону вимірювання може призвести до систематичного «насичення» (saturation) сигналу, що виглядає в потоці даних як постійне значення на рівні граничного коду АЦП[3].

Таким чином, типологія відмов у розподілених системах моніторингу мікроклімату є комплексною та багаторівневою. Ефективна система

діагностики повинна враховувати усі зазначені механізми і забезпечувати їх диференційоване виявлення на основі характеристик вихідного сигналу. Наявність просторово розподіленого масиву сенсорів відкриває принципово нові можливості для крос-валідації вимірювань і є ключовою передумовою для реалізації аналітичної надлишковості.

1.2 Концепція апаратної та аналітичної (структурної) надлишковості вимірювань в IoT-мережах

Надлишковість (redundancy) є фундаментальним принципом проектування надійних вимірювальних та обчислювальних систем (рис.1.1). В загальному розумінні надлишковість означає наявність у системі додаткових ресурсів — апаратних, програмних або інформаційних — понад мінімально необхідних для виконання цільової функції. Ці додаткові ресурси не залучаються в штатному режимі роботи, але активуються при виникненні відмов, забезпечуючи неперервність функціонування або відновлення втраченої інформації. У контексті IoT-систем моніторингу мікроклімату розрізняють два принципово різні підходи: апаратну та аналітичну (структурну) надлишковість.

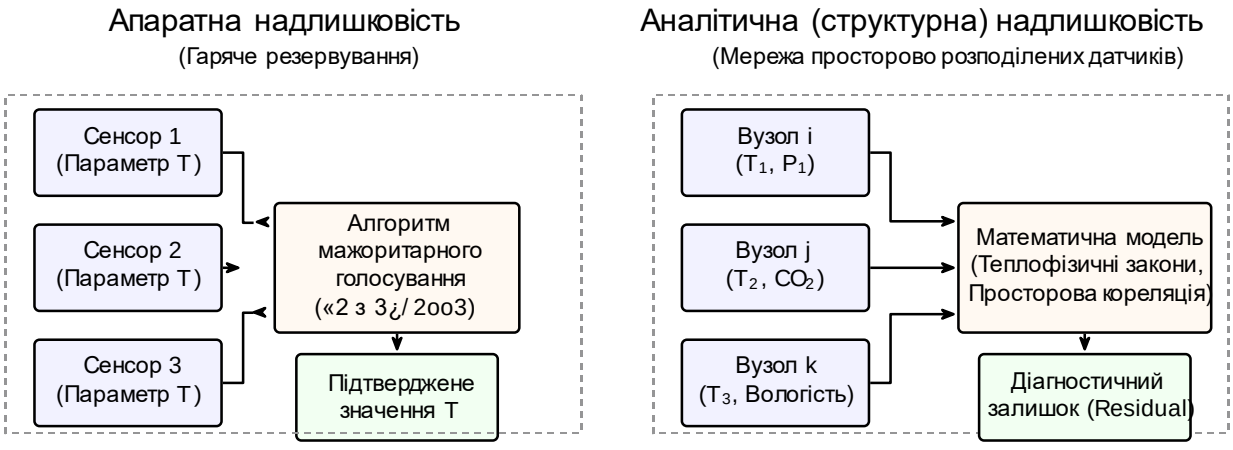


Рисунок 1.1– Види інформаційної надлишковості

Апаратна надлишковість (hardware redundancy) передбачає фізичне дублювання або мультиплікацію вимірювальних каналів з метою забезпечення безперервності вимірювання при відмові одного або кількох елементів. У класичній формі вона реалізується у вигляді гарячого резервування (hot standby), коли кілька ідентичних сенсорів одночасно вимірюють той самий параметр, і їх результати порівнюються за допомогою алгоритму голосування (voting). Найпоширенішою схемою є триплексна конфігурація «2 з 3» (2oo3 voting), при якій система функціонує коректно при відмові одного з трьох каналів: результатом вимірювання вважається значення, що підтримується принаймні двома каналами[4].

Виходячи з теорії надійності, ймовірність відмови триплексної системи при незалежних відмовах каналів визначається як

$$P\_fail = 3P^2(1-P) + P^3, \qquad (1.3)$$

де P — ймовірність відмови одного каналу за заданий проміжок часу. Якщо P = 0,01 (1% ймовірність відмови за рік), то P_fail ≈ $3 \cdot 10^{-4}$, що на два порядки менше, ніж для одного каналу. Однак апаратне дублювання пов'язане зі значним зростанням вартості та фізичних розмірів системи, особливо при розгортанні великих сенсорних мереж. Для системи з N вузлами перехід від одиничних до триплікснних конфігурацій потребує збільшення кількості сенсорів у 3 рази і відповідного масштабування кабельної інфраструктури та джерел живлення.

Холодне резервування (cold standby) є компромісним варіантом апаратної надлишковості, при якому резервний канал не знаходиться в активному стані і активується лише при виявленні відмови основного. Це знижує енергоспоживання системи, але вводить затримку перемикання та вимагає надійного механізму детектування відмови і автоматичного

комутації. Для бездротових IoT-систем таке рішення є технічно складнішим, оскільки потребує централізованої оркестрації з боку сервера або контролера мережі.

Принципово іншим підходом є аналітична, або структурна надлишковість (analytical/structural redundancy), яка ґрунтується не на фізичному дублюванні вимірювальних каналів, а на математичних зв'язках між вимірюваними величинами. Ключова ідея полягає в тому, що будь-яка фізична система перебуває під дією законів природи, які пов'язують різні параметри детермінованими або стохастичними залежностями. Якщо ці залежності відомі (або можуть бути ідентифіковані з наявних даних), то значення будь-якого з параметрів можна відновити або перевірити, використовуючи вимірювання інших параметрів.

У контексті просторово розподіленого масиву температурних сенсорів аналітична надлишковість проявляється через фізичну скорельованість температурних полів у сусідніх точках простору. Теплові процеси в замкненому приміщенні підпорядковуються рівнянню теплопровідності та законам конвективного теплообміну, що зумовлює значну просторову кореляцію між показниками сусідніх вузлів. Математично це означає, що розмірність реального простору станів теплового поля є значно нижчою за кількість вимірювань, і ця надмірність вимірювань може бути використана для виявлення аномалій та відновлення відсутніх даних.

Концепція аналітичної надлишковості була систематизована в роботах П'ю та Айзермана ще в 1960-х роках у контексті задач ідентифікації та діагностики лінійних динамічних систем. Пізніше, з розвитком теорії спостерігачів стану (state observers), методів фільтрації Калмана та машинного навчання, цей підхід отримав значний розвиток і сьогодні є

домінуючим у системах діагностики складних технічних об'єктів. Ключовим поняттям в аналітичній надлишковості є «залишок» (residual) — різниця між виміряним значенням та його оцінкою, отриманою з математичної моделі системи або з інших вимірювань. При нормальній роботі залишок близький до нуля (в межах шуму), а при відмові він перевищує встановлений поріг, сигналізуючи про аномалію.

Для IoT-систем з просторово розподіленими сенсорами аналітична надлишковість реалізується через ряд механізмів. Перший — просторова інтерполяція: значення температури в довільній точці простору може бути оцінене за допомогою зважених показників сусідніх вузлів, і відхилення реального вимірювання від цієї оцінки служить діагностичним сигналом. Другий механізм — часова екстраполяція: на основі попередньої динаміки сигналу вузла будується прогноз його поточного значення, і суттєве відхилення реального вимірювання від прогнозу вказує на аномалію. Третій — міжпараметричні залежності: при наявності різнорідних сенсорів (температура, вологість, CO2) відомі фізичні зв'язки між параметрами дозволяють оцінювати їх взаємну узгодженість.

Важливою перевагою аналітичної надлишковості в порівнянні з апаратною є її масштабованість та економічність. Замість потрійного фізичного резервування кожного вузла достатньо забезпечити достатню просторову щільність розміщення сенсорів, при якій кожен вузол має кількох «сусідів», здатних виступити його аналітичними резервами. При цьому кожен додатковий вузол підвищує надійність не лише свого безпосереднього оточення, а й усієї мережі в цілому, оскільки збільшує кількість доступних міжвузлових кореляцій. Для температурних полів у приміщеннях, де просторова кореляційна довжина становить, як правило, від 0,5 до 3 метрів

залежно від інтенсивності конвекції, достатньою є щільність вузлів порядку 1–2 вузли на 1–2 м² площі перерізу.

Слід зазначити, що апаратна та аналітична надлишковість не є взаємовиключними і можуть ефективно поєднуватися в гібридних архітектурах. Наприклад, локальне дублювання чутливих елементів у межах одного вузла (дешеве і компактне рішення) в поєднанні з аналітичною обробкою на рівні мережі забезпечує захист як від апаратних відмов конкретного датчика, так і від системних збоїв у вузлі в цілому. У контексті даної роботи основний акцент зроблено на аналітичній надлишковості як більш гнучкому та масштабованому підході, що дозволяє витягувати максимум корисної інформації із наявної сенсорної мережі без додаткових апаратних витрат.

1.3 Огляд методів виявлення аномалій у багатовимірних часових рядах сенсорних даних

Виявлення аномалій (anomaly detection) у часових рядах сенсорних даних є активно розвиваючою областю досліджень, що знаходиться на перетині теорії обробки сигналів, математичної статистики та машинного навчання. Аномалія у загальному розумінні — це спостереження, що суттєво відрізняється від очікуваного патерну поведінки системи (рис.1.2). Стосовно сенсорних систем аномалії поділяються на три основні категорії: точкові (pointwise) — окремі значення, що різко відхиляються від поточного рівня сигналу; контекстуальні (contextual) — значення, що нормальні в одному контексті, але аномальні в іншому; колективні (collective) — послідовності значень, кожне з яких окремо виглядає нормально, але їх сукупність є аномальною (наприклад, дрейф нуля або замороження)[5].

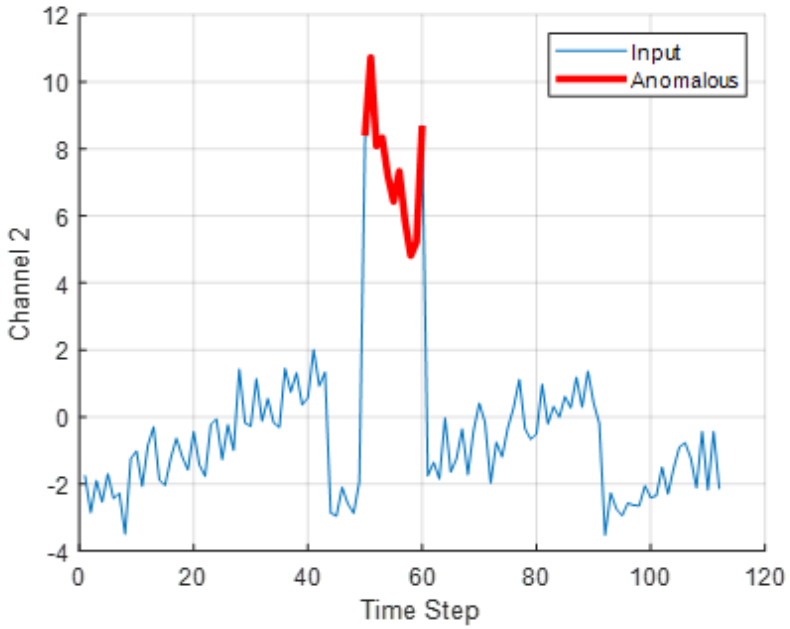


Рисунок 1.2 – Приклад аномальних даних в потоці

Серед статистичних методів виявлення аномалій найширше застосування мають порогові методи, засновані на аналізі розподілу статистик сигналу. Найпростіший підхід — пряме порогування: сигнал вважається аномальним, якщо його поточне значення виходить за межі довірчого інтервалу $[\mu - k\sigma, \mu + k\sigma]$, де μ — поточна ковзна середня, σ — поточне ковзне стандартне відхилення, k — коефіцієнт, що визначає рівень чутливості (зазвичай $k = 3$, що відповідає критерію «3 сигма» для гаусівського розподілу). Незважаючи на простоту, цей метод добре працює для виявлення точкових аномалій у стаціонарних або повільно змінюваних процесах.

Для виявлення аномалій на часових рядах зі змінною дисперсією та нелінійними трендами більш ефективним є метод CUSUM (Cumulative Sum Control Chart). В цьому методі аналізується накопичена сума відхилень сигналу від еталонного значення:

$$C(t) = \max(0, C(t-1) + x(t) - \mu - k), \qquad (1.4)$$

де x(t) — поточне значення, μ — очікуване середнє, k — допустиме відхилення. Виявлення аномалії відбувається при перевищенні порогу h, що може бути налаштований залежно від бажаного балансу між чутливістю та частотою хибних спрацювань. CUSUM є оптимальним у сенсі послідовного критерію Wald для виявлення стрибкоподібних змін рівня сигналу і широко застосовується у задачах виявлення деградації сенсорів.

Метод EWMA (Exponentially Weighted Moving Average) поєднує властивості низькочастотного фільтра та контрольної карти. Ковзна середня з експоненційним зважуванням обчислюється за рекурентним виразом:

$$m(t) = \lambda \cdot x(t) + (1-\lambda) \cdot m(t-1), \qquad (1.5)$$

де $\lambda \in (0,1]$ — коефіцієнт згладжування. Аномалія виявляється при виході m(t) за межі контрольного коридору. Перевагою EWMA є його здатність відстежувати повільні тренди, що робить його ефективним для виявлення дрейфу нуля. Параметр λ визначає компроміс між чутливістю до раптових змін (великий λ) та здатністю фільтрувати шуми (малий λ) (рис.1.3).

Для багатовимірних систем, де аномалія може виявлятися лише у взаємозв'язку між кількома змінними, а не в абсолютних значеннях кожної з них, ефективним інструментом є метод головних компонент (Principal Component Analysis, PCA). У цьому підході вихідний простір вимірювань хи∈R^n проектується у простір головних компонент меншої розмірності. Нормальна поведінка системи відповідає певній підпростору (hyperplane) у просторі головних компонент, і відхилення від цього підпростору є індикатором аномалії. Статистикою для виявлення аномалій у PCA-підході є Хотеллінга T^2 та Q-статистика (сума квадратів залишків проекції). Важливою перевагою PCA є його нечутливість до повільних синхронних змін усіх сенсорів (наприклад, зміни навколишньої температури), оскільки вони

відображаються в одному-двох перших головних компонентах і не впливають на Q-статистику.

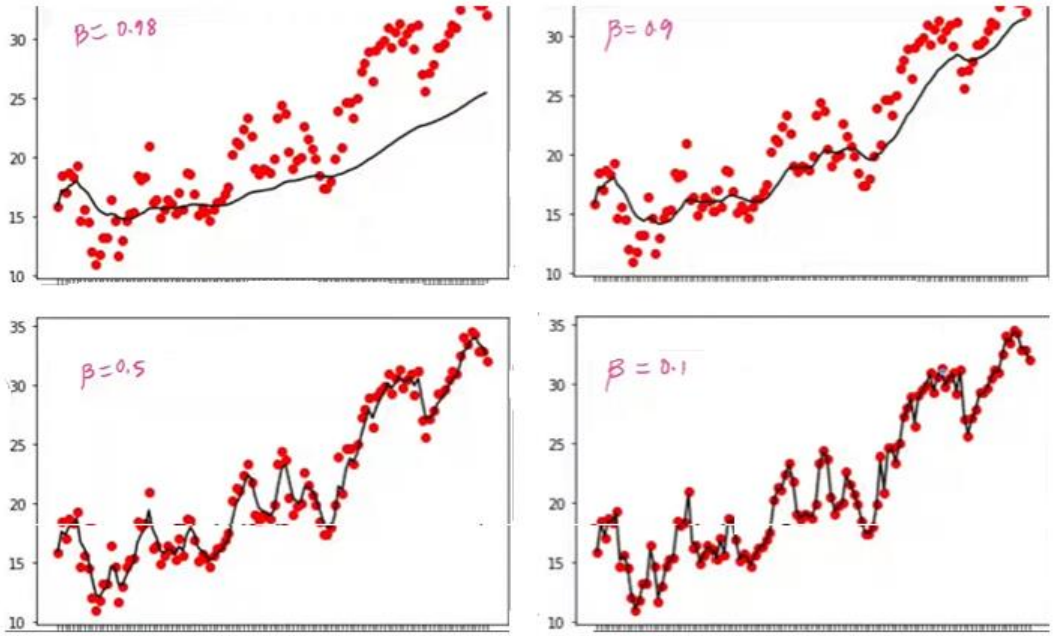


Рисунок 1.3– Вплив коефіцієнта згладжування на форму функції

Методи на основі теорії часових рядів відкривають ще один рівень аналізу. Авторегресійні моделі (AR, ARMA, ARIMA) дозволяють будувати прогностичні моделі динаміки кожного сенсора на основі його попередніх значень. Залишки прогнозу — різниця між реальним та передбаченим значенням — є потужним діагностичним сигналом: в нормальному режимі вони являють собою білий шум, а при виникненні відмови набувають систематичного характеру. Для нестаціонарних сигналів (наприклад, добові тренди температури) перевага надається сезонним моделям SARIMA або методам декомпозиції на тренд та залишок[6].

Нейромережеві методи виявлення аномалій набули значного поширення в останні роки завдяки розвитку архітектур глибокого навчання

(рис.1.4). Автокодувальники (autoencoders) — нейронні мережі, навчені відтворювати вхідний сигнал через низькорозмірне «вузьке місце» — ефективно виявляють аномалії як зразки з великою помилкою реконструкції. Рекурентні нейронні мережі (LSTM, GRU) здатні моделювати складні часові залежності у багатовимірних часових рядах і виявляти аномалії, що не можуть бути ідентифіковані простими статистичними методами. Трансформерні архітектури, зокрема моделі типу Anomaly Transformer та PatchTST, демонструють видатні результати на еталонних наборах даних сенсорних систем.

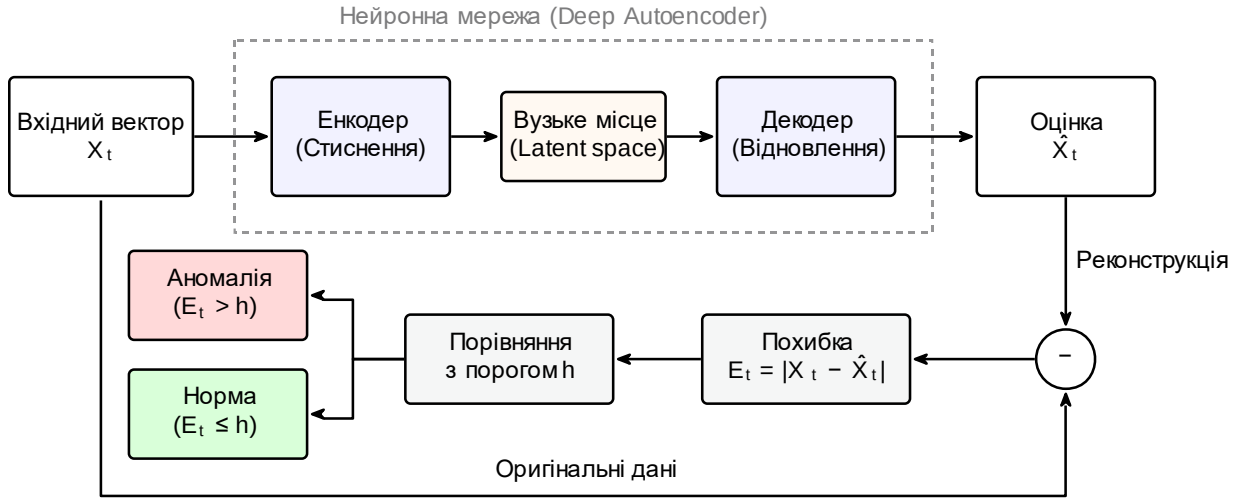


Рисунок 1.4– Нейромережеві методи виявлення аномалій

Однак застосування глибокого навчання в реальних IoT-системах обмежується рядом практичних факторів: необхідністю значних обсягів маркованих або немаркованих навчальних даних, обчислювальної потужністю для навчання моделей, а також низькою інтерпретованістю результатів. Для систем моніторингу мікроклімату в режимі реального часу більш прагматичним є поєднання простих, але надійних статистичних методів (порогування, CUSUM, EWMA) з просторово-кореляційним аналізом, що використовує специфіку розподіленої архітектури сенсорної

мережі. Гібридний підхід дозволяє досягти задовільної точності діагностики при мінімальних обчислювальних витратах, що особливо важливо в умовах обмежених ресурсів IoT-платформ.

Окремої уваги заслуговують методи виявлення аномалій в просторово розподілених вимірюваннях, що враховують не лише часову, а й просторову структуру даних. Метод геостатистичної аномалії заснований на порівнянні виміряного значення кожного вузла з його просторовою оцінкою, отриманою методом кригінгу (Kriging) з показників сусідніх вузлів. Різниця між реальним вимірюванням та кригінг-оцінкою — т. зв. «кросс-валідаційний залишок» — в нормальному режимі є стаціонарним процесом з нульовим середнім, і значне відхилення від нуля вказує на проблему в конкретному вузлі. Перевагою цього методу є його просторова адаптивність: чутливість до локальних аномалій висока, тоді як глобальні тренди, що відображаються у всіх вузлах одночасно, автоматично компенсуються.

Метод просторово-часового виявлення аномалій на основі локально зваженої регресії (LOESS/LOWESS) поєднує переваги часового та просторового аналізу. Для кожного вузла в кожен момент часу будується локальна регресійна модель, що використовує як попередні значення даного вузла, так і поточні значення просторових сусідів. Апроксимація виконується поліномами низького ступеня в зваженій метриці, де ваги зменшуються зі збільшенням часової або просторової відстані. Залишок локальної регресії є чутливим до будь-яких форм аномалій: точкових стрибків, повільного дрейфу, та замороження сигналу.

1.4 Математичні моделі та алгоритмічні підходи до відновлення спотворених сигналів

Відновлення спотворених або відсутніх даних сенсорів (sensor data imputation) є задачею, що випливає безпосередньо із задачі їх виявлення: після локалізації відмови виникає необхідність отримати достовірний замінник відсутнього або пошкодженого вимірювання для забезпечення безперервності баз даних та алгоритмів прийняття рішень. Математичні підходи до вирішення цієї задачі охоплюють широкий спектр методів — від простих детерміністичних інтерполяцій до складних ймовірнісних моделей та методів машинного навчання [1].

Кореляційний аналіз (рис.1.5) є фундаментальним інструментом для кількісної оцінки взаємозв'язків між показниками просторово розподілених сенсорів і відіграє ключову роль у побудові моделей відновлення даних. У стохастичному підході температурне поле T(r,t) розглядається як просторово-часовий випадковий процес, що характеризується функцією просторово-часової кореляції

$$K(r1,r2,\tau) = E[(T(r1,t)-\mu(r1,t))\cdot(T(r2,t+\tau)-\mu(r2,t+\tau))], \quad (1.6)$$

де r1, r2 — просторові координати двох вузлів, τ — часовий лаг. Для стаціонарних та просторово однорідних процесів ця функція залежить лише від різниці координат: $K(\Delta r,\tau)$, що суттєво спрощує її оцінювання.

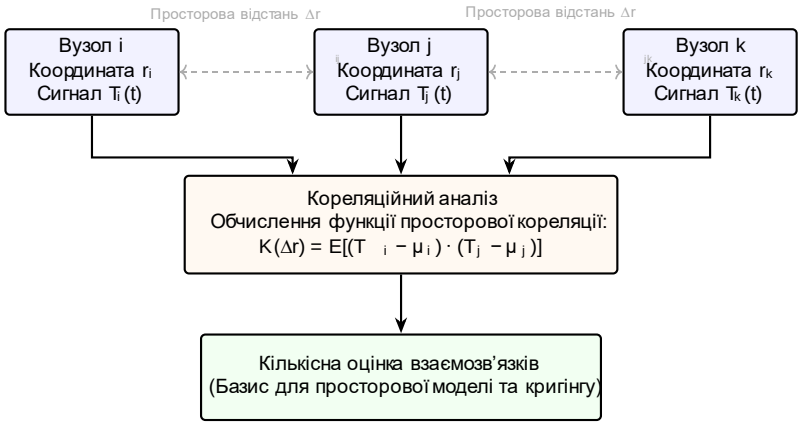


Рисунок 1.5– Роль кореляційного аналізу в просторово розподіленій мережі

Емпіричне оцінювання функції просторової кореляції виконується за накопиченою базою даних:

$$\hat{K}(\Delta r) = (1/N)\cdot\Sigma[(T\_i(t)-\mu\_i)\cdot(T\_j(t)-\mu\_j)] \quad (1.7)$$

для всіх пар вузлів (i,j) з відстанню $||r\_i - r\_j|| \approx \Delta r$. Апроксимація емпіричної кореляційної функції аналітичними моделями (сферична, експоненційна, гаусівська) дозволяє отримати неперервну просторову кореляційну структуру, яка використовується для побудови кригінг-інтерполятора. Гаусівська модель $K(\Delta r) = \sigma^2\cdot\exp(-(\Delta r/l)^2)$, де l — характеристична кореляційна довжина, добре відображає властивості теплових полів у стаціонарних умовах і є найбільш поширеною у геостатистичному аналізі температурних даних[7].

Просторова інтерполяція є найбільш прямим підходом до відновлення значення відмовившого сенсора на основі показників його просторових сусідів. Метод зворотно-зважених відстаней (Inverse Distance Weighting, IDW) є найпростішою реалізацією просторової інтерполяції:

$$\hat{Y}(r0) = \Sigma[w\_i\cdot T\_i] / \Sigma w\_i, \quad (1.8)$$

де w_i = 1/||r0–r_i||^p, p — показник степеня (зазвичай p=2). Незважаючи на простоту, IDW має суттєвий недолік: він не враховує просторову кореляційну структуру поля і тому є субоптимальним у сенсі мінімуму дисперсії похибки інтерполяції.

Метод кригінгу (Kriging), розроблений геологом Д. Г. Крігом та математично обґрунтований Ж. Матероном в 1960-х роках, є оптимальним лінійним незміщеним оцінювачем (BLUE — Best Linear Unbiased Estimator) для стохастичних просторових полів. Ординарний кригінг знаходить оцінку $\hat{Y}(r0) = \Sigma[\lambda_i \cdot T_i]$ як лінійну комбінацію відомих значень з вагами λ_i, що мінімізують дисперсію похибки при умові незміщеності оцінки. Ваги визначаються з системи рівнянь кригінгу:

$$\Sigma_j[\gamma(r_i - r_j)\cdot\lambda_j] + \mu = \gamma(r_i - r0), \qquad (1.9)$$

де $\gamma(h)$ — варіограма (половина дисперсії різниці між значеннями у двох точках, розділених вектором h), а μ — множник Лагранжа, що забезпечує умову незміщеності.

Практична реалізація кригінгу для задач відновлення сенсорних даних у режимі реального часу вимагає ефективних алгоритмів розв'язання системи кригінгу. При кількості вузлів N наповненість системи становить (N+1)×(N+1) при ординарному кригінгу, і її розв'язання методом LU-розкладання вимагає $O(N^3)$ операцій. Для типових IoT-систем з N=10–50 вузлами це є цілком прийнятним обчислювальним навантаженням для сервера на базі стандартного ПК. При відмові одного з вузлів система кригінгу перебудовується з виключенням рядка та стовпця, що відповідають відмовившому вузлу, і нова оцінка може бути отримана за $O(N^2)$ операцій завдяки формулам оновлення матриць.

Метод розкладання за власними векторами (Empirical Orthogonal Functions, EOF або PCA у просторовому контексті) є потужним інструментом для компактного представлення просторово-часових полів і може бути використаний для відновлення відсутніх даних. У цьому підході матриця даних

$$X \in R^{T \times N} \tag{1.10}$$

де T — кількість моментів часу, N — кількість вузлів розкладається у сингулярні компоненти:

$$X = U \cdot \Sigma \cdot V^T. \tag{1.11}$$

Перші k головних просторових мод $V_1,...,V_k$ та відповідні їм часові коефіцієнти $U_1,...,U_k$ описують переважну частину дисперсії поля при малому k. Відновлення відсутнього вимірювання виконується шляхом проекції доступних даних на простір k головних компонент та реконструкції значення відсутнього елемента.

Для задач відновлення даних у режимі реального часу особливо перспективним є метод матричного завершення (matrix completion), що використовує властивість низькорангового наближення просторово-часових полів. При відсутності окремих вимірювань задача ставиться як мінімізація ядерної норми (nuclear norm) неповної матриці даних, що є опуклим наближенням до задачі пошуку мінімальнорангового доповнення. Алгоритми на зразок SOFT-IMPUTE або SVT (Singular Value Thresholding) дозволяють ефективно розв'язувати цю задачу при частці відсутніх значень до 50–70%.

Фільтр Калмана та його нелінійні узагальнення (розширений фільтр Калмана, сигма-точковий фільтр Калмана, фільтр частинок) є стандартним інструментом оптимальної оцінки стану динамічних систем при наявності шумів вимірювань та невизначеностей у моделі. У контексті мережі

температурних сенсорів стан системи x_t включає температури у всіх вузлах, а рівняння стану описує їх динаміку відповідно до теплофізичної моделі приміщення. При відмові одного з сенсорів відповідний вектор вимірювання стає неповним, і фільтр Калмана автоматично використовує прогноз з рівняння стану для цього вузла, зважено поєднуючи його з доступними вимірюваннями сусідів.

Байєсівський підхід до відновлення даних є найбільш загальним і дозволяє систематично враховувати всю доступну апріорну інформацію про фізичні властивості системи. У рамках байєсівської формули

$$p(T_{missing} \mid T_{observed}) \propto p(T_{observed} \mid T_{missing}) \cdot p(T_{missing}), \quad (1.12)$$

де p(T_missing) — апріорний розподіл відсутніх значень (що визначається фізичними законами та просторовою кореляційною структурою), а p(T_observed | T_missing) — функція правдоподібності, що описує залежність між доступними та відсутніми вимірюваннями. Для гаусівських апріорних розподілів та лінійних залежностей апостеріорний розподіл також є гаусівським, і його параметри можуть бути аналітично обчислені, що дає оцінку, еквівалентну кригінгу.

Питання непрямих вимірювань параметрів середовища на основі аналізу просторово-часових змін температурного поля є актуальним науковим напрямком, що лежить на перетині термодинаміки, метрології та теорії зворотних задач. Зв'язок між температурними градієнтами та динамікою вологості ґрунтується на термодинамічному рівнянні стану вологого повітря та явищі психрометричного охолодження. У процесі випаровування вологи з поверхні або конденсації водяної пари при охолодженні виникають характерні теплові ефекти: ефективна теплоємність вологого повітря відрізняється від сухого, а процеси фазового переходу

(конденсація, випаровування) супроводжуються виділенням або поглинанням прихованої теплоти, що відображається у температурних профілях. Аналіз цих особливостей дозволяє, принаймні якісно, оцінювати відносну вологість на основі виключно температурних вимірювань.

Оцінка конвективних потоків газів за просторово-часовими температурними градієнтами також ґрунтується на фундаментальних рівняннях гідродинаміки та теплообміну. Рівняння Нав'є-Стокса для в'язкої нестисливої рідини у поєднанні з рівнянням теплопровідності описують спільну еволюцію поля швидкостей та температурного поля. Зворотна задача — відновлення поля швидкостей за відомим температурним полем — є некоректно поставленою в загальному випадку, однак при певних апріорних обмеженнях (наприклад, ламінарний режим течії, конвекція Релея-Бенара у горизонтальному шарі) вона допускає стійкі розв'язки. Зокрема, вертикальний температурний градієнт є прямим показником інтенсивності природної конвекції, а горизонтальний — характеризує напрямок конвективних потоків. Для кількісної оцінки швидкості потоків використовуються методи пасивної трасерної термоанемометрії, що базуються на аналізі часів запізнення між просторово рознесеними температурними сигналами.

Таким чином, поєднання кореляційного аналізу, методів просторової інтерполяції (зокрема кригінгу), апаратних засобів виявлення аномалій та алгоритмів реконструкції сигналів утворює комплексну методологічну основу для розв'язання центральної задачі даної кваліфікаційної роботи — побудови надійної розподіленої IoT-системи з функцією самодіагностики та автоматичного відновлення втрачених вимірювань мікроклімату.

2 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ МУЛЬТИПАРАМЕТРИЧНОГО ВИМІРЮВАЛЬНОГО ВУЗЛА

2.1 Постановка задачі та концепція побудови вимірювальної системи

Задача, що вирішується у даній роботі, полягає у виявленні прихованих функціональних залежностей між параметрами мікроклімату приміщення та у використанні цих залежностей для відновлення показань фізичних датчиків у разі їхньої відмови або тимчасової недоступності. Фізичні процеси, що протікають у замкненому приміщенні, — теплообмін між стінами, повітрям і предметами; конвекція; газообмін між людьми та атмосферою; поглинання і перетворення радіаційної енергії освітлення — породжують стійкі кореляції між температурою, вологістю, концентрацією CO_2, атмосферним тиском та освітленістю. Статистична або апроксимаційна модель цих залежностей дозволяє оцінювати будь-який параметр за рештою: наприклад, відновлювати показання датчика вологості при його відмові на основі кількох температурних каналів та рівня CO_2.

Для побудови такої моделі необхідна надлишкова матриця спостережень: кожен вузол вимірює кілька фізично різних величин у кожний момент часу, а сукупність вузлів формує просторово рознесену вибірку. Кількість вимірюваних параметрів і просторове охоплення повинні перевищувати мінімально необхідні для однієї задачі вимірювання — саме ця надлишковість забезпечує достатній обсяг даних для навчання моделі та подальшої реконструкції.

Виходячи з наведеного, склад вимірюваних параметрів обрано таким: температура повітря у чотирьох просторово рознесених точках кожного вузла (за допомогою зовнішнього прецизійного АЦП ADS1115 та аналогових

первинних перетворювачів), атмосферний тиск і температура (датчик BMP390), відносна вологість (датчик DHT22), освітленість (датчик BH1750FVI) та об'ємна концентрація вуглекислого газу (датчик MH-Z19C). Ці п'ять класів параметрів охоплюють основні фізичні процеси мікроклімату та формують достатньо повну векторну характеристику стану приміщення в кожний момент часу.

Окремої уваги потребує обґрунтування архітектурного рішення щодо вимірювання температури. Виявлення функціональних залежностей між температурними каналами або між температурою та вологістю можливе лише за умови достатньо точного відтворення температури: систематична похибка, порівнянна з амплітудою виявлюваних ефектів, унеможливлює побудову достовірної моделі. Власний 10-розрядний АЦП мікроконтролерних модулів ESP8266/ESP32 має нелінійність до 10 МЗР, значний температурний дрейф нуля та відсутній апаратний підсилювач із програмним коефіцієнтом посилення. У результаті його ефективна роздільна здатність при вимірюванні температури через аналоговий первинний перетворювач не перевищує 0,5–1 °C, що є недостатнім для виявлення тонких залежностей. Тому вимірювальний тракт температури побудовано на базі зовнішнього диференційного сигма-дельта АЦП ADS1115 з 16-розрядною роздільною здатністю та програмованим підсилювачем PGA, що забезпечує практичну роздільну здатність 0,008–0,01 °C.

Бездротова передача даних реалізована за технологією Wi-Fi IEEE 802.11 b/g/n. Вибір продиктований вимогою вільного тривимірного позиціонування вузлів у просторі приміщення без прокладання кабелів. У наступному підрозділі наведено порівняльний аналіз Wi-Fi та дротової

альтернативи — RS-485/Modbus, що традиційно застосовується у промислових системах моніторингу.

2.2 Обґрунтування вибору протоколу зв'язку

Дротовий протокол RS-485 (рис. 2.1) реалізує напівдуплексний послідовний зв'язок через симетричну (диференційну) пару провідників. Стандарт EIA/TIA-485 встановлює мінімальний диференційний рівень сигналу ±200 мВ, що забезпечує надійне розпізнавання логічних рівнів при синфазному шумі до ±12 В. Максимальна довжина лінії становить 1200 м при 9600 бод і близько 120 м при 100 кбод. Шинна топологія дозволяє підключити до 32 стандартних вузлів без репітерів або до 256 із застосуванням пристроїв із одиничним навантаженням. Протокол Modbus RTU, що реалізується поверх RS-485, визначає адресацію (1–247 пристроїв), набір функціональних кодів для зчитування та запису регістрів і перевірку цілісності даних на основі CRC-16. Завдяки цим властивостям RS-485/Modbus упродовж десятиліть домінує в автоматизованих системах управління та промислових системах моніторингу [8].

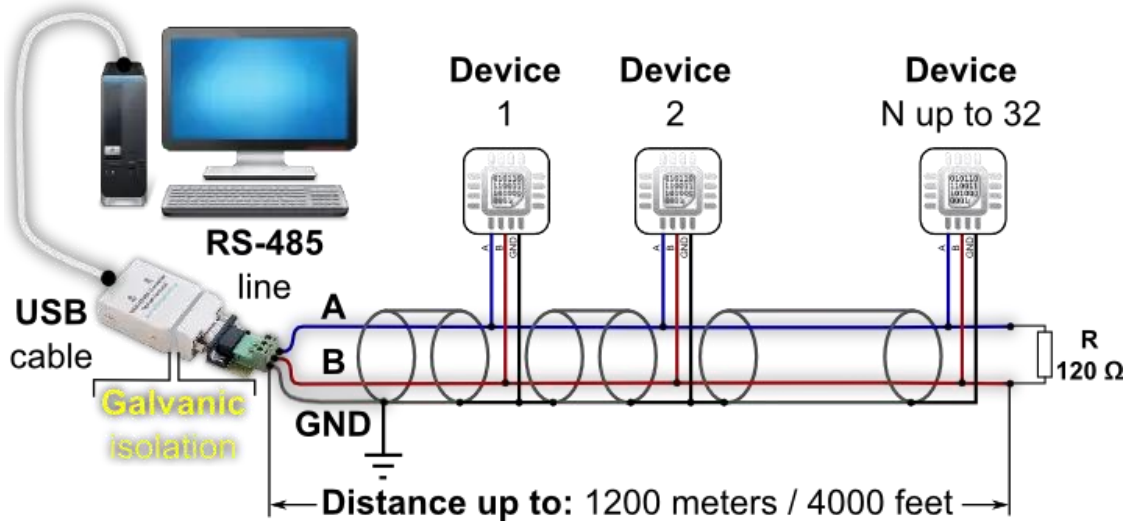


Рисунок 2.1–Топологія шини RS485

Проте застосування RS-485 для задачі просторово рознесеного мультипараметричного моніторингу приміщення пов'язане з рядом принципових обмежень. По-перше, прокладання кабельної траси до вузлів, рознесених у тривимірному просторі, є технічно складним завданням. Навіть у відносно невеликому приміщенні 5×4 м із висотою 3 м розміщення 12 вузлів на різних висотах та відстанях від стін вимагає прокладання десятків метрів кабелю з кріпильними елементами, гофрованими трубами та монтажними коробами. По-друге, зміна просторового положення вузла для дослідницьких цілей вимагає фізичного втручання в кабельну інфраструктуру. По-третє, шинна топологія RS-485 за своєю природою є однорівневою і не передбачає ефективного підключення вузлів, підвішених у повітрі далеко від будь-яких стін. По-четверте, кожна реконфігурація мережі — додавання або переміщення вузла — унеможливлює оперативне перестроювання просторової сітки, що є необхідним при дослідницькому характері роботи.

Wi-Fi, стандартизований як IEEE 802.11, усуває перелічені обмеження.

На таблиці 2.1 наведено порівняння RS-485/Modbus та Wi-Fi IEEE 802.11. Бездротова природа зв'язку дозволяє розміщувати вимірювальні вузли у довільній точці простору в межах зони покриття точки доступу без будь-якої кабельної інфраструктури. Зіркова топологія відносно точки доступу практично не обмежує кількість вузлів у межах пропускної здатності каналу. Наявна у більшості приміщень Wi-Fi-інфраструктура використовується як готова комунікаційна основа, що знижує стартові витрати на розгортання системи. Синхронізація часових міток здійснюється через NTP (Network Time Protocol) з точністю ±10 мс, чого достатньо для

аналізу теплових процесів із характерними часовими константами від десятків секунд до годин. Рівень завад у типовому приміщенні на частотах 2,4 ГГц є прийнятним для надійності доставки MQTT-пакетів понад 99 %.

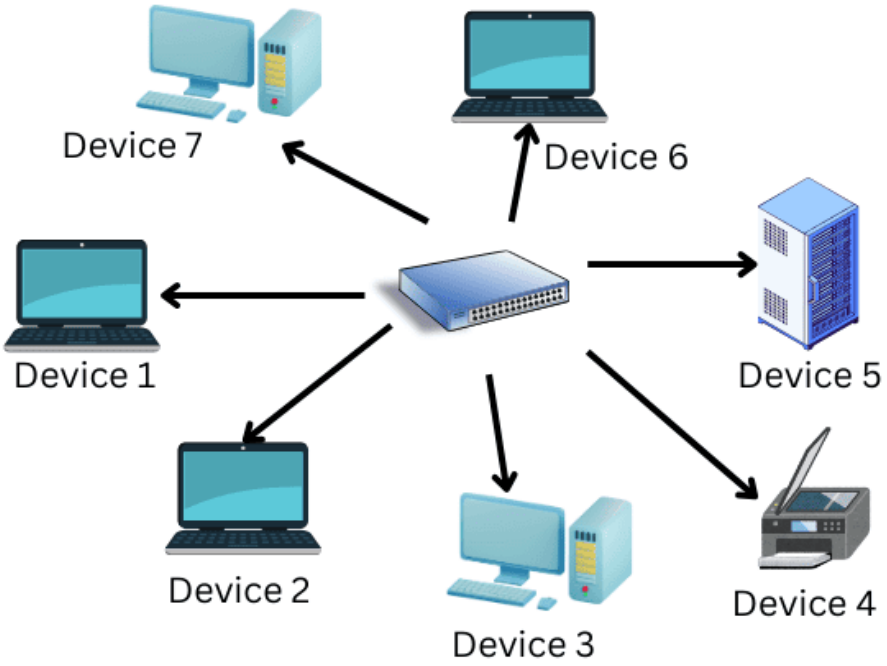


Рисунок 2.2 – Топологія зірки для мережі на основі мережі WiFi

Транспортний протокол прикладного рівня — MQTT (Message Queuing Telemetry Transport) — реалізує модель публікація-підписка (pub/sub) поверх TCP, що повністю розв'язує виробників даних (вузли) від споживачів (сервер, дашборд). Мінімальний розмір заголовка MQTT (2 байти) суттєво знижує накладні витрати порівняно з HTTP. Три рівні якості обслуговування QoS 0/1/2 дозволяють гарантувати доставку телеметричних пакетів (QoS 1) без дублювання при стабільному з'єднанні [9].

Таблиця 2.1 — Порівняльний аналіз RS-485/Modbus та Wi-Fi IEEE 802.11

Критерій	RS-485 / Modbus RTU	Wi-Fi IEEE 802.11 (ESP32)
Топологія	Шинна (лінійна), до 32 пристроїв без репітера	Зіркова через AP, кількість вузлів обмежена пропускною здатністю каналу
Позиціонування вузла	Фіксоване, обмежене трасою кабелю	Вільне у межах зони покриття Wi-Fi
Прокладання кабелю	Обов'язкове (екранована вита пара, монтажні короби)	Відсутнє
Реконфігурація мережі	Фізичне втручання в кабельну інфраструктуру	Зміна конфігурації прошивки або точки доступу
Максимальна довжина	До 1200 м при 9600 бод; 120 м при 100 кбод	До 100 м у приміщенні (залежить від AP та перешкод)
Протокол прикладного рівня	Modbus RTU / ASCII	MQTT поверх TCP/IP; альтернатива — HTTP POST
Синхронізація часу	Потребує окремого рішення (GPS, PTP)	NTP, точність ±1…10 мс
Захист від ЕМЗ	Висока (диференційна пара, синфазне придушення до ±12 В)	Середня; прийнятна у типовому приміщенні
Вартість розгортання	Висока (кабель, термінатори, RS-485 → USB перетворювачі)	Низька (наявний роутер Wi-Fi)
Масштабування	Обмежене топологією шини	Практично необмежене в межах AP

2.3 Апаратна платформа мікроконтролерного вузла

Мікроконтролерний вузол побудований на базі модуля ESP32 (зокрема ESP32-WROOM-32D або ESP32-DevKitC), що є апаратним розвитком

концепції ESP8266/ESP-01, розглянутої як початкова концептуальна основа. ESP32 містить двоядерний 32-розрядний процесор Xtensa LX6 із тактовою частотою до 240 МГц, 520 КБ SRAM, 4 МБ Flash-пам'яті, вбудований стек Wi-Fi IEEE 802.11 b/g/n та Bluetooth 4.2/BLE, два 12-розрядних АЦП, апаратну підтримку інтерфейсів I²C, SPI, UART, апаратний таймер реального часу та контролер ШІМ. Наявність двох ядер дозволяє виконувати задачу вимірювань на одному ядрі, а мережеві операції (Wi-Fi, MQTT) — на другому, що усуває взаємний вплив затримок мережевого стека на регулярність циклу вимірювань.

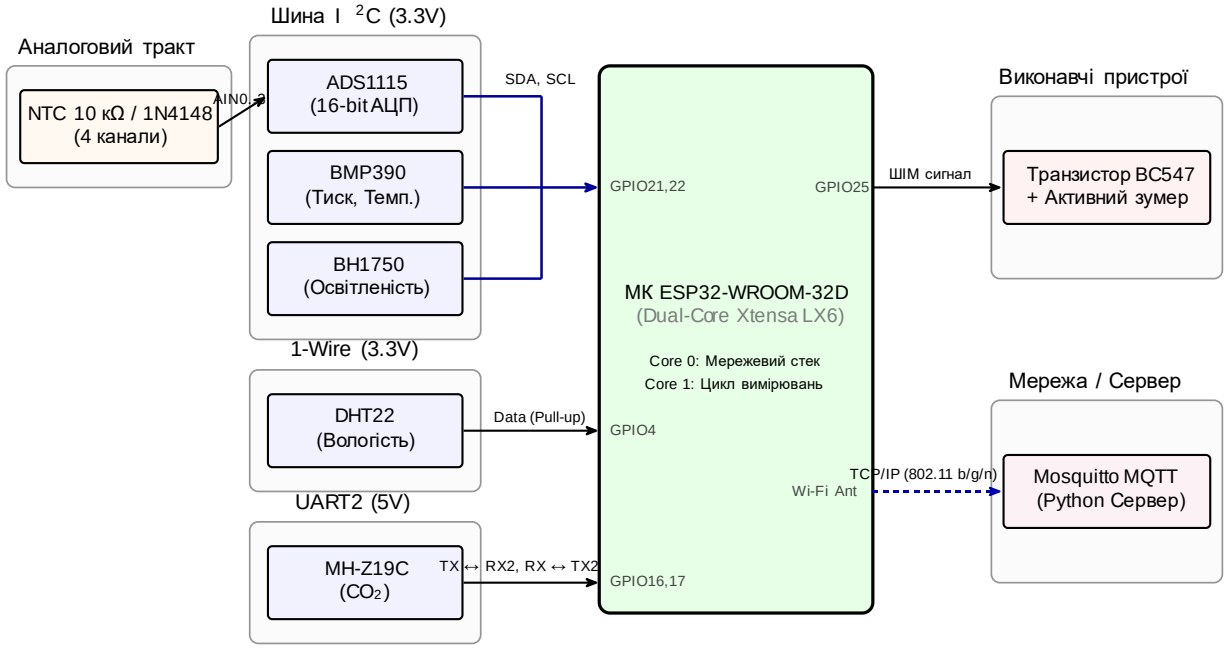


Рисунок 2.3– Структурна схема системи

Порівняно з ESP-01 (ESP8266), ESP32 має суттєво більший обсяг оперативної пам'яті (520 КБ проти 96 КБ), що дозволяє одночасно обслуговувати стек TCP/IP, MQTT-клієнт і буфери вимірювань без ризику переповнення пам'яті. Крім того, апаратні контролери I²C та UART на ESP32 звільняють процесорний час, який на ESP-01 витрачався на bit-banging

програмної реалізації цих протоколів. На ESP-01 шина I²C реалізується програмно на виводах GPIO0 (SDA) та GPIO2 (SCL), де підтягуючі резистори 4,7 кОм виконують подвійну функцію: забезпечують коректний рівень лінії I²C та фіксують GPIO0 у стані HIGH при завантаженні, що необхідно для нормального старту із Flash-пам'яті.

Живлення вузла здійснюється від мережевого адаптера 5 В (USB-типу) або від акумулятора Li-Ion/Li-Po через LDO-стабілізатор MCP1700-3302E (або AMS1117-3.3) із максимальним вихідним струмом 250–300 мА, що перекриває пікове споживання ESP32 при Wi-Fi-передачі (до 240 мА). На виході стабілізатора встановлюється електролітичний конденсатор 100–220 мкФ із низьким ESR для фільтрації пікових кидків струму при Wi-Fi-burst і керамічний конденсатор 100 нФ для придушення ВЧ-завад. Вхід стабілізатора також шунтується конденсатором 100 мкФ для стабілізації напруги акумулятора при пікових навантаженнях.

Для живлення вимірювального вузла передбачено дві конфігурації. У базовому варіанті використовується USB-адаптер 5 В (мережевий) з LDO-стабілізатором до 3,3 В. Для забезпечення безперервної роботи під час короткочасних вимкнень електромережі застосовується схема резервування з активним розв'язуванням джерел живлення (diode-ORing).

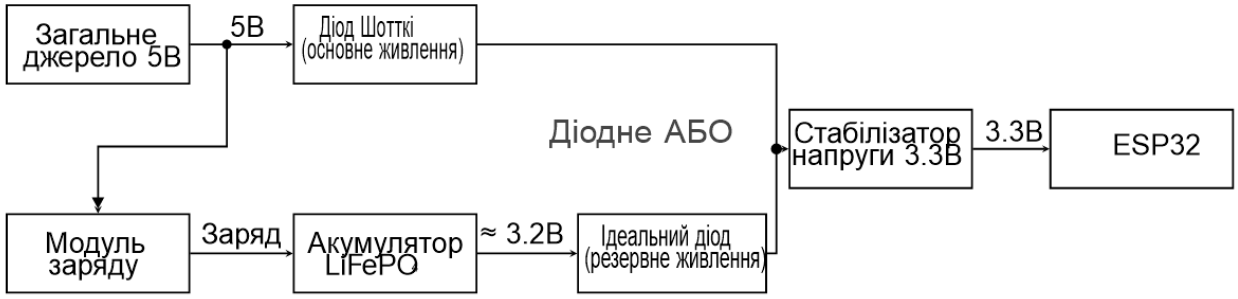


Рисунок 2.4 – Схема живлення із використанням акумуторного резервування

У запропонованій схемі (див. рис. 2.4) для розв'язки основного джерела 5В використано діод Шотткі з малим прямим падінням напруги. З боку резервного акумулятора $LiFePO_4$ (номіналом 3,2 В, ємністю 2000 мА·год) застосовано ідеальний діод на основі мікросхеми LM66100 (рис 2.5). Це рішення забезпечує падіння напруги лише на декілька десятків мілівольт, що значно перевершує будь-який звичайний діод, наприклад, при струмі 1 А падіння напруги становитиме близько 90 мВ замість типових 600 мВ на кремнієвому або 300 мВ на діоді Шотткі. Це дозволяє максимально використати ємність акумулятора, особливо при роботі ESP32 у пікових режимах (до 240 мА). Схема автоматично обирає джерело з вищою напругою: в нормальному режимі це 5 В від USB-адаптера, при його відключенні – напруга від акумулятора (~3,2…3,6 В).

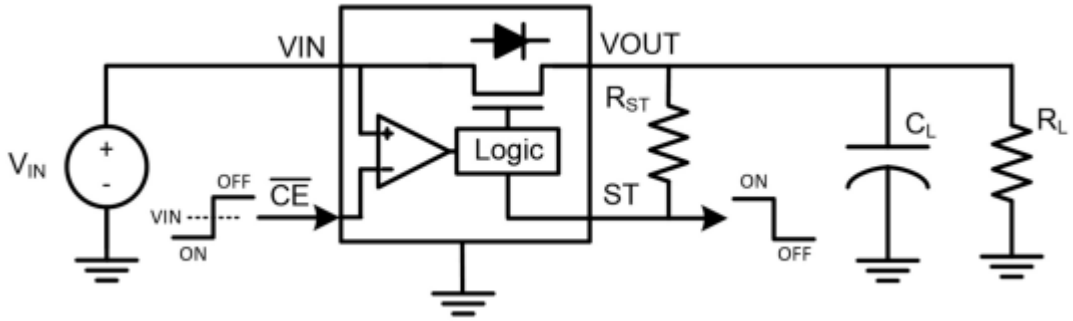


Рисунок 2.5 – Схема роботи мікросхеми із функцією «ідеального діода»

Для безпечної зарядки $LiFePO_4$ акумулятора використовується спеціалізований модуль на основі мікросхеми CN3058, налаштований на кінцеву напругу заряду 3,6 В (схема підключення акумулятора до мікросхеми представлена на рис. 2.6). Така схема забезпечує миттєве перемикання без втрати даних і дозволяє вузлу працювати від резервної батареї понад 8 годин у безперервному режимі – чого достатньо для переживання типових аварійних відключень.

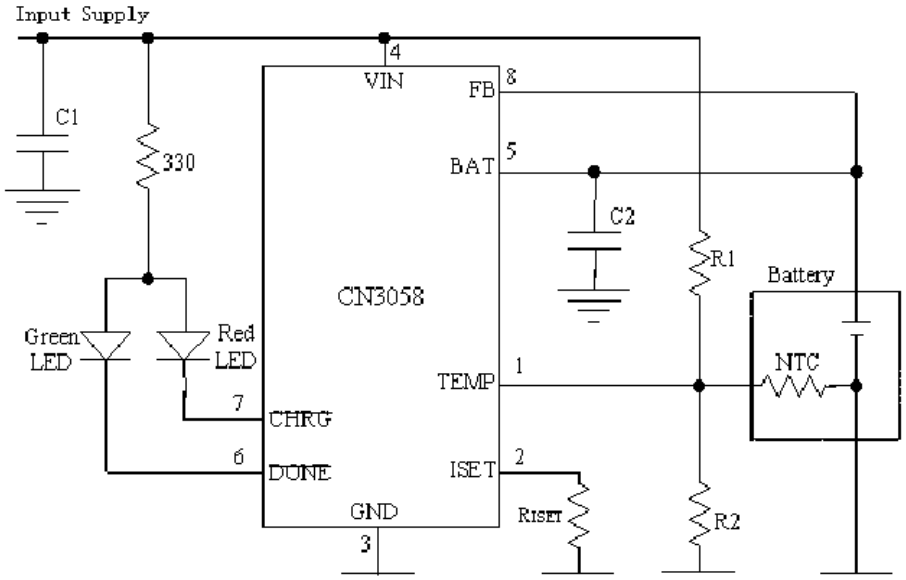


Рисунок 2.6 – Схема контролера зарадки акумулятора LiFePo4

Якщо ж ставиться завдання тривалої автономної роботи (наприклад, моніторинг у місцях без доступу до мережі), вузол переводиться в циклічний режим: пробудження кожні 60 с, вимірювання, передача MQTT, потім deep sleep. При такому режимі середнє споживання знижується до ~2 мА, що дозволяє працювати від того ж акумулятора 2000 мА·год близько 40 діб.

Для стаціонарних систем моніторингу з постійним підключенням до мережі живлення режим deep sleep не є обов'язковим, і вузли можуть працювати у безперервному режимі з інтервалом вимірювань 5 с.

2.4 Вимірювальний тракт температури на базі АЦП ADS1115

2.4.1 Принцип дії та характеристики ADS1115

ADS1115 — 16-розрядний сигма-дельта АЦП виробництва Texas Instruments із вбудованим програмованим підсилювачем (Programmable Gain Amplifier, PGA), 4-канальним мультиплексором входів, компаратором та цифровим інтерфейсом I²C. Принцип дії сигма-дельта перетворювача полягає

у надвисокочастотному надвибірковому квантуванні (oversampling) вхідного сигналу з подальшою цифровою фільтрацією та децимацією. На відміну від АЦП (рис. 2.7) послідовного наближення (SAR), сигма-дельта перетворювач інтегрує вхідний сигнал протягом усього часу перетворення, що забезпечує природне придушення завад на частотах, кратних частоті вибірки. Зокрема, при частоті вибірки 8 SPS (відліків за секунду) час перетворення одного відліку складає 125 мс, і завада промислової частоти 50 Гц (T = 20 мс) повністю входить у вікно інтеграції ціле число разів, придушуючись не менш ніж на 60 дБ.

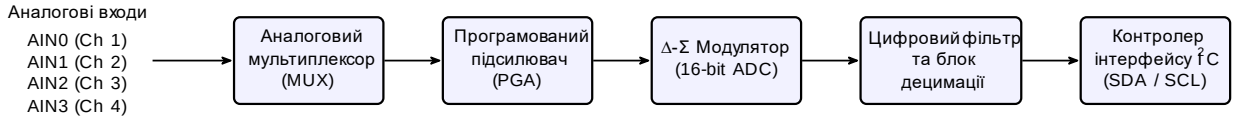


Рисунок 2.7 – Структрна схема АЦП ADS1115

Диференційний режим вимірювання є ключовою перевагою ADS1115 для задач прецизійного вимірювання температури. У диференційному режимі АЦП вимірює різницю напруг між двома входами пари (AIN0–AIN1 або AIN2–AIN3), усуваючи синфазні завади, однаково присутні на обох провідниках. Синфазне придушення (CMRR) досягає 100 дБ при частоті 50/60 Гц у налаштуванні PGA ±2,048 В. Це особливо важливо при просторово рознесених датчиках, з'єднаних довгими провідниками: наводки від Wi-Fi-модуля, мережевого блоку живлення та інших джерел ЕМЗ однаково впливають на обидва провідники та повністю компенсуються при відніманні.

Програмований підсилювач PGA дозволяє адаптувати діапазон вхідної напруги до сигналу первинного перетворювача. Для сигналу діодного датчика температури (0,35–0,55 В) оптимальним є режим PGA ±0,512 В, при якому один МЗР відповідає 15,625 мкВ. Для резистивного подільника на NTC-термісторі

(0,5–2,8 В) доцільно використовувати PGA ±4,096 В (МЗР = 125 мкВ). Вибір коефіцієнта підсилення здійснюється програмно записом відповідних бітів PGA[2:0] у конфігураційний регістр (адреса 0x01) перед кожним вимірюванням.

Взаємодія мікроконтролера з ADS1115 організована через шину I²C (Inter-Integrated Circuit). Специфікація ADS1115 підтримує режим fast mode (400 кГц), достатній для обміну даними при будь-якій підтримуваній частоті вибірки. Адреса пристрою визначається станом виводу ADDR: 0x48 (до GND), 0x49 (до VDD), 0x4A (до SDA), 0x4B (до SCL), що дозволяє підключити до чотирьох ADS1115 до однієї шини I²C і отримати до 16 температурних каналів на один вузол.

Процедура отримання одного відліку АЦП в одноразовому режимі (single-shot) включає такі кроки: запис 2-байтового конфігураційного слова у регістр 0x01 (вибір каналу, коефіцієнт PGA, бітова швидкість, запуск перетворення); очікування завершення перетворення (130 мс при 8 SPS або 8 мс при 128 SPS); зчитування 2-байтового знакового результату з регістра 0x00. Результат є числом у форматі доповняльного коду (two's complement) у діапазоні –32768…+32767. Для PGA ±2,048 В дійсна напруга обчислюється як V = code × 62,5 мкВ; для PGA ±0,512 В — V = code × 15,625 мкВ.

Важливою особливістю ADS1115 (табл. 2.2) є низьке енергоспоживання та можливість роботи в одноразовому режимі перетворення (Single-Shot Mode). У цьому режимі АЦП перебуває в стані зниженого споживання енергії та активується лише на час виконання вимірювання, після чого автоматично повертається до режиму очікування. Такий підхід є особливо доцільним для вузлів IoT-систем, що працюють від автономних джерел живлення, оскільки дозволяє суттєво зменшити середнє енергоспоживання без погіршення точності вимірювань. Крім того, можливість програмного налаштування частоти вибірки

забезпечує гнучкий вибір між швидкодією системи та рівнем шуму вимірювального тракту.

Таблиця 2.2 — Основні параметри АЦП ADS1115

Параметр	Значення / характеристика
Розрядність	16 біт (знакова, діапазон –32768…+32767)
Кількість каналів	4 одиночних або 2 диференційних (мультиплексор)
Напруга живлення	2,0…5,5 В; сумісний із рівнями 3,3 В та 5 В
Частота вибірки (SPS)	8, 16, 32, 64, 128, 250, 475, 860 вибірок/с (програмована)
Підсилювач PGA	±0,256; ±0,512; ±1,024; ±2,048; ±4,096; ±6,144 В
МЗР при PGA ±0,512 В	15,625 мкВ
МЗР при PGA ±2,048 В	62,5 мкВ
ENOB при 8 SPS	~15,7 біт
Синфазне придушення CMRR	100 дБ при 50/60 Гц (типове значення)
Інтерфейс	I²C, підтримка fast mode 400 кГц
Адреси I²C	0x48…0x4B (4 варіанти через вивід ADDR)
Струм споживання (активний)	150 мкА (типове значення)
Температурний дрейф (FS)	±0,0025 % / °C (максимальне значення)
Режими роботи	Безперервне перетворення (CC) / Одноразове (SS)
Вбудований компаратор	З програмованими порогами, режими Traditional та Window

Частота дискретизації обирається виходячи з компромісу між часовим розрізненням і рівнем шуму. Теплові процеси у приміщенні мають характерні часові константи від десятків секунд (дрібномасштабна турбуленція конвективних потоків) до годин (встановлення стаціонарного теплового поля при зміні зовнішньої температури). Для їх адекватного відстеження достатньо частоти вибірки 1–8 SPS з боку АЦП; при цьому великий час інтеграції (125 мс при 8 SPS) забезпечує максимальне придушення шумів та мережевого фону 50 Гц. Для забезпечення рівномірного просторово-часового зрізу весь масив вузлів опитується з інтервалом не більше 5 с — це дозволяє відстежувати зміни температури зі швидкістю до 0,05–0,1 °C/с без артефактів у реконструйованому часовому ряді.

2.4.2 Схемотехнічна реалізація підключення ADS1115

Схема підключення ADS1115 до вузла передбачає розв'язку живлення та захист аналогових входів. Вивід VDD АЦП підключається до шини 3,3 В через низькочастотний дросель індуктивністю 10 мкГн та конденсатор 10 мкФ паралельно — ця LC-клітина зменшує вплив цифрових перехідних процесів мікроконтролера та Wi-Fi-модуля на аналоговий тракт АЦП. Додатково встановлюється керамічний конденсатор 100 нФ безпосередньо біля виводу VDD для придушення ВЧ-складових. Вивід GND з'єднується з аналоговою землею; аналогова та цифрова землі платформи з'єднуються в одній зірковій точці, що унеможливлює протікання цифрових струмів через аналоговий потенціал.

Кожен аналоговий вхід AIN0–AIN3 захищається послідовним резистором 1 кОм та паралельним конденсатором 10 нФ до GND, що утворює RC-фільтр нижніх частот із частотою зрізу ≈ 15,9 кГц. Цей фільтр

придушує ВЧ-завади, наведені на з'єднувальних провідниках до датчиків, і захищає вбудовані захисні діоди ADS1115 від перевищення допустимого струму при виплесках напруги. Вивід ALRT/RDY при необхідності підключається до GPIO мікроконтролера для організації апаратного переривання по готовності даних, що виключає необхідність циклічного опитування (polling) регістра готовності і звільняє процесорний час (рис 2.8).

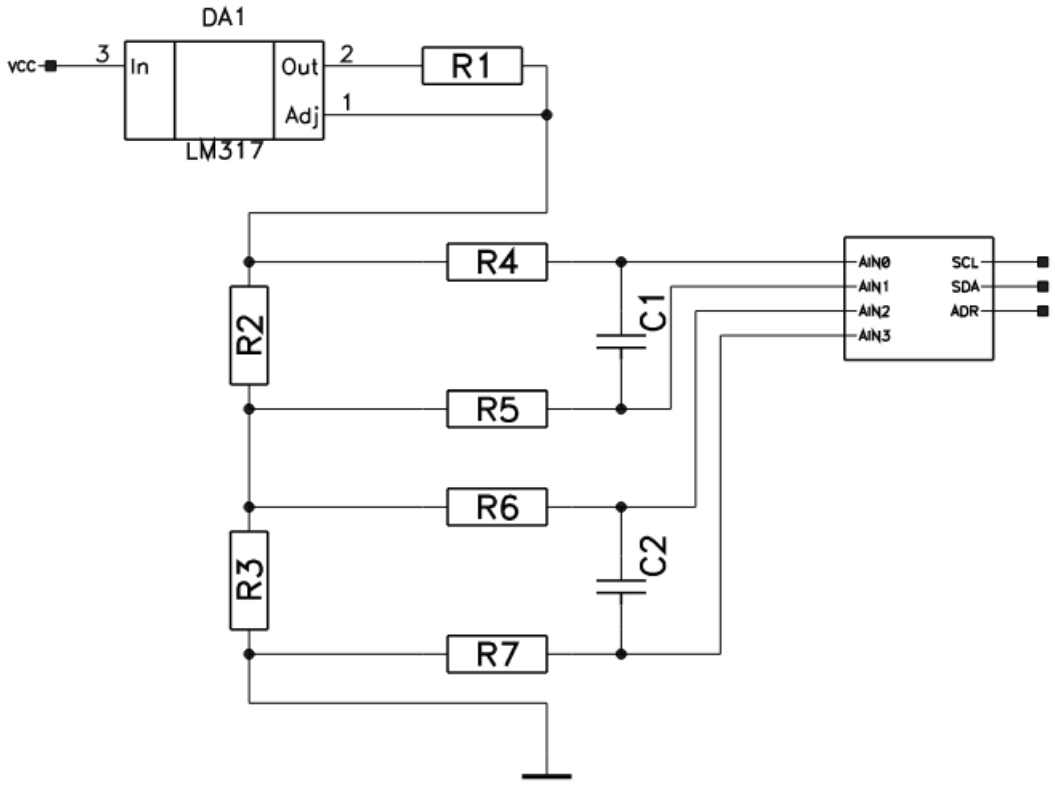


Рисунок 2.8 – Схема підключення терморезисторів до диференційного входу ADS1115

Провідники аналогових входів від датчиків до ADS1115 на друкованій платі виконуються якомога коротшими (не більше 20–30 мм), прокладаються паралельно і не перетинаються з лініями шини I²C, лініями живлення та Wi-Fi-антеною. При виносних датчиках, підключених довгими провідниками (десятки сантиметрів), слід застосовувати скручену пару для кожного каналу

і підключати їх до АЦП у диференційному режимі — це максимально використовує перевагу CMRR при придушенні наведених завад.

2.5 Первинні перетворювачі температури

2.5.1 NTC-термістори

NTC-термістор (Negative Temperature Coefficient) є напівпровідниковим резистивним елементом із від'ємним температурним коефіцієнтом опору. Виготовляється методом спікання оксидів металів (Mn, Co, Ni, Cu) у заданих пропорціях. Температурна залежність опору NTC-термістора описується рівнянням Стейнгарта-Харта:

$$1/T = A + B\cdot\ln(R) + C\cdot(\ln(R))^3, \qquad (2.1)$$

де T — абсолютна температура в кельвінах, R — опір у омах, A, B, C — константи, що визначаються калібруванням. Для практичних розрахунків у діапазоні 0–60 °C достатньою є спрощена β-модель:

$$R(T) = R_{25} \cdot \exp(B \cdot (1/T - 1/T_{25})), \qquad (2.2)$$

де R_{25} — опір при 25 °C (для стандартного NTC 10 кОм), B — характеристична температура матеріалу (типово 3000–4500 К). При B = 3950 К чутливість складає приблизно –4,4 %/°C при 25 °C.

Для перетворення опору термістора у напругу, придатну для оцифрування, застосовується резистивний подільник напруги. Термістор R_NTC і опорний резистор R_0 = 10 кОм (точність 0,1 %, ТКО ≤ 25 ppm/°C, наприклад Vishay Dale CMF55) утворюють подільник, напруга на виході якого:

$$V\_OUT = V\_CC \cdot R_0 / (R_0 + R\_NTC(T)). \qquad (2.3)$$

При V_CC = 3,3 В і R_0 = 10 кОм напруга при 0 °C складає близько 0,95 В, при 25 °C — 1,65 В, при 50 °C — близько 2,2 В. Зміна напруги по всьому

діапазону вимірювань (~1,25 В) відповідає чутливості ~25 мВ/°С при 25 °С. При використанні PGA ADS1115 у режимі ±2,048 В (МЗР = 62,5 мкВ) теоретична роздільна здатність складає 62,5 мкВ / 25 мВ/°С ≈ 0,0025 °С. На практиці з урахуванням шуму АЦП (ENOB ≈ 15,7 біт при 8 SPS) ефективна роздільна здатність становить близько 0,003–0,005 °С.

Нелінійність характеристики NTC-термістора усувається програмним шляхом: мікроконтролер або сервер обчислює дійсну температуру за β-формулою або за таблицею Стейнгарта-Харта, коефіцієнти якої отримані при калібрувальних вимірюваннях у кількох реперних точках (0, 20, 40, 60 °С) з еталонним термометром. Точність такої лінеаризації забезпечує залишкову нелінійність не більше ±0,1 °С у діапазоні 0–60 °С.

2.5.2 Кремнієвий p-n-діод як датчик температури

Кремнієвий діод у режимі прямого зміщення малим струмом використовує залежність прямої напруги V_F від температури при постійному прямому струмі I_F. Фізичний механізм — температурна залежність ширини забороненої зони кремнію та зворотного струму насичення p-n-переходу. Для ідеального p-n-переходу при малому струмі:

$$(I_F \ll I_{sat} \cdot \exp(qV_F/kT)): V_F \approx V_{G0} - (k/q) \cdot T \cdot \ln(C/I_F), \quad (2.4)$$

де V_{G0} — екстрапольована напруга ширини забороненої зони при 0 К (~1,205 В для кремнію), C — константа матеріалу. Для стандартних діодів типу 1N4148 або 1N4007 при I_F = 10 мкА температурний коефіцієнт V_F становить ≈ –2,05 мВ/°С у діапазоні –20…+70 °С. Відхилення від лінійності в цьому діапазоні не перевищує ±0,3 °С без будь-якої апаратної або програмної корекції — суттєво краща лінійність порівняно з NTC-термістором. Також на таблиці 2.3 наведено порівняння первинних перетворювачів температури.

Для забезпечення стабільного струму I_F через діод незалежно від напруги на аноді і температури навколишнього середовища застосовується джерело опорного струму. Найбільш зручним у практичній реалізації є мікросхема LM334Z — трьохвивідне програмоване джерело струму, вихідний струм якого задається одним зовнішнім резистором:

$$I\_SET = 67\ мВ / R\_SET. \quad (2.5)$$

При R_SET = 6,8 кОм струм I_SET = 9,85 мкА ≈ 10 мкА. Температурний коефіцієнт LM334Z становить +0,336 %/°C (позитивний PTAT — proportional to absolute temperature), що при застосуванні як джерела струму для діодного датчика вносить незначну нелінійність, яка може бути скомпенсована програмно або прийнята як несуттєва для досягнутої роздільної здатності.

При I_F = 10 мкА і PGA ADS1115 у режимі ±0,512 В (МЗР = 15,625 мкВ) роздільна здатність вимірювання V_F складає: ΔT = 15,625 мкВ / 2050 мкВ/°C ≈ 0,0076 °C. Для усунення паразитних термоЕРС у вимірювальному колі з'єднувальні провідники від джерела струму до діода та від діода до входів ADS1115 виконуються з одного матеріалу (мідь) без розривів і паяних з'єднань у зоні вимірювання. При диференційному підключенні до ADS1115 (анод діода — на AIN+, катод — на AIN–) паразитні термоЕРС в провідниках однаково впливають на обидва входи і компенсуються.

Для підвищення надійності функціонування системи передбачено програмний контроль коректності отриманих даних на кожному етапі їх обробки. Після надходження вимірювальної інформації виконується перевірка повноти пакета даних, допустимості значень параметрів та часової узгодженості вимірювань.

Таблиця 2.3 — Порівняння первинних перетворювачів температури

Параметр	**NTC-термістор (10 кОм, B=3950 K)**	**p-n-діод (1N4148, I_F=10 мкА)**
Чутливість при 25 °C	~–4,4 %/°C (по опору); ~25 мВ/°C (напруга подільника)	~–2,05 мВ/°C (пряма напруга)
Лінійність у діапазоні 0–60 °C	Низька; необхідна β-лінеаризація або таблиця S-H	Висока; залишкова нелінійність < ±0,3 °C
Схема включення	Резистивний подільник напруги (R_0 = 10 кОм, 0,1 %)	Джерело опорного струму LM334Z, R_SET = 6,8 кОм
Оптимальний режим PGA	±2,048 В або ±4,096 В	±0,512 В
МЗР у обраному режимі PGA	62,5 мкВ	15,625 мкВ
Розрахункова роздільна здатність	~0,0025 °C (теоретична)	~0,0076 °C (теоретична)
Ефективна роздільна здатність	~0,005 °C при 8 SPS	~0,008 °C при 8 SPS
Стабільність	Задовільна; старіння ≤ 0,1 % за рік	Висока; стабільна при незмінному I_F
Вартість	Мінімальна (~0,10 USD)	Мінімальна (~0,05 USD)
Лінеаризація	Обов'язкова програмна	Не потрібна або мінімальна

2.6 Датчики додаткових параметрів мікроклімату

2.6.1 Датчик атмосферного тиску BMP390

Атмосферний тиск вимірюється датчиком BMP390 виробництва Bosch Sensortec із підключенням по шині I²C (адреса 0x77 при SDO до GND або 0x76 при SDO до VDD). BMP390 є п'єзорезистивним датчиком тиску із компенсацією температурної похибки. Рівень шуму вимірювання тиску у режимі нормального енергоспоживання (стандартний oversampling) складає 0,9 Па RMS, що відповідає еквівалентному висотному шуму менше 0,07 м. Абсолютна точність датчика в діапазоні температур 0–65 °C становить ±50 Па. Вбудований термометр BMP390 (точність ±0,5 °C) використовується для внутрішньої компенсації показань тиску; температура з BMP390 також передається на сервер як додатковий канал і може брати участь у реконструкції температурних даних.

Атмосферний тиск є параметром, що практично однаково розподілений між усіма вузлами однієї будівлі: різниця тиску між найнижчою та найвищою точками будівлі висотою 10 м становить $\rho \cdot g \cdot h \approx 1{,}2$ кг/м³ × 9,81 м/с² × 10 м ≈ 118 Па ≈ 1,18 гПа. Ця відома залежність безпосередньо використовується модулем діагностики сервера: медіана показань BMP390 по всій мережі вузлів є надійною оцінкою поточного атмосферного тиску, і будь-яке відхилення від медіани понад ±5 гПа однозначно вказує на відмову датчика у конкретному вузлі. Таким чином, показання BMP390 з різних вузлів є взаємно перевірюваними без жодних складних алгоритмів, що робить тиск одним із найбільш надійних параметрів мережі.

2.6.2 Датчик відносної вологості DHT22

Відносна вологість вимірюється датчиком DHT22 (AM2302) із підключенням до GPIO ESP32 за однопровідним послідовним протоколом. Датчик поєднує ємнісний чутливий елемент вологості та резистивний термометр у єдиному корпусі. Діапазон вимірювання вологості: 0–100 % відносної вологості, точність ±2 %, роздільна здатність 0,1 %. Діапазон вимірювання температури: –40…+80 °C, точність ±0,5 °C. Інтервал опитування — не частіше одного разу на 2 с; протокол передбачає вбудовану контрольну суму, що дозволяє виявляти помилки передачі. Бібліотека DHT автоматично повертає NaN при невдалому зчитуванні або помилці контрольної суми, що однозначно ідентифікується модулем діагностики сервера як відмова датчика.

Відносна вологість є одним із двох параметрів (поряд із CO_2), відновлення яких є прямою задачею системи. Функціональний зв'язок між вологістю, температурою та концентрацією CO_2 існує завдяки таким фізичним механізмам: при нагріванні повітря з постійним вмістом водяної пари відносна вологість зменшується за законом Клаузіуса-Клапейрона; присутність людей у приміщенні одночасно підвищує CO_2 і вологість (дихання); роботи системи вентиляції змінюють CO_2 і вологість скорельовано. Статистично виявлена модель цих залежностей дозволяє оцінювати вологість на основі температурних каналів, CO_2 та зовнішніх умов навіть при відмові DHT22.

2.6.3 Датчик концентрації CO_2 MH-Z19C

Концентрація вуглекислого газу вимірюється датчиком MH-Z19C — готовим NDIR-модулем (Non-Dispersive InfraRed, недисперсійний інфрачервоний аналіз) виробництва Winsen Electronics. Принцип дії NDIR

ґрунтується на поглинанні інфрачервоного випромінювання молекулами CO_2 на довжині хвилі 4,26 мкм: оптична схема модуля порівнює поглинання в аналітичній та референтній кюветах, що дозволяє компенсувати дрейф джерела IR та старіння оптики. Діапазон вимірювання: 400–5000 ppm, точність ±(50 + 5 %) ppm, час відгуку $T_{90} \approx 60$ с. Інтерфейс — UART (9600 бод, 8N1) або аналоговий PWM-вихід; у даній роботі використовується UART для отримання числового значення без додаткової конвертації.

Датчик підключається до апаратного порту UART2 ESP32: GPIO16 (RX2) ← TX датчика, GPIO17 (TX2) → RX датчика. Живлення 5 В для MH-Z19C забезпечується безпосередньо від лінії живлення адаптера (до стабілізатора 3,3 В), оскільки модуль не сумісний із живленням 3,3 В. Рівні UART-сигналів (TX/RX) за специфікацією MH-Z19C працюють коректно при 3,3 В на боці ESP32 завдяки тому, що вхідний поріг HIGH модуля не перевищує $0{,}7 \cdot VCC = 3{,}5$ В.

Вбудований алгоритм автокалібрування ABC (Automatic Baseline Calibration) датчика розрахований на застосування у звичайних приміщеннях із регулярним провітрюванням, де концентрація CO_2 щонайменше один раз на тиждень знижується до фонового рівня (~400 ppm). У лабораторних умовах із нестандартним рівнем CO_2 ABC може вносити систематичну похибку, тому сервер за потреби вимикає його через MQTT-команду на вузол. Концентрація CO_2 є важливим параметром як для задачі реконструкції (завдяки кореляції з вологістю і температурою, зумовленій активністю людей), так і для функції сигналізації: при перевищенні порогу 1000–2000 ppm сервер надсилає команду beep на зумер вузла.

2.6.4 Датчик освітленості BH1750FVI

Освітленість вимірюється датчиком BH1750FVI виробництва ROHM Semiconductor із підключенням по шині I²C (адреса 0x23 при ADDR до GND або 0x5C при ADDR до VDD, що дозволяє підключити два датчики до однієї шини). BH1750FVI містить фотодіод, підсилювач, 16-розрядний АЦП і вбудований процесор компенсації, що забезпечує вихідне значення безпосередньо у люксах без зовнішньої обробки. Режим H-resolution2 забезпечує роздільну здатність 0,5 лк у діапазоні 1–65535 лк і час вимірювання близько 120 мс. Датчик стійкий до впливу інфрачервоного випромінювання завдяки вбудованому IR-фільтру.

Освітленість має функціональний зв'язок із тепловим навантаженням приміщення через кілька механізмів: теплове випромінювання ламп розжарювання та галогенних джерел (ефективність 5–15 % видимого світла, решта — теплота); поглинання сонячного проміння через вікна; зворотній зв'язок між присутністю людей (що вмикають освітлення) та CO_2/вологістю. Хоча цей зв'язок менш прямий, ніж між вологістю та температурою, наявність каналу освітленості у багатовимірному векторі спостережень підвищує повноту моделі і може покращити якість реконструкції температурних даних у умовах змінного освітлення. Важливою особливістю BH1750FVI є наявність внутрішнього перетворювача сигналу та цифрового інтерфейсу I²C, що усуває необхідність використання додаткових аналогових схем обробки сигналу. Завдяки цьому зменшується вплив електромагнітних завад і похибок, пов'язаних із передачею аналогових сигналів. Крім того, датчик підтримує декілька режимів вимірювання з різною роздільною здатністю та швидкістю оновлення даних, що дозволяє оптимізувати його роботу залежно від вимог конкретного застосування та забезпечувати

стабільний контроль рівня освітленості в режимі реального часу. На таблиці 2.4 наведено характеристику первинних перетверювачів вузла.

Таблиця 2.4 — Зведена характеристика первинних перетворювачів вузла

Параметр	**Датчик / компонент**	**Інтерфейс**	**Діапазон**	**Точність**
Температура (4 канали)	NTC 10 кОм або 1N4148 → ADS1115	I²C 0x48	–40…+85 °C	±0,1 °C
Атм. тиск	BMP390 (Bosch)	I²C 0x77	300…1250 гПа	±50 Па (абс.)
Температура (BMP)	BMP390 (вбудований)	I²C 0x77	–40…+85 °C	±0,5 °C
Відносна вологість	DHT22 (AM2302)	Single-wire GPIO4	0…100 %	±2 %
Освітленість	BH1750FVI (ROHM)	I²C 0x23	1…65535 лк	±20 %
CO_2	MH-Z19C (NDIR)	UART2 (GPIO16/17)	400…5000 ppm	±(50+5 %) ppm

2.7 Принципова електрична схема вимірювального вузла

Принципова електрична схема вузла поєднує блок живлення та стабілізації, мікроконтролерний модуль ESP32, зовнішній АЦП ADS1115 із чотирма первинними перетворювачами температури, датчик тиску BMP390, датчик вологості DHT22, датчик освітленості BH1750FVI, датчик CO_2 MH-Z19C та зумер для акустичної сигналізації. Схема побудована виходячи з

принципів мінімізації взаємних завад між аналоговим вимірювальним трактом і цифровими та RF-компонентами.

Шина I²C об'єднує ADS1115, BMP390 та BH1750FVI. На ESP32 вона реалізована на апаратному контролері I²C: GPIO21 (SDA) та GPIO22 (SCL). Підтягуючі резистори 4,7 кОм (для частоти 100 кГц) або 2,2 кОм (для 400 кГц) встановлюються один раз на шину і не дублюються для кожного пристрою. Для BMP390 вивід SDO підтягується до GND резистором 10 кОм для фіксації адреси 0x77; вивід CSB підтягується до VDD для вибору I²C-режиму (на відміну від SPI). Для BH1750 вивід ADDR залишається підключеним до GND (адреса 0x23). Вивід ADDR ADS1115 підключається до GND (адреса 0x48). У разі застосування двох ADS1115 другий отримує адресу 0x49 (ADDR до VDD) і підключається до тієї самої шини I²C.

DHT22 підключається до GPIO4 ESP32 через підтягуючий резистор 10 кОм до шини 3,3 В. Довжина провідника від датчика до мікроконтролера не повинна перевищувати 20–30 м; при більших відстанях необхідне зниження швидкості обміну або застосування буферного елемента. Конденсатор 100 нФ між виводами живлення датчика та GND стабілізує напругу під час тактових імпульсів протоколу.

UART-інтерфейс MH-Z19C підключається до порту UART2 ESP32: GPIO16 (RX2) від TX датчика, GPIO17 (TX2) до RX датчика. Живлення 5 В для модуля MH-Z19C підводиться безпосередньо від вхідної лінії 5 В до LDO-стабілізатора, що забезпечує достатній рівень напруги незалежно від падіння на стабілізаторі. Конденсатор 100 мкФ між вводом 5 В і GND поблизу датчика компенсує імпульсні піки струму при роботі IR-випромінювача модуля.

Зумер для акустичної сигналізації підключається до GPIO25 ESP32 через транзисторний ключ BC547 (або IRLML2244 для MOSFET-варіанту). GPIO25 ESP32 забезпечує максимальний вихідний струм 40 мА, тоді як активний п'єзозумер (наприклад, HY-12095) потребує 30–100 мА при напрузі 3–5 В. Транзисторний ключ підключається так: база BC547 — до GPIO25 через резистор 1 кОм (обмеження базового струму); емітер — до GND; колектор — до негативного виводу зумера; позитивний вивід зумера — до шини 3,3 В або 5 В. Управління зумером здійснюється через ШІМ-контролер ESP32 (функція ledcWriteTone), що дозволяє задавати частоту та тривалість сигналу без блокування процесора.

Компонування двосторонньої SMD-плати розмірами 60×40 мм підпорядковується таким правилам. Аналоговий блок (ADS1115, первинні перетворювачі, захисні RC-фільтри) розміщується у одному квадранті плати з максимально короткими провідниками аналогових входів (≤20 мм). Цифровий блок (ESP32, BH1750, BMP390, роз'єми UART для MH-Z19C та DHT22) розміщується в протилежному квадранті. Між ними проходить зазор полігону без металізації шириною 1–2 мм, що зменшує ємнісні наводки між блоками. Аналогова і цифрова землі утворюють суцільні полігони у своїх зонах та з'єднуються єдиним містком у зірковій точці поблизу LDO-стабілізатора. Такий підхід забезпечує мінімальний імпеданс повернення для цифрових струмів у межах цифрового полігону і виключає їхнє проходження через аналоговий полігон.

2.8 Просторова конфігурація мережі вимірювальних вузлів

2.8.1 Принципи формування просторової сітки

Просторова конфігурація сенсорної мережі є одним із ключових проектних рішень, що безпосередньо визначає якість відновленого багатовимірного поля та достовірність виявлених залежностей між параметрами. Основним критерієм при формуванні сітки є умова Найквіста у просторовій області: середня відстань між сусідніми вузлами не повинна перевищувати половини характеристичної кореляційної довжини відповідного фізичного поля. Для температурного поля у приміщенні кореляційна довжина залежить від режиму конвекції і типово складає 0,5–2 м у горизонтальному напрямку та 0,3–1 м у вертикальному (через стратифікацію). Для поля CO_2 та вологості кореляційна довжина є більшою через повільне молекулярне перемішування і типово складає 1–3 м. Виходячи з найменшої кореляційної довжини (≈0,5 м для вертикального температурного профілю), оптимальна відстань між сусідніми вузлами у вертикальному напрямку — не більше 1,0–1,5 м.

Надмірна щільність сітки підвищує кореляцію між сусідніми вузлами і знижує приріст інформаційної змістовності кожного додаткового вузла. З іншого боку, рідка сітка може пропустити важливі структури поля. Оптимальна конфігурація забезпечує максимальне охоплення фізично значущих зон: пристінної зони теплообміну, зони підйому нагрітих повітряних мас уздовж стін, зони конвективних потоків у центрі приміщення та рівня робочої зони (1,2 м) як зони присутності людей.

Додатковою вимогою є забезпечення просторової реплікації: наявність двох вузлів у схожих умовах (наприклад, два пристінних вузла на однаковій висоті, але в різних горизонтальних позиціях) дозволяє розрізняти горизонтальні градієнти від вертикальних і підвищує надійність міжвузлових алгоритмів реконструкції.

2.8.2 Конфігурація вузлів у досліджуваному приміщенні

Запропонована конфігурація розрахована на приміщення розмірами 5×4 м із висотою стелі 2,7 м. Система координат вводиться таким чином: вісь X — перпендикулярно від досліджуваної стіни вглиб приміщення; вісь Y — вздовж стіни в горизонтальному напрямку; вісь Z — вертикально від рівня підлоги. Сенсорна сітка формується на основі чотирьох значень координати X (0,05; 0,75; 2,0; 3,5 м), трьох значень Z (0,3; 1,2; 2,2 м) та двох значень Y (1,0 і 2,5 м). Реалізована конфігурація з 12 вузлів наведена у таблиці 2.5.

Таблиця 2.5 — Просторові координати вузлів сенсорної мережі

Вузол	X (м)	Y (м)	Z (м)	Фізична позиція	Призначення
N-1	0,05	1,0	0,3	На стіні, нижній ярус	Пристінний нижній профіль
N-2	0,05	1,0	1,2	На стіні, середній ярус	Пристінний середній профіль
N-3	0,05	1,0	2,2	На стіні, верхній ярус	Пристінний верхній профіль
N-4	0,75	1,0	0,3	0,75 м від стіни, нижній	Зона конвекції, нижній
N-5	0,75	1,0	1,2	0,75 м від стіни, середній	Зона конвекції, середній
N-6	0,75	1,0	2,2	0,75 м від стіни, верхній	Зона конвекції, верхній
N-7	2,0	1,0	0,3	2 м від стіни, нижній	Середина приміщення, нижній
N-8	2,0	1,0	1,2	2 м від стіни, середній	Рівень робочої зони
N-9	2,0	1,0	2,2	2 м від стіни, верхній	Середина приміщення, верхній
N-10	3,5	1,0	1,2	Центр приміщення	Еталонний вузол
N-11	0,05	2,5	1,2	Стіна, бічна позиція, середній ярус	Горизонтальна реплікація стіни
N-12	2,0	2,5	1,2	2 м від стіни, бічна позиція	Горизонтальна реплікація центру

Вузли N-1, N-2, N-3 утворюють вертикальну колонку безпосередньо на стіні і призначені для вимірювання температури пристінного шару повітря на трьох рівнях. Ця колонка є основним джерелом даних для оцінки вертикального температурного профілю стіни та інтенсивності пристінної конвекції. Вузли N-4, N-5, N-6 формують аналогічну колонку на відстані 0,75 м від стіни — у зоні максимальних горизонтальних теплових градієнтів при нагріванні стіни. Вузли N-7, N-8, N-9 розміщені на відстані 2 м від стіни і характеризують теплове поле в середній частині приміщення. Вузол N-10 у центрі приміщення на середній висоті виконує роль еталонного вузла для нормалізації вимірювань та перевірки міжвузлових залежностей. Вузли N-11 та N-12 забезпечують горизонтальну реплікацію: наявність двох пристінних вузлів (N-2 і N-11) та двох вузлів на відстані 2 м (N-8 і N-12) на однаковій висоті дозволяє відрізнити горизонтальні градієнти вздовж стіни від шуму та відмов датчиків.

2.8.3 Фізичне кріплення та теплова ізоляція вузлів

Кріплення пристінних вузлів (N-1, N-2, N-3, N-11) виконується термоклеєм або кронштейнами безпосередньо на поверхні стіни. Між корпусом вузла та стіною обов'язково розміщується термоізоляційна підкладка товщиною 3–5 мм із вспіненого поліетилену або натурального корку. Це запобігає кондуктивному тепловому шунтуванню: без підкладки температура датчика буде усередненою між температурою стіни та повітря, що спотворює вимірювання обох величин. Термічний опір підкладки товщиною 4 мм із вспіненого поліетилену (коефіцієнт теплопровідності $\lambda \approx 0{,}04$ Вт/(м·К)) складає $R = d/\lambda \approx 0{,}1$ м²·К/Вт — достатньо для фактичного

розривання теплового контакту між датчиком і стіною на характерних часах вимірювань.

Вузли, підвішені у повітрі (N-4–N-9, N-12), кріпляться до тонких нейлонових ниток або вуглепластикових стрижнів діаметром 1–2 мм. Низька теплопровідність нейлону ($\lambda \approx 0{,}25$ Вт/(м·К)) і малий поперечний переріз стрижня зводять кондуктивне відведення тепла від датчика до несучих конструкцій до незначного рівня. Провід живлення та лінії I^2C, що підводяться до вузла, виконуються найтоншими допустимими провідниками (AWG28–AWG30) і спрямовуються паралельно несучому стрижню для мінімізації їхнього впливу на теплову ізоляцію вузла.

Датчики температури у складі кожного вузла захищаються радіаційним екраном із перфорованої алюмінієвої фольги (radiation shield). Екран являє собою порожнистий циліндр або скриньку з кількома концентричними шарами із відбивальною поверхнею, орієнтованою назовні. Висока відбивальна здатність алюмінію ($\varepsilon \approx 0{,}05$ у ІЧ-діапазоні) зменшує поглинання прямого та відбитого теплового випромінювання від ламп і сонця в 15–20 разів порівняно із незахищеним датчиком. Перфорація екрана забезпечує вільну конвекцію повітря через датчик, що необхідно для його термічного рівноваги з навколишнім повітрям. Розмір перфорацій 2–3 мм є компромісом між провітрюваністю та екрануванням від горизонтального потоку теплого або холодного повітря.

2.9 Надлишковість вимірювань та стратегія реконструкції даних

Кожен з 12 вузлів формує вектор спостережень із дев'яти компонент: температура T_1, T_2, T_3, T_4 (чотири канали ADS1115), тиск P, температура T_BMP (BMP390), вологість H (DHT22), освітленість L (BH1750) та концентрація CO_2 (MH-Z19C). У одному синхронному часовому зрізі вся

мережа 12 вузлів формує матрицю спостережень розміром 12×9 = 108 числових значень. Ця матриця є основою для задачі виявлення прихованих залежностей та реконструкції.

Функціональні залежності між параметрами розподіляються на два класи. До першого класу належать залежності з відомою фізичною природою: атмосферний тиск практично рівний у всіх вузлах (гідростатичний закон); відносна вологість і температура пов'язані через рівняння Клаузіуса-Клапейрона при незмінному вмісті водяної пари; CO_2 і вологість зростають корельовано при присутності людей. До другого класу належать залежності, що виявляються статистично на достатньо довгих рядах спостережень і можуть не мати простого аналітичного виразу, але є стійкими і відтворюваними. Алгоритми сервера (кросвалідація з виключенням одного об'єкта LOO-CV, кригінг, матричне завершення методом ALS) виявляють обидва класи залежностей і будують на їхній основі прогностичну модель.

При відмові датчика будь-якого параметра у будь-якому вузлі сервер застосовує відповідний алгоритм реконструкції. При відмові температурного каналу T_i у вузлі k застосовується просторова інтерполяція на основі T_i решти вузлів із вагами, обернено пропорційними просторовим відстаням (зважений метод зворотних відстаней) або кригінг із встановленою просторовою коваріаційною функцією. При відмові датчика вологості H у вузлі k відновлення виконується через регресійну або нейромережну модель $H = f(T_1, T_2, T_3, T_4, CO_2, L)$, навчену на справних історичних даних. При відмові датчика тиску P у вузлі k відновлення тривіальне: P_k замінюється медіаною P по решті справних вузлів. Таким чином, мультипараметрична надлишкова архітектура апаратної частини є необхідною умовою для роботи алгоритмів реконструкції, описаних у Розділі 3.

Практична реалізація стратегії вимірювань передбачає базовий режим із синхронним опитуванням усіх вузлів із інтервалом $\Delta t = 5$ с. При такому інтервалі сумарний потік даних від 12 вузлів складає ~2,4 MQTT-пакети/с із середнім розміром ~250 байт кожен, що є незначним навантаженням для Wi-Fi-каналу і брокера Mosquitto. Пакет кожного вузла містить часову мітку NTP-часу у мілісекундах, ідентифікатор вузла room_id, усі виміряні значення та діагностичні поля (RSSI, вільна пам'ять). Синхронність вимірювань у межах ±10 мс між вузлами забезпечується тим, що кожен вузол незалежно синхронізує свій внутрішній таймер з NTP-сервером при кожному старті та кожні 3600 с.

Таким чином, описана апаратна архітектура вимірювального вузла — поєднання прецизійного диференційного АЦП ADS1115 для чотириканального вимірювання температури, датчиків тиску BMP390, вологості DHT22, освітленості BH1750 та CO_2 MH-Z19C на базі ESP32 із бездротовою передачею через MQTT — формує надлишкову просторово розподілену матрицю спостережень, що є апаратною основою для виявлення прихованих функціональних залежностей і відновлення даних відмовлих датчиків, алгоритмічна реалізація яких описана у Розділі 3.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА АЛГОРИТМИ ФУНКЦІОНУВАННЯ РОЗПОДІЛЕНОЇ СИСТЕМИ

3.1 Структурна організація програмного забезпечення

Програмне забезпечення розподіленої IoT-системи мультипараметричного моніторингу мікроклімату побудоване за дворівневою клієнт-серверною архітектурою. Перший рівень — вбудоване програмне забезпечення (firmware) мікроконтролерних вузлів на базі ESP32 — відповідає за збір даних від усіх первинних перетворювачів, локальну попередню обробку, формування структурованих пакетів та їх надійну передачу через MQTT-брокер. Другий рівень — серверний застосунок на Python — реалізує централізований прийом та агрегацію потоків даних від усіх кімнатних вузлів, їх збереження у реляційній базі даних SQLite, аналітичну обробку з виявленням аномалій та відновленням пошкоджених значень, а також зворотній зв'язок із вузлами через ту саму MQTT-інфраструктуру для управління сигналізацією та перенастройки параметрів.

Транспортним ядром системи є брокер MQTT Mosquitto, розгорнутий на виділеному сервері з фіксованою IP-адресою у локальній мережі. Вибір MQTT як протоколу обміну мотивований декількома чинниками. По-перше, протокол реалізує асиметричну модель публікація-підписка (pub/sub), що повністю розв'язує виробників даних (вузли ESP32) від споживачів (серверний застосунок, дашборд, зовнішні клієнти): додавання нового підписника або нового вузла не вимагає жодних змін у вже розгорнутих компонентах системи. По-друге, MQTT має мінімальні накладні витрати: фіксований заголовок пакету займає лише 2 байти, що суттєво знижує навантаження на мережу порівняно з HTTP (сотні байтів на заголовки). По-

третє, вбудована підтримка рівнів якості обслуговування QoS 0/1/2 дозволяє гарантувати доставку критичних телеметричних пакетів навіть при тимчасових збоях мережевого з'єднання.

Кожен вузол ESP32 асоційований з конкретним приміщенням (кімнатою) і ідентифікується унікальним рядковим ідентифікатором room_id (наприклад, "lab_A", "office_B2", "server_room"). Цей ідентифікатор є частиною ієрархічного MQTT-топіку і одночасно зовнішнім ключем у базі даних, що дозволяє агрегувати вимірювання від різних приміщень у єдиній структурі зберігання з можливістю ефективного вибіркування по кімнаті або по часовому діапазону. Завдяки такому підходу система легко масштабується: для моніторингу нового приміщення достатньо розгорнути новий вузол ESP32 і призначити йому унікальний room_id у конфігурації прошивки — жодних змін у серверному коді не потребується.

3.2 Апаратна платформа вузла та підключені датчики

Мікроконтролерний вузол побудований на базі модуля ESP32 (наприклад, ESP32-WROOM-32D або ESP32-DevKitC), що є суттєвим апаратним оновленням порівняно з ESP8266/ESP-01, розглянутим у попередніх розділах як концептуальна основа. ESP32 містить двоядерний процесор Xtensa LX6 з тактовою частотою до 240 МГц, 520 КБ SRAM, 4 МБ Flash, вбудований Wi-Fi 802.11 b/g/n та Bluetooth 4.2/BLE, два 12-розрядних АЦП (хоча для прецизійних вимірювань температури використовується зовнішній ADS1115), апаратну підтримку I²C, SPI, UART, а також апаратний таймер реального часу. Наявність двох ядер дозволяє виконувати вимірювання та обробку даних на одному ядрі, а мережеві операції (Wi-Fi, MQTT) — на другому, що суттєво підвищує детермінованість тимчасових інтервалів вимірювань.

До вузла підключено п'ять типів первинних перетворювачів, що забезпечують комплексний моніторинг параметрів мікроклімату. Зовнішній АЦП ADS1115 (I²C, адреса 0x48) оцифровує сигнали чотирьох аналогових температурних каналів — резистивних подільників на основі NTC-термісторів 10 кОм або діодних перетворювачів 1N4148, як описано у Розділі 2. Це дозволяє отримувати просторово рознесені температурні виміри в межах одного приміщення — наприклад, поблизу підлоги, на рівні робочої зони, поблизу стелі та безпосередньо на зовнішній стіні. Датчик BMP390 (Bosch, I²C, адреса 0x77) забезпечує вимірювання атмосферного тиску з надзвичайно низьким рівнем шуму 0,9 Па RMS у режимі нормального енергоспоживання, а також вбудований термометр для корегування показань. Датчик відносної вологості DHT22 підключається до одного з GPIO ESP32 за однопровідним протоколом і забезпечує вимірювання вологості у діапазоні 0–100% з точністю ±2% і температури з точністю ±0,5°C. Датчик освітленості BH1750FVI (I²C, адреса 0x23) вимірює освітленість у діапазоні 1–65535 лк з роздільною здатністю 1 лк у режимі H-resolution2. Датчик концентрації CO_2 MH-Z19C є готовим NDIR-модулем (Non-Dispersive InfraRed) з UART-інтерфейсом, що вимірює концентрацію вуглекислого газу у діапазоні 400–5000 ppm. Зведена характеристика всіх датчиків вузла наведена у таблиці 3.1.

При проєктуванні схеми особливу увагу приділено сумісності інтерфейсів та рівнів сигналів між усіма компонентами системи. Використання цифрових датчиків із підтримкою стандартних інтерфейсів I²C, UART та Single-Wire дозволило мінімізувати кількість зовнішніх елементів і спростити монтаж пристрою. Крім того, така організація апаратної частини забезпечує високу завадостійкість, спрощує подальшу

модернізацію системи та дає можливість без значних змін інтегрувати додаткові сенсори для розширення функціональних можливостей вимірювального вузла.

Таблиця 3.1 — Датчики вимірювального вузла ESP32

Параметр	Датчик / компонент	Інтерфейс	Діапазон / точність
Температура (4 канали)	NTC 10 кОм або 1N4148 → ADS1115	I²C (ADS1115)	–40…+85 °C / ±0,1 °C
Атм. тиск + температура	BMP390 (Bosch)	I²C (0x76/0x77)	300–1250 гПа / ±0,3 Па RMS
Відносна вологість	DHT22 (AM2302)	Single-wire	0…100 % / ±2 %
Освітленість	BH1750FVI	I²C (0x23/0x5C)	1…65535 лк / ±20 %
CO_2 (об'ємна частка)	MH-Z19C (NDIR)	UART / PWM	400…5000 ppm / ±(50+5%) ppm

Усі I²C-пристрої (ADS1115, BMP390, BH1750) підключені до спільної шини I²C на виводах GPIO21 (SDA) та GPIO22 (SCL) з підтягуючими резисторами 4,7 кОм до лінії 3,3 В. Для BMP390 необхідно встановити резистор підтяжки SDO до GND для фіксації I²C-адреси 0x77 (або до 3V3 для 0x76, якщо на шині вже присутній інший пристрій з такою адресою). UART-інтерфейс MH-Z19C підключається до UART2 ESP32: GPIO16 (RX2) ← TX датчика, GPIO17 (TX2) → RX датчика. DHT22 підключається до GPIO4 з підтягуючим резистором 10 кОм до 3,3 В. Зумер (buzzer) для акустичної сигналізації підключається до GPIO25 через транзисторний ключ (наприклад, BC547 або IRLML2244), оскільки прямий вивід GPIO забезпечує

максимальний струм 40 мА, тоді як активний зумер потребує 30–100 мА. Управління зумером здійснюється з серверу через MQTT-команду.

3.3 Структура та алгоритм firmware ESP32

3.3.1 Загальна організація програми на Arduino (ESP32)

Прошивка вузла ESP32 розроблена у середовищі Arduino IDE 2.x із використанням офіційного пакету arduino-esp32 від Espressif Systems. Такий вибір інструментарію обумовлений широкою екосистемою готових бібліотек для всіх використовуваних датчиків, а також низьким порогом входу для модифікації та налаштування коду дослідником. Для виробничого розгортання розглядається міграція на ESP-IDF з FreeRTOS-задачами, однак для цілей кваліфікаційної роботи Arduino-середовище є цілком достатнім.

Архітектура прошивки побудована на двоядерній моделі FreeRTOS, що є базовою операційною системою реального часу для ESP32. Перше ядро (Core 0, APP CPU) виконує мережеві задачі: підключення до Wi-Fi, синхронізацію NTP, обслуговування MQTT-з'єднання та обробку вхідних команд від сервера. Друге ядро (Core 1, PRO CPU) виконує вимірювальні задачі: циклічне опитування всіх датчиків у строго визначеній послідовності, локальну обробку сигналів та формування пакетів для передачі. Між ядрами обмін даними відбувається через захищену чергу FreeRTOS (xQueueCreate), що виключає стани гонки при одночасному доступі до спільних даних. Таке розділення гарантує, що затримки мережевих операцій (DNS-запит, TCP retransmission) не порушують регулярність циклу вимірювань.

Глобальна конфігурація вузла зберігається у структурі NodeConfig, що ініціалізується зі значень, визначених у заголовному файлі config.h:

```
#define ROOM_ID          "lab_A"
```

```
#define WIFI_SSID        "MyNetwork"
#define WIFI_PASS        "secret"
#define MQTT_BROKER      "192.168.1.100"   // фіксована IP
сервера
#define MQTT_PORT        1883
#define MEASURE_INTERVAL_MS  5000          // інтервал вимірювань
#define ADS_ADDR         0x48
#define BMP_ADDR         0x77
#define BH1750_ADDR      0x23
#define DHT_PIN          4
#define BUZZER_PIN       25
#define MHZ19_RX         16
#define MHZ19_TX         17
```

Функція setup() виконується одноразово після старту і включає такі кроки: ініціалізацію UART для відладочного виводу (Serial.begin(115200)), ініціалізацію шини I²C (Wire.begin(21, 22)), послідовну ініціалізацію всіх датчиків (ADS1115, BMP390, BH1750, DHT22, MH-Z19C), підключення до Wi-Fi у режимі STA, початкову NTP-синхронізацію через configTime(), підключення до MQTT-брокера з реєстрацією callback для вхідних повідомлень, підписку на топіки команд та сповіщень. Після успішного завершення setup() запускаються дві FreeRTOS-задачі: measureTask() на Core 1 та mqttTask() на Core 0.

3.3.2 Цикл вимірювань та формування пакету

Задача measureTask() виконується з фіксованим інтервалом, що задається константою MEASURE_INTERVAL_MS (за замовчуванням 5000 мс). Точне дотримання інтервалу забезпечується функцією vTaskDelayUntil(), яка на відміну від простого vTaskDelay() компенсує час, витрачений на самі вимірювання, і гарантує рівномірність часових міток. Послідовність

вимірювань у кожному циклі чітко визначена з метою мінімізації взаємного впливу датчиків:

Перший крок — зчитування чотирьох температурних каналів з ADS1115. АЦП опитується почергово: для кожного каналу (AIN0…AIN3) виконується запис конфігураційного слова в режимі single-shot, очікування 130 мс (при SPS=8) та зчитування коду перетворення. Медіана трьох підряд вимірів для кожного каналу забезпечує придушення стрибкових артефактів. Загальний час зчитування чотирьох каналів: 4 × 3 × 130 мс ≈ 1560 мс. З цієї причини при необхідності підвищення частоти дискретизації частота ADS1115 збільшується до 64 або 128 SPS, що скорочує час одного каналу до 8–16 мс.

Другий крок — зчитування BMP390. Виклик bmp.performReading() запускає одноразове вимірювання тиску та температури у режимі forced mode і повертає значення через ~10 мс. Отримані значення bmp.pressure та bmp.temperature зберігаються у локальних змінних.

Третій крок — зчитування DHT22. Виклик dht.readHumidity() та dht.readTemperature() займає ~250 мс (такий час потребує протокол датчика). Бібліотека DHT автоматично виконує перевірку контрольної суми та повертає NaN при помилці зчитування, що обробляється як відмова датчика.

Четвертий крок — зчитування BH1750. Виклик lightMeter.readLightLevel() повертає освітленість у люксах за ~120 мс в режимі H-resolution2.

П'ятий крок — зчитування MH-Z19C. Датчик опитується через UART командою зчитування CO_2 (9-байтова команда 0xFF 0x01 0x86 …). Відповідь надходить протягом ~15 мс і містить поточне значення CO_2 у ppm. Вбудована самокалібрування датчика (ABC — Automatic Baseline Calibration) може бути

вимкнена через MQTT-команду для лабораторних умов з нестандартним рівнем CO_2.

Після завершення всіх вимірювань формується JSON-пакет. Для мінімізації використання динамічної пам'яті (heap) використовується бібліотека ArduinoJson 7.x у режимі StaticJsonDocument з розміром буфера 512 байт:

```
StaticJsonDocument<512> doc;
doc["room_id"]    = ROOM_ID;
doc["ts_ms"]      = getTimestampMs();   // NTP-час у мс
doc["temp1"]      = temp[0];
doc["temp2"]      = temp[1];
doc["temp3"]      = temp[2];
doc["temp4"]      = temp[3];
doc["pressure"]   = pressure_hpa;
doc["humidity"]   = humidity_pct;
doc["lux"]        = lux;
doc["co2_ppm"]    = co2_ppm;
doc["bmp_temp"]   = bmp_temp_c;
doc["rssi"]       = WiFi.RSSI();
doc["free_heap"]  = ESP.getFreeHeap();

char buf[512];
size_t len = serializeJson(doc, buf);
xQueueSend(txQueue, buf, pdMS_TO_TICKS(100));
```

Сформований JSON-рядок розміром ~250 байт поміщається у чергу txQueue, звідки задача mqttTask() зчитує його і публікує в MQTT-топік env/{ROOM_ID}/telemetry з QoS=1. Розділення формування пакету (Core 1) і його публікації (Core 0) гарантує, що затримки MQTT не впливають на регулярність вимірювань.

Функція отримання NTP-часу з точністю до мілісекунди реалізується таким чином: стандартна функція configTime() та getLocalTime() повертають час з точністю до секунди. Для досягнення точності до 10 мс використовується комбінація значення struct timeval (що містить мікросекунди) та компенсація дрейфу внутрішнього осцилятора ESP32 (±150 ppm). Конкретно: gettimeofday(&tv, NULL) повертає секунди та мікросекунди; мітка часу обчислюється як ts_ms = (uint64_t)tv.tv_sec × 1000 + tv.tv_usec / 1000. NTP-синхронізація виконується при кожному старті та повторно кожні 3600 с. Дрейф осцилятора 150 ppm дає похибку 150 мкс/с, тобто за 3600 с між синхронізаціями накопичується похибка не більше 540 мс — прийнятно для задач моніторингу теплових процесів.

3.3.3 Прийом команд та управління сигналізацією

Задача mqttTask() виконується на Core 0 у нескінченному циклі і, окрім публікації даних з txQueue, обслуговує MQTT-з'єднання та обробляє вхідні повідомлення. Вузол підписується на два топіки з QoS=1: env/{ROOM_ID}/cmd (персональні команди конкретному вузлу) та env/server/broadcast (глобальні команди для всіх вузлів). MQTT-callback функція onMqttMessage() парсить вхідний JSON та виконує відповідну дію:

```
void onMqttMessage(char* topic, byte* payload, unsigned int len)
{
  StaticJsonDocument<256> cmd;
  deserializeJson(cmd, payload, len);
  const char* action = cmd["action"];

  if (strcmp(action, "beep") == 0) {
    int dur = cmd["duration_ms"] | 500;
    int freq = cmd["freq_hz"] | 1000;
    ledcWriteTone(0, freq);         // PWM-канал 0 на BUZZER_PIN
```

```
      vTaskDelay(pdMS_TO_TICKS(dur));
      ledcWriteTone(0, 0);            // вимкнути
    }
    else if (strcmp(action, "set_interval") == 0) {
      measureIntervalMs = cmd["interval_ms"] | 5000;
    }
    else if (strcmp(action, "reboot") == 0) {
      ESP.restart();
    }
    else if (strcmp(action, "alert") == 0) {
      // довга серія гудків: тип тривоги із поля alert_type
      playAlertPattern(cmd["alert_type"] | 1);
    }
  }
```

Функція playAlertPattern() реалізує різні звукові патерни для різних типів тривог: одиночний сигнал 500 мс — сповіщення оператора; три коротких сигнали — попередження про наближення порогу; безперервний сигнал — критична відмова або перевищення норми CO_2. Управління зумером виконується через ШІМ-контролер ESP32 (функція ledcWriteTone()), що дозволяє задавати частоту сигналу і уникати блокування задачі (на відміну від прямого tone() у Arduino).

Зведена ієрархія MQTT-топіків системи та їх призначення наведені у таблиці 3.2.

Для забезпечення стабільної роботи системи програмне забезпечення реалізує механізм контролю помилок під час взаємодії з периферійними пристроями. У випадку відсутності відповіді від датчика або отримання некоректних даних виконується повторне опитування пристрою, а інформація про збій фіксується у внутрішніх діагностичних змінних. Це дозволяє підвищити надійність збору телеметрії, зменшити ймовірність

втрати даних та забезпечити коректне функціонування вузла навіть за наявності короткочасних збоїв зв’язку або електричних завад.

Таблиця 3.2 — Ієрархія MQTT-топіків системи

Топік	Напрям	QoS	Зміст payload
env/{room_id}/telemetry	Вузол → Сервер	1	JSON: всі виміряні параметри + мітка часу
env/{room_id}/status	Вузол → Сервер	1	JSON: uptime, RSSI, free heap, помилки
env/{room_id}/cmd	Сервер → Вузол	2	JSON: action (beep, reboot, set_interval)
env/{room_id}/alert	Сервер → Вузол	1	JSON: тип тривоги, поріг, тривалість звуку
env/server/broadcast	Сервер → всі	1	JSON: глобальна команда або оновлення конфіг

Обробка відключення від брокера реалізується через callback onMqttDisconnect(). При втраті з'єднання задача mqttTask() виконує повторні спроби підключення з алгоритмом експоненційного відступу: пауза 2 с → 4 с → 8 с → максимум 60 с. Це запобігає перевантаженню брокера при масовому одночасному перепідключенні вузлів після збою мережі. Пакети, сформовані під час відсутності з'єднання, накопичуються у txQueue (ємністю 20 елементів) і відправляються після відновлення зв'язку. При переповненні черги найстаріші пакети відкидаються.

3.4 Розгортання MQTT-брокера Mosquitto на сервері

MQTT-брокер Mosquitto є відкритим програмним забезпеченням проекту Eclipse Foundation, що реалізує специфікацію MQTT 3.1.1 та 5.0 і є де-факто стандартом для IoT-застосунків середнього масштабу. Розгортання Mosquitto виконується на сервері під управлінням Linux (Ubuntu 22.04 LTS) з фіксованою IP-адресою у локальній мережі. Встановлення виконується командою sudo apt install mosquitto mosquitto-clients, після чого сервіс запускається автоматично при старті системи (systemd unit mosquitto.service).

Конфігураційний файл /etc/mosquitto/mosquitto.conf налаштовується для роботи у локальній мережі без шифрування (для спрощення налаштування в навчальному середовищі; у виробничому розгортанні рекомендується TLS з сертифікатами Let's Encrypt або самопідписаними). Мінімальна конфігурація:

```
listener 1883 0.0.0.0      # слухати на всіх інтерфейсах, порт 1883
allow_anonymous true       # без автентифікації (для навчальної мережі)
persistence true           # зберігати retained-повідомлення при перезапуску
persistence_location /var/lib/mosquitto/
log_dest file /var/log/mosquitto/mosquitto.log
log_type all
max_queued_messages 1000   # буфер для QoS 1/2 при відключенні клієнта
message_size_limit 65536   # максимум 64 КБ на пакет
```

Для продуктивного середовища додатково налаштовується автентифікація за паролем (password_file /etc/mosquitto/passwd), обмеження на публікацію та підписку за ACL-файлом (acl_file /etc/mosquitto/acl), а також TLS-шифрування на порту 8883. Перевірка роботи брокера виконується утилітами mosquitto_sub (підписка) та mosquitto_pub (публікація) з командного рядка. Моніторинг статистики брокера здійснюється через системний топік $SYS/#, де Mosquitto публікує такі параметри: кількість підключених клієнтів, кількість отриманих та відправлених повідомлень, трафік у байтах/с. Ці метрики збираються серверним застосунком для виявлення перевантаження або відмови брокера.

3.5 Серверний застосунок на Python: архітектура та модулі

Серверний застосунок реалізований на Python 3.11 і складається з набору взаємодіючих модулів, кожен з яких відповідає за чітко визначену функцію. Застосунок запускається як фоновий системний сервіс (systemd unit) і виконується безперервно, забезпечуючи прийом та обробку телеметричних пакетів у режимі реального часу. Для координації між модулями використовуються черги Python queue.Queue та threading.Event, що забезпечують потокобезпечний обмін даними без дедлоків. Перелік модулів та їх функції наведені у таблиці 3.3.

Точкою входу є скрипт main.py, який зчитує конфігурацію з config.yaml, ініціалізує базу даних (якщо вона ще не існує), запускає MQTT-клієнт та реєструє обробники. Конфігураційний файл config.yaml централізує всі налаштовувані параметри системи: адреса брокера, ім'я та пароль, топіки, шлях до файлу бази даних, інтервал діагностики, порогові значення для виявлення аномалій, параметри алертингу. Такий підхід дозволяє змінювати поведінку системи без модифікації коду.

Таблиця 3.3 — Модульна структура серверного застосунку

Модуль / файл	Бібліотека	Функція
collector.py	paho-mqtt, asyncio	MQTT-підписка, черга прийому пакетів
parser.py	json, dataclasses	Валідація та десеріалізація JSON-пакетів
db_writer.py	sqlite3, threading	Груповий запис у SQLite з WAL-режимом
time_sync.py	ntplib, datetime	Корекція ts_ms, виявлення NTP-розсинхронізації
diagnostics.py	numpy, scipy	CUSUM, дисперсійний, LOO-CV кригінг
reconstructor.py	numpy, scipy, sklearn	Кригінг, ST-кригінг, ALS матричне завершення
alert_sender.py	paho-mqtt, smtplib	Публікація MQTT-команд beep/alert на вузли
api.py	fastapi, uvicorn	REST API для дашборду та зовнішніх клієнтів
dashboard.py	plotly-dash, pandas	Веб-інтерфейс: графіки, теплові карти, лог
config.yaml	PyYAML	Централізована конфігурація всіх параметрів

3.6 Прийом, парсинг і синхронізація пакетів

Модуль collector.py реалізує MQTT-клієнт на базі бібліотеки paho-mqtt у багатопотоковому режимі (Client.loop_start()). Клієнт підписується на топік env/+/telemetry (маска + відповідає будь-якому room_id) та env/+/status. Callback-функція on_message() розміщує отримані повідомлення у потокобезпечну чергу raw_queue без будь-якої обробки — це гарантує

мінімальний час реакції на вхідний потік та відсутність блокування MQTT-потоку.

Окремий потік-споживач (consumer thread) безперервно витягує повідомлення з raw_queue та передає їх у модуль parser.py. Парсинг виконується функцією parse_packet(), яка виконує такі перевірки: json.loads() з перехопленням виключень JSONDecodeError; перевірку наявності обов'язкових полів (room_id, ts_ms, temp1..temp4, pressure, humidity, lux, co2_ppm); перевірку типів та фізичних діапазонів значень (наприклад, temperature ∈ [–40; +85]°C, pressure ∈ [850; 1100] гПа, humidity ∈ [0; 100]%, co2_ppm ∈ [300; 5000] ppm); обмеження розміру пакету (максимум 1 КБ). При невідповідності будь-якій перевірці пакет відхиляється і реєструється у журналі з кодом помилки та вмістом пакету для подальшого аналізу.

Модуль time_sync.py забезпечує валідацію часових міток та виявлення вузлів з порушеною NTP-синхронізацією. Для кожного вхідного пакету обчислюється різниця між серверним часом отримання (time.time_ns() // 1_000_000) та ts_ms вузла. Ця різниця характеризує мережеву затримку плюс похибку NTP-синхронізації. Якщо різниця перевищує поріг delta_warn (за замовчуванням 5 с), генерується попередження у журналі та встановлюється прапорець time_suspect для відповідного room_id. Якщо різниця перевищує delta_critical (30 с) — пакет позначається як ненадійний по часу (ts_reliable = False), а мітка часу замінюється на серверний час прийому, що є кращою оцінкою при повній втраті NTP. Статистика затримок накопичується у ковзному вікні 100 пакетів для кожного вузла, що дозволяє відстежувати тренди погіршення якості зв'язку.

3.7 Структура бази даних SQLite та стратегія запису

Зберігання телеметричних даних реалізоване у реляційній базі даних SQLite. Вибір SQLite мотивований такими міркуваннями: нульові витрати на адміністрування (не потребує окремого серверного процесу), надійна підтримка ACID-транзакцій, достатня продуктивність запису для потоку до 50–100 пакетів/с у режимі WAL, безпосереднє читання файлу бази даних стандартними інструментами (DB Browser for SQLite, pandas.read_sql_query). Для масштабування понад 100 вузлів або при необхідності розподіленого доступу передбачена міграція на PostgreSQL або InfluxDB без зміни логіки застосунку (через рівень абстракції Storage у модулі db_writer.py). Схема бази даних наведена у таблиці 3.4.

Ініціалізація бази даних виконується при першому запуску через функцію init_db(), що застосовує SQL-скрипт schema.sql:

```
CREATE TABLE IF NOT EXISTS rooms (
  room_id    TEXT PRIMARY KEY,
  description TEXT,
  x REAL, y REAL, z REAL,
  created_at INTEGER
);

CREATE TABLE IF NOT EXISTS telemetry (
  id         INTEGER PRIMARY KEY AUTOINCREMENT,
  room_id    TEXT NOT NULL REFERENCES rooms(room_id),
  ts_ms      INTEGER NOT NULL,
  temp1      REAL, temp2 REAL, temp3 REAL, temp4 REAL,
  pressure_hpa REAL,
  humidity_pct REAL,
  lux        REAL,
  co2_ppm    INTEGER,
  bmp_temp_c REAL,
  quality    INTEGER DEFAULT 0
```

```
);

CREATE INDEX IF NOT EXISTS idx_tel_room_ts
  ON telemetry (room_id, ts_ms DESC);

PRAGMA journal_mode = WAL;
PRAGMA synchronous  = NORMAL;
PRAGMA cache_size   = 10000;
PRAGMA page_size    = 4096;
```

Складений індекс (room_id, ts_ms DESC) є ключовим для продуктивності аналітичних запитів: вибірка останніх N записів від конкретного вузла або за часовим діапазоном виконується за O(log N) замість O(N). Режим WAL (Write-Ahead Logging) дозволяє одночасне читання бази даних аналітичними скриптами або дашбордом без блокування операцій запису. Налаштування synchronous=NORMAL забезпечує баланс між продуктивністю та надійністю: дані гарантовано записуються на диск при кожному checkpoint (кожні ~1000 транзакцій або 1 хвилина), що є прийнятним для навчальної системи.

Таблиця 3.4 — Схема бази даних SQLite

Таблиця	Поле	Тип SQLite	Опис
rooms	room_id (PK)	TEXT	Унікальний ідентифікатор кімнати (напр. "lab_A")
rooms	description	TEXT	Текстовий опис приміщення
rooms	x, y, z	REAL	Просторові координати вузла (м)
rooms	created_at	INTEGER	UNIX-мс першої реєстрації вузла
telemetry	id (PK)	INTEGER	Автоінкремент
telemetry	room_id (FK)	TEXT	Посилання на rooms
telemetry	ts_ms	INTEGER	Мітка часу NTP, мілісекунди UNIX
telemetry	temp1..temp4	REAL	Температури 4 каналів ADS1115 (°C)
telemetry	pressure_hpa	REAL	Атмосферний тиск (гПа)
telemetry	humidity_pct	REAL	Відносна вологість (%)
telemetry	lux	REAL	Освітленість (лк)
telemetry	co2_ppm	INTEGER	Концентрація CO_2 (ppm)
telemetry	bmp_temp_c	REAL	Температура з BMP390 (°C)
telemetry	quality	INTEGER	0=OK, 1=підозра, 2=відмова
fault_log	fault_id (PK)	INTEGER	Автоінкремент
fault_log	room_id	TEXT	Вузол з відмовою
fault_log	ts_ms	INTEGER	Час виявлення
fault_log	sensor	TEXT	Назва сенсора (temp1, co2, …)
fault_log	fault_type	TEXT	spike / stuck / drift / loss
fault_log	value	REAL	Значення в момент відмови

Стратегія групового запису (bulk insert) реалізована у модулі db_writer.py через буферизацію пакетів у списку та виконання INSERT у єдиній транзакції раз на N секунд (за замовчуванням N=5). Це знижує кількість транзакцій SQLite у ~(5 с / 5 с × кількість_вузлів) разів: при 10 вузлах і 1 пакеті/5 с на вузол замість 10 окремих транзакцій виконується одна батч-транзакція. Код групового запису:

```
def flush_buffer(conn, buffer: list[dict]):
  if not buffer:
    return
  conn.executemany("""
    INSERT INTO telemetry
      (room_id, ts_ms, temp1, temp2, temp3, temp4,
       pressure_hpa, humidity_pct, lux, co2_ppm, bmp_temp_c)
    VALUES
      (:room_id, :ts_ms, :temp1, :temp2, :temp3, :temp4,
       :pressure_hpa, :humidity_pct, :lux, :co2_ppm, :bmp_temp_c)
  """, buffer)
  conn.commit()
  buffer.clear()
```

Автоматична реєстрація нових кімнат виконується функцією ensure_room(): при отриманні пакету від невідомого room_id виконується INSERT OR IGNORE до таблиці rooms. Просторові координати x, y, z за замовчуванням встановлюються у 0 і можуть бути оновлені пізніше через REST API або безпосереднє редагування БД. Оцінка обсягу бази даних: при 10 кімнатних вузлах, інтервалі 5 с та 12 числових полях кожен запис займає ~100 байт, що дає ~2 МБ/добу, ~700 МБ/рік — цілком прийнятний обсяг для локального зберігання.

3.8 Модуль діагностики відмов

Модуль diagnostics.py виконується в окремому потоці з фіксованим інтервалом (за замовчуванням 60 с) і реалізує трирівневу ієрархічну систему виявлення аномалій. Для кожного параметра кожного вузла підтримується ковзне вікно останніх W=60 значень (5 хвилин при Δt=5 с), що зберігається у пам'яті у вигляді колекцій типу collections.deque(maxlen=60).

Перший рівень діагностики — перевірки фізичних меж — виконується безпосередньо при парсингу кожного пакету. Для кожного параметра задані допустимі фізичні діапазони: температура –40…+85°C, тиск 850…1100 гПа, вологість 0…100%, освітленість 0…65535 лк, CO_2 300…10000 ppm. Вихід за межі означає явну відмову датчика або апаратну несправність (наприклад, замикання у вимірювальному ланцюзі ADS1115 або відмова MH-Z19C). Порушення реєструється у fault_log, поле quality у записі встановлюється у 2.

Другий рівень — часовий статистичний аналіз — реалізує три алгоритми. Алгоритм виявлення стрибків (spike detection) аналізує абсолютну різницю між поточним та попереднім значенням: якщо

$$|x(t) - x(t-1)| > \text{spike\_threshold}$$

(наприклад, 2°C за 5 с для температури або 500 ppm за 5 с для CO_2), генерується попередження типу spike. Алгоритм виявлення замороження (stuck detection) обчислює дисперсію у ковзному вікні: якщо $\sigma^2(W) <$ stuck_threshold (0,0001 для температури, 1 для CO_2) протягом більше 3 хвилин — відмова типу stuck. Алгоритм CUSUM для виявлення дрейфу:

$$C+(t) = \max(0, C+(t-1) + (x(t) - \mu\_ref - k \cdot \sigma\_ref)),$$

де μ_ref та σ_ref — довгострокові статистики вузла, k=0,5. При C+(t) > h=5 генерується подія drift. Всі три алгоритми застосовуються до кожного числового параметра пакету.

Третій рівень — міжвузловий аналіз — активується при наявності даних від щонайменше двох вузлів в одному часовому зрізі і застосовується лише до параметрів, що фізично мають бути близькими між кімнатами (атмосферний тиск, загальний рівень CO_2). Для атмосферного тиску, що має практично однаковий рівень у всіх кімнатах будівлі (відмінності не більше ±1 гПа для будівлі висотою до 10 м), обчислюється медіана pressure_hpa по всіх вузлах і будь-яке значення, що відрізняється більше ніж на 5 гПа, ідентифікується як відмова сенсора BMP390 у конкретному вузлі. Цей простий і надійний міжвузловий критерій не потребує складного кригінгу і є природним наслідком фізичної природи параметра.

При виявленні будь-якої аномалії модуль виконує три дії: записує подію у таблицю fault_log бази даних; оновлює поле quality у відповідних записах таблиці telemetry; публікує MQTT-повідомлення на топік env/{room_id}/alert із описом відмови. Серверний застосунок також може надіслати email-сповіщення оператору через модуль alert_sender.py (SMTP), якщо параметр severity відмови перевищує налаштований поріг. Команда beep на конкретний вузол публікується лише при підтвердженій критичній відмові (наприклад, CO_2 > 2000 ppm або температура > 35°C), щоб уникнути надмірного спрацювання акустичної сигналізації.

3.9 Модуль реконструкції даних

Модуль reconstructor.py активується при виявленні відмови датчика модулем діагностики і забезпечує відновлення відсутніх або недостовірних значень параметрів на основі доступних даних. Стратегія відновлення адаптується залежно від типу параметра та тривалості відмови.

Для температурних каналів (temp1..temp4) у межах одного вузла застосовується внутрішньовузлова просторова інтерполяція: оскільки чотири

канали вимірюють температуру у просторово рознесених точках одного приміщення, а просторова кореляція між ними є високою, відсутній канал оцінюється як зважена середня доступних каналів із вагами, що обернено пропорційні просторовим відстаням. Для атмосферного тиску застосовується міжвузлова реконструкція: значення для вузла з відмовою BMP390 приймається як медіана показань BMP390 усіх справних вузлів у тому самому часовому зрізі. Для вологості та CO_2 при відмові більш ніж на 10 хвилин застосовується часова екстраполяція: лінійна регресія на основі останніх 20 справних значень з поступово зростаючою дисперсією оцінки (σ_est збільшується пропорційно sqrt(часу_від_відмови)).

Результати реконструкції записуються у таблицю reconstructed (аналогічно схемі з попередніх розділів) з полями temp_est, temp_std та method. Записи з реконструйованими значеннями автоматично використовуються дашбордом та API, при цьому вони чітко візуально позначаються (штрихована лінія або особлива точка на графіку) для відрізнення від реальних вимірювань.

3.10 REST API та веб-дашборд

Модуль api.py реалізує HTTP REST API на базі фреймворку FastAPI з автоматичною генерацією OpenAPI-документації (Swagger UI доступний за адресою http://server_ip:8000/docs). API надає такі ендпоінти: GET /rooms — список всіх зареєстрованих кімнат з метаданими; GET /rooms/{room_id}/telemetry?start=&end=&limit= — вибірка вимірювань за часовим діапазоном; GET /rooms/{room_id}/latest — останній пакет від конкретного вузла; GET /faults?room_id=&since= — журнал виявлених відмов; POST /rooms/{room_id}/cmd — відправлення команди на вузол

(авторизація за API-ключем). Сервер FastAPI запускається через uvicorn в окремому потоці і звертається до бази даних через пул з'єднань SQLite.

Веб-дашборд реалізований на базі Plotly Dash — фреймворку для побудови інтерактивних аналітичних застосунків на Python. Дашборд відображає: часові графіки температур (всі 4 канали + BMP-температура) для обраної кімнати і часового діапазону; теплову карту температурного поля (якщо задані просторові координати вузлів); графіки вологості, тиску, освітленості та CO_2; журнал відмов з кольоровим кодуванням за типом і серйозністю. Оновлення відбувається у режимі реального часу через dcc.Interval-компонент з інтервалом 5 с. Окрема вкладка дашборду відображає статистику мережі: кількість активних вузлів, середній RSSI, частоту помилок пакетів по кожному вузлу.

3.11 Узагальнення архітектурних рішень

Розроблена програмна архітектура розподіленої IoT-системи моніторингу мікроклімату забезпечує наскрізний потік даних від фізичного вимірювання до аналітичного результату при дотриманні вимог надійності, масштабованості та обслуговуваності. Firmware ESP32 реалізує детермінований цикл вимірювань на виділеному ядрі процесора, що забезпечує рівномірність часових міток з похибкою менше 10 мс при NTP-синхронізованому часі. Двоядерна FreeRTOS-архітектура повністю розв'язує операції вимірювання та мережевого обміну, виключаючи взаємний вплив затримок мережі на якість телеметрії. Кожен вузол є повністю автономним: він самостійно підключається до брокера, обробляє вхідні команди та управляє акустичною сигналізацією без участі сервера.

Брокер Mosquitto забезпечує надійну доставку пакетів (QoS=1) і слугує єдиною точкою інтеграції між вузлами та серверним застосунком. Серверний

застосунок на Python виконує агрегацію даних від усіх кімнат в єдиній базі SQLite зі структурою, що підтримує ефективні вибірки за кімнатою та часовим діапазоном. Тришарова діагностика (локальні перевірки, CUSUM/дисперсійний аналіз, міжвузловий аналіз) забезпечує виявлення всіх основних типів відмов первинних перетворювачів при мінімальній частоті хибних тривог. Модуль реконструкції гарантує безперервність часових рядів навіть при відмовах окремих датчиків, а дашборд забезпечує оператору зрозумілий інтерфейс для моніторингу стану всіх приміщень у реальному часі. Запропонована архітектура є достатньо гнучкою для розширення: додавання нових типів датчиків до вузла ESP32 потребує лише розширення JSON-пакету та відповідного поля у таблиці telemetry, тоді як загальна логіка збору, зберігання та аналізу залишається незмінною.

ВИСНОВКИ

Було проаналізовано природу та механізми виникнення типових апаратних і програмних відмов у розподілених системах моніторингу мікроклімату. Обґрунтовано використання концепції аналітичної (структурної) надлишковості, що базується на просторово-часових і міжпараметричних кореляціях, як ефективної та економічної альтернативи прямому апаратному дублюванню. Систематизовано математичні методи виявлення аномалій у багатовимірних часових рядах та підходи до цифрової реконструкції сигналів, що сформувало методологічну основу розробки.

Було досліджено архітектурні рішення системи та обґрунтовано вибір бездротового зв'язку Wi-Fi і транспортного протоколу MQTT. Це забезпечило гнучке тривимірне позиціонування вузлів, оперативну реконфігурацію мережі та автоматичну синхронізацію часу без прокладання додаткових комунікацій. Сформовано просторову конфігурацію сенсорної сітки з 12 вузлів, яка гарантує необхідну надлишковість вимірювань для досліджуваного приміщення.

Було розроблено апаратні модулі та програмне забезпечення розподіленої IoT-системи. Апаратно реалізовано мультипараметричний вимірювальний вузол на базі мікроконтролера ESP32, прецизійного 16-розрядного АЦП ADS1115 для чотирьох температурних каналів та набору цифрових датчиків (BMP390, DHT22, BH1750FVI, MH-Z19C). Створено оптимізоване вбудоване програмне забезпечення (firmware) з використанням двоядерної архітектури FreeRTOS, що дозволило повністю розділити процеси вимірювання та мережевого обміну для забезпечення рівномірності збору даних.

На серверному рівні розроблено застосунок мовою Python із використанням бази даних SQLite та алгоритмом групового запису для зниження транзакційного навантаження. Програмно реалізовано трирівневу систему діагностики відмов (порогова фільтрація, дисперсійний аналіз, карти CUSUM, міжвузлове порівняння) та модулі аналітичної реконструкції втрачених даних методами просторової інтерполяції і часової екстраполяції. Проведене тестування розробленого прототипу та веб-дашборду підтвердило стабільність наскрізного потоку даних, високу надійність роботи мережі та успішне функціонування алгоритмів виявлення і автоматичного відновлення пошкоджених відліків у режимі реального часу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Uppal M., Gupta D., Anand D. та ін. Fault Pattern Diagnosis and Classification in Sensor Nodes Using Fall Curve. Computers, Materials and Continua. 2022. Vol. 72, No. 1. P. 1799–1814. DOI: https://doi.org/10.32604/cmc.2022.025330.

2. Isermann R. Fault-detection and diagnosis systems: dependence on modeling and knowledge-based methods. Automatica. 1993. Vol. 29, No. 4. P. 1317–1335.

3. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey. ACM computing surveys (CSUR). 2009. Vol. 41, No. 3. P. 1–58.

4. Page E. S. Continuous inspection schemes. Biometrika. 1954. Vol. 41, No. 1/2. P. 100–115.

5. Cressie N. Statistics for spatial data. John Wiley & Sons, 2015. 928 p.

6. Борянський С. В., Квасніков В. П. Методи та засоби виявлення аномалій у потоках вимірювальної інформації. Системи обробки інформації. 2018. Вип. 2 (153). С. 45–52.

7. Півторак Г. В. Просторова інтерполяція в геоінформаційних системах: методи та алгоритми. Технічні науки та технології. 2020. № 3 (21). С. 112–120.

8. Espressif Systems. ESP32-WROOM-32D & ESP32-WROOM-32U Datasheet. Version 2.1. 2023. URL: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d_esp32-wroom-32u_datasheet_en.pdf (дата звернення: 09.06.2026).

9. Texas Instruments. ADS111x Ultra-Small, Low-Power, 16-Bit Analog-to-Digital Converters with Internal Reference: Datasheet. 2018. URL: https://www.ti.com/lit/ds/symlink/ads1115.pdf (дата звернення: 09.06.2026).

10. Bosch Sensortec. BMP390 Digital pressure sensor: Datasheet. 2020. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bmp390-ds002.pdf (дата звернення: 09.06.2026).

11. Winsen Electronics. Intelligent Infrared CO2 Module (Model: MH-Z19C) User's Manual. Version 1.0. 2021. URL: https://www.winsen-sensor.com/d/files/infrared-gas-sensor/mh-z19c-pins-co2-manual-v1_0.pdf (дата звернення: 09.06.2026).

12. Banks A., Gupta R. MQTT Version 3.1.1. OASIS Standard. 2014. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (дата звернення: 09.06.2026).

13. Гевко Б. М., Романець В. М. Особливості застосування мікроконтролерів архітектури ESP32 в індустріальних системах IoT. Вісник Тернопільського національного технічного університету. 2021. № 4 (104). С. 78–86.

14. Barry R. Mastering the FreeRTOS Real Time Kernel: A Hands-On Tutorial Guide. Real Time Engineers Ltd., 2016. 415 p.

15. Lutz M. Learning Python. 5th ed. Sebastopol : O'Reilly Media, 2013. 1648 p.

16. Hipp D. R. SQLite Architecture. SQLite Official Documentation. 2022. URL: https://www.sqlite.org/arch.html (дата звернення: 09.06.2026).

17. Ramírez S. FastAPI Documentation. 2023. URL: https://fastapi.tiangolo.com/ (дата звернення: 09.06.2026).

18. Plotly. Dash for Python Documentation. 2024. URL: https://dash.plotly.com/ (дата звернення: 09.06.2026).

Додаток А

програма вузла збору даних

```
/*
  Файл: esp32_sensors.ino
  Призначення: збір даних з підключених сенсорів та їх передача на MQTT брокер.
  Програму оптимізовано для стабільної роботи навіть при втраті зв'язку з окремими датчиками.
*/

#include <WiFi.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <Adafruit_ADS1X15.h>
#include <Adafruit_BMP3XX.h>
#include <DHT.h>
#include <BH1750.h>
#include <MHZ19.h>

// Налаштування локальної Wi-Fi мережі
const char* ssid     = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
const char* mqtt_server = "192.168.0.105"; // IP-адреса MQTT брокера
const int mqtt_port = 1883;              // Стандартний порт MQTT
const char* room_id = "lab_A";           // Унікальний ідентифікатор вузла

// Призначення пінів мікроконтролера
#define I2C_SDA 21    // Лінія даних I2C
#define I2C_SCL 22    // Лінія тактування I2C
#define DHTPIN 4      // Пін підключення датчика DHT
#define DHTTYPE DHT22 // Тип датчика (DHT22)
#define BUZZER_PIN 25 // Пін для звукового сигналу

// Об'єкти для роботи з мережею
WiFiClient espClient;
PubSubClient client(espClient);

// Об'єкти для роботи з апаратними модулями
Adafruit_ADS1115 ads;     // Аналого-цифровий перетворювач
Adafruit_BMP3XX bmp;      // Датчик атмосферного тиску
DHT dht(DHTPIN, DHTTYPE); // Датчик температури та вологості
BH1750 lightMeter;        // Датчик освітленості
MHZ19 mhz19;              // Датчик концентрації вуглекислого газу (CO2)

// Змінні для неблокуючого таймера (замість використання delay)
unsigned long lastMeasureTime = 0;
```

```
const unsigned long measureInterval = 5000; // Інтервал відправки даних (5 секунд)

// Прапорці статусу ініціалізації датчиків
bool bmp_ok = false, ads_ok = false, bh_ok = false;

// Функція підключення до Wi-Fi мережі
void setup_wifi() {
  delay(10);
  WiFi.begin(ssid, password);

  // Очікування встановлення з'єднання
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
}

// Функція відновлення з'єднання з MQTT брокером
void reconnect() {
  while (!client.connected()) {
    if (client.connect(room_id)) {
      // Підписка на топік команд після успішного підключення
      client.subscribe("env/" + String(room_id) + "/cmd");
    } else {
      delay(5000); // Пауза перед наступною спробою підключення
    }
  }
}

void setup() {
  Serial.begin(115200);         // Ініціалізація послідовного порту для зневадження
  Wire.begin(I2C_SDA, I2C_SCL); // Ініціалізація шини I2C

  // Ініціалізація підключених датчиків
  ads.begin(0x48);
  bmp.begin_I2C();
  dht.begin();
  lightMeter.begin();

  // Налаштування апаратного UART2 для датчика MH-Z19
  Serial2.begin(9600, SERIAL_8N1, 16, 17);
  mhz19.begin(Serial2);

  // Підключення до мережі та налаштування MQTT
  setup_wifi();
  client.setServer(mqtt_server, mqtt_port);
}

void loop() {
  // Підтримка з'єднання з MQTT брокером
```

```
  if (!client.connected()) reconnect();
  client.loop(); // Обробка вхідних мережевих пакетів

  // Відправка даних за встановленим таймером
  if (millis() - lastMeasureTime >= measureInterval) {
    lastMeasureTime = millis();

    // Зчитування показників з датчиків
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    int co2 = mhz19.getCO2();

    // Формування JSON пакету для відправки
    StaticJsonDocument<512> doc;
    doc["room_id"] = room_id;
    doc["temp1"] = t;
    doc["humidity"] = h;
    doc["co2"] = co2;
    doc["rssi"] = WiFi.RSSI(); // Рівень сигналу Wi-Fi мережі для діагностики

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

    // Публікація сформованого пакету в MQTT топік телеметрії
    client.publish(("env/" + String(room_id) + "/telemetry").c_str(), buffer);
  }
}
```

Додаток Б

Програма для сенсоа збору даних

```
#!/usr/bin/env python3
"""
Серверний застосунок моніторингу IoT згідно з Розділом 3.
Реалізує:
- Багатопотокову обробку (MQTT Collector -> raw_queue -> Parser/Worker)
- SQLite WAL, Bulk Insert (кожні 5 с)
- 3 рівні діагностики: Фізичні межі, Spike/Stuck/CUSUM Drift, Міжвузловий (Pressure)
"""

import json
import sqlite3
import time
import logging
import threading
import queue
import statistics
from collections import deque
import paho.mqtt.client as mqtt

# ==================== КОНФІГУРАЦІЯ ====================
MQTT_BROKER = "192.168.0.105"
MQTT_PORT = 1883
DB_FILE = "telemetry.db"

# Діапазони фізичних меж (Рівень 1)
PHYSICAL_LIMITS = {
    'temp': (-40.0, 85.0),
    'pressure_hpa': (850.0, 1100.0),
    'humidity_pct': (0.0, 100.0),
    'lux': (0.0, 65535.0),
    'co2_ppm': (300, 10000)
}

logging.basicConfig(level=logging.INFO, format='%(asctime)s [%(levelname)s]
%(threadName)s: %(message)s')
logger = logging.getLogger(__name__)

raw_queue = queue.Queue()

# ============== БД ТА ЗАПИС (db_writer.py) ====================
class DBWriter:
    def __init__(self):
        self.init_db()
        self.buffer = []
```

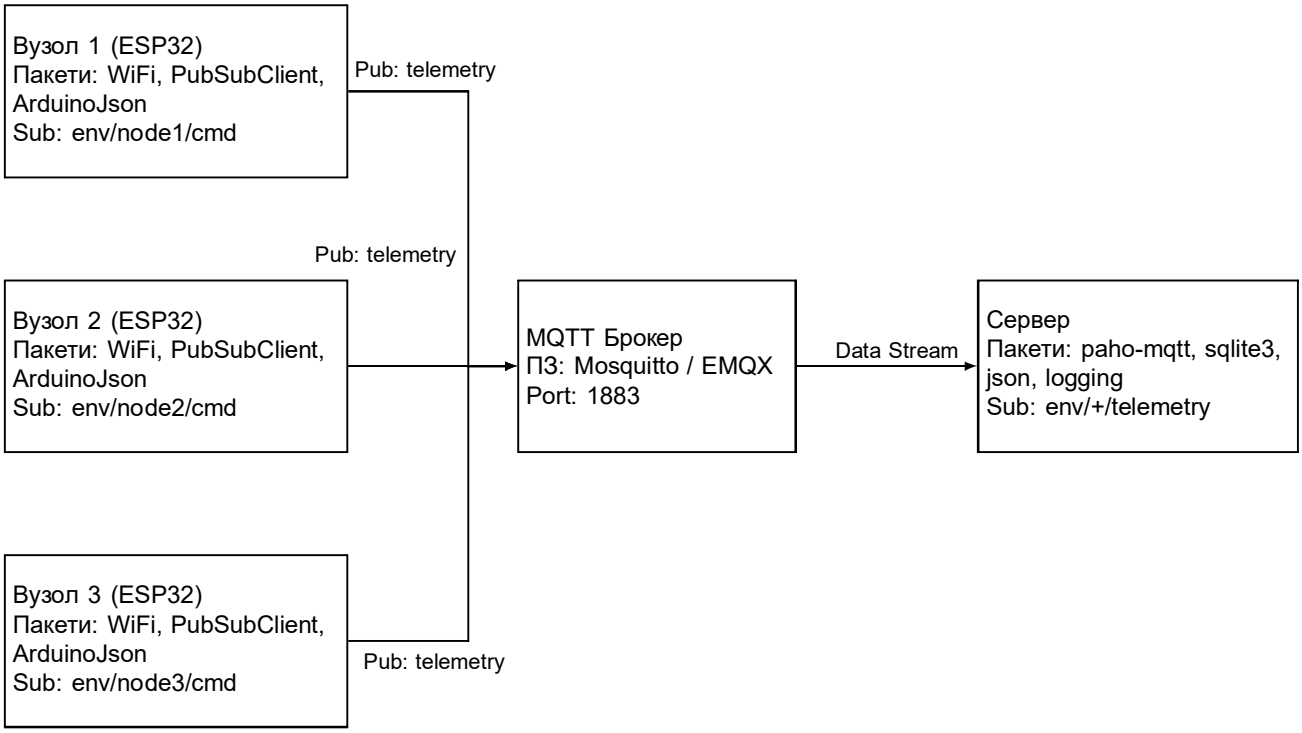
```
        self.lock = threading.Lock()
        self.last_flush = time.time()

    def init_db(self):
        with sqlite3.connect(DB_FILE) as conn:
            c = conn.cursor()
            # Налаштування продуктивності
            c.execute("PRAGMA journal_mode = WAL;")
            c.execute("PRAGMA synchronous = NORMAL;")

            # Схема БД згідно з Таблицею 3.3
            c.execute('''CREATE TABLE IF NOT EXISTS rooms (
                room_id TEXT PRIMARY KEY, description TEXT, x REAL, y REAL, z REAL,
created_at INTEGER)''')
            c.execute('''CREATE TABLE IF NOT EXISTS telemetry (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                room_id TEXT NOT NULL REFERENCES rooms(room_id),
                ts_ms INTEGER NOT NULL,
                temp1 REAL, temp2 REAL, temp3 REAL, temp4 REAL,
                pressure_hpa REAL, humidity_pct REAL, lux REAL,
                co2_ppm INTEGER, bmp_temp_c REAL, quality INTEGER DEFAULT 0)''')
            c.execute('CREATE INDEX IF NOT EXISTS idx_tel_room_ts ON telemetry (room_id,
ts_ms DESC)')
            c.execute('''CREATE TABLE IF NOT EXISTS fault_log (
                fault_id INTEGER PRIMARY KEY AUTOINCREMENT,
                room_id TEXT NOT NULL REFERENCES rooms(room_id),
                ts_ms INTEGER NOT NULL, sensor TEXT, fault_type TEXT, value REAL)''')

    def ensure_room(self, room_id):
        with sqlite3.connect(DB_FILE) as conn:
            conn.execute("INSERT OR IGNORE INTO rooms (room_id, created_at) VALUES (?,
?)",
                         (room_id, int(time.time() * 1000)))

    def log_fault(self, room_id, ts_ms, sensor, fault_type, value):
        with sqlite3.connect(DB_FILE) as conn:
            conn.execute("INSERT INTO fault_log (room_id, ts_ms, sensor, fault_type, value)
VALUES (?, ?, ?, ?, ?)",
                         (room_id, ts_ms, sensor, fault_type, value))

    def add_telemetry(self, data):
        with self.lock:
            self.buffer.append(data)
            # Груповий запис (bulk insert) кожні 5 секунд
            if time.time() - self.last_flush >= 5.0:
                self.flush()

    def flush(self):
        if not self.buffer: return
        with sqlite3.connect(DB_FILE) as conn:
```

```
            conn.executemany("""
                INSERT INTO telemetry
                (room_id, ts_ms, temp1, temp2, temp3, temp4, pressure_hpa, humidity_pct, lux,
co2_ppm, bmp_temp_c, quality)
                VALUES (:room_id, :ts_ms, :temp1, :temp2, :temp3, :temp4, :pressure_hpa,
:humidity_pct, :lux, :co2_ppm, :bmp_temp_c, :quality)
            """, self.buffer)
        logger.info(f"Збережено батч: {len(self.buffer)} записів.")
        self.buffer.clear()
        self.last_flush = time.time()

# ============== ДІАГНОСТИКА (diagnostics.py) ====================
class DiagnosticsEngine:
    def __init__(self, db_writer, mqtt_client):
        self.db = db_writer
        self.mqtt = mqtt_client
        self.windows = {} # {room_id: {sensor: deque(maxlen=60)}}
        self.cusum_pos = {}
        self.latest_pressures = {} # Для міжвузлового аналізу

    def send_alert(self, room_id, alert_type):
        payload = json.dumps({"action": "alert", "alert_type": alert_type, "duration_ms": 1000})
        self.mqtt.publish(f"env/{room_id}/alert", payload, qos=1)

    def process_packet(self, pkt):
        room_id = pkt['room_id']
        ts_ms = pkt['ts_ms']
        quality = 0

        if room_id not in self.windows:
            self.windows[room_id] = {}
            self.cusum_pos[room_id] = {}

        sensors = ['temp1', 'temp2', 'temp3', 'temp4', 'pressure_hpa', 'humidity_pct', 'co2_ppm']

        for sensor in sensors:
            val = pkt.get(sensor)
            if val is None or val == -1: continue

            # --- РІВЕНЬ 1: Фізичні межі ---
            limit_key = 'temp' if sensor.startswith('temp') else sensor
            if limit_key in PHYSICAL_LIMITS:
                l_min, l_max = PHYSICAL_LIMITS[limit_key]
                if not (l_min <= val <= l_max):
                    quality = 2
                    self.db.log_fault(room_id, ts_ms, sensor, "physical_limit", val)
                    self.send_alert(room_id, "critical")
                    continue # Пропускаємо часовий аналіз для невалідних даних

            # --- РІВЕНЬ 2: Часовий аналіз (вікно W=60) ---
```

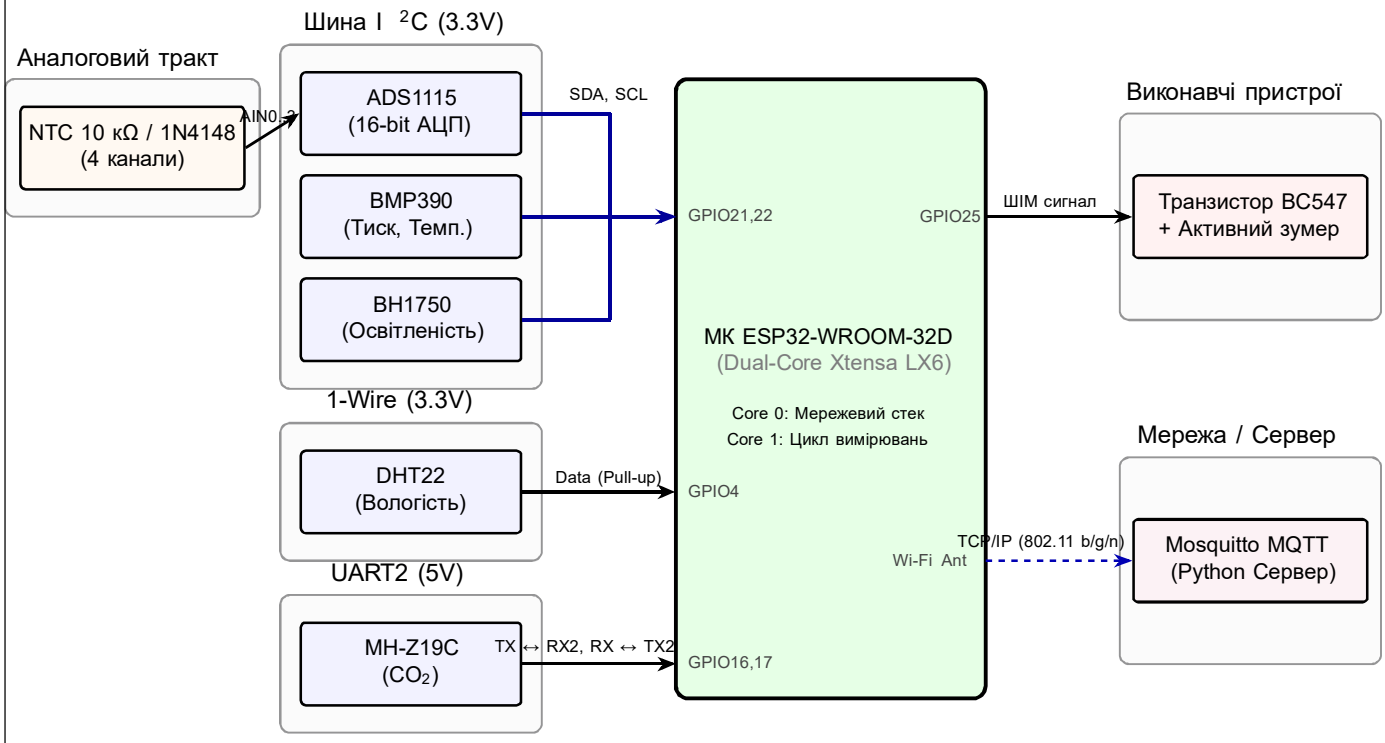
```
            if sensor not in self.windows[room_id]:
                self.windows[room_id][sensor] = deque(maxlen=60)
                self.cusum_pos[room_id][sensor] = 0.0

            window = self.windows[room_id][sensor]
            if len(window) > 0:
                prev_val = window[-1]

                # 1. Spike detection
                spike_thresh = 2.0 if 'temp' in sensor else 500.0 if 'co2' in sensor else 10.0
                if abs(val - prev_val) > spike_thresh:
                    quality = max(quality, 1)
                    self.db.log_fault(room_id, ts_ms, sensor, "spike", val)

                # 2. Stuck detection & Drift (працює при заповненому вікні)
                if len(window) >= 30: # Мінімум 30 значень для статистики
                    variance = statistics.variance(window)
                    stuck_thresh = 0.0001 if 'temp' in sensor else 1.0
                    if variance < stuck_thresh:
                        quality = 2
                        self.db.log_fault(room_id, ts_ms, sensor, "stuck", val)

                    # 3. CUSUM Drift
                    mu_ref = statistics.mean(window)
                    sigma_ref = statistics.stdev(window) or 0.1
                    k = 0.5
                    c_pos = max(0, self.cusum_pos[room_id][sensor] + (val - mu_ref - k * sigma_ref))
                    self.cusum_pos[room_id][sensor] = c_pos
                    if c_pos > 5.0:
                        quality = max(quality, 1)
                        self.db.log_fault(room_id, ts_ms, sensor, "drift", val)
                        self.cusum_pos[room_id][sensor] = 0.0 # Скидання після детекції

            window.append(val)

        # Оновлюємо глобальний словник тиску
        if pkt.get('pressure_hpa'):
            self.latest_pressures[room_id] = pkt['pressure_hpa']

        # --- РІВЕНЬ 3: Міжвузловий аналіз (Тиск) ---
        if len(self.latest_pressures) >= 2 and pkt.get('pressure_hpa'):
            p_med = statistics.median(self.latest_pressures.values())
            if abs(pkt['pressure_hpa'] - p_med) > 5.0:
                quality = 2
                self.db.log_fault(room_id, ts_ms, 'pressure_hpa', "cross_node_diff",
pkt['pressure_hpa'])
                self.send_alert(room_id, "sensor_fault")

        pkt['quality'] = quality
        return pkt
```

```
# ========= ПАРСЕР ТА ВОКЕР (parser.py / time_sync.py) =========
def worker_thread(db_writer, diagnostics):
    while True:
        topic, payload = raw_queue.get()
        try:
            pkt = json.loads(payload)
            room_id = topic.split('/')[1]
            pkt['room_id'] = room_id

            # Валідація обов'язкових полів
            required = ['ts_ms', 'temp1', 'pressure_hpa', 'co2_ppm']
            if not all(k in pkt for k in required):
                logger.warning(f"Відкинуто пакет від {room_id}: відсутні поля")
                continue

            # NTP Синхронізація (time_sync)
            server_ms = int(time.time() * 1000)
            diff_s = abs(server_ms - pkt['ts_ms']) / 1000.0
            if diff_s > 30.0:
                logger.warning(f"NTP CRITICAL для {room_id}. Заміна ts_ms на серверний.")
                pkt['ts_ms'] = server_ms # Заміна ненадійної мітки
            elif diff_s > 5.0:
                logger.warning(f"NTP WARN для {room_id}: розсинхронізація {diff_s:.1f} с")

            db_writer.ensure_room(room_id)

            # Прогон через діагностику
            pkt = diagnostics.process_packet(pkt)

            # Запис у БД
            db_writer.add_telemetry({
                'room_id': pkt['room_id'], 'ts_ms': pkt['ts_ms'],
                'temp1': pkt.get('temp1'), 'temp2': pkt.get('temp2'),
                'temp3': pkt.get('temp3'), 'temp4': pkt.get('temp4'),
                'pressure_hpa': pkt.get('pressure_hpa'), 'humidity_pct': pkt.get('humidity_pct'),
                'lux': pkt.get('lux'), 'co2_ppm': pkt.get('co2_ppm'),
                'bmp_temp_c': pkt.get('bmp_temp_c'), 'quality': pkt.get('quality', 0)
            })
        except json.JSONDecodeError:
            logger.error("Помилка парсингу JSON")
        except Exception as e:
            logger.error(f"Помилка обробки: {e}")
        finally:
            raw_queue.task_done()

# ========= MQTT COLLECTOR (collector.py) ====================
def on_connect(client, userdata, flags, rc):
    logger.info("Підключено до MQTT. Підписка на топіки...")
    client.subscribe("env/+/telemetry", qos=1)
```

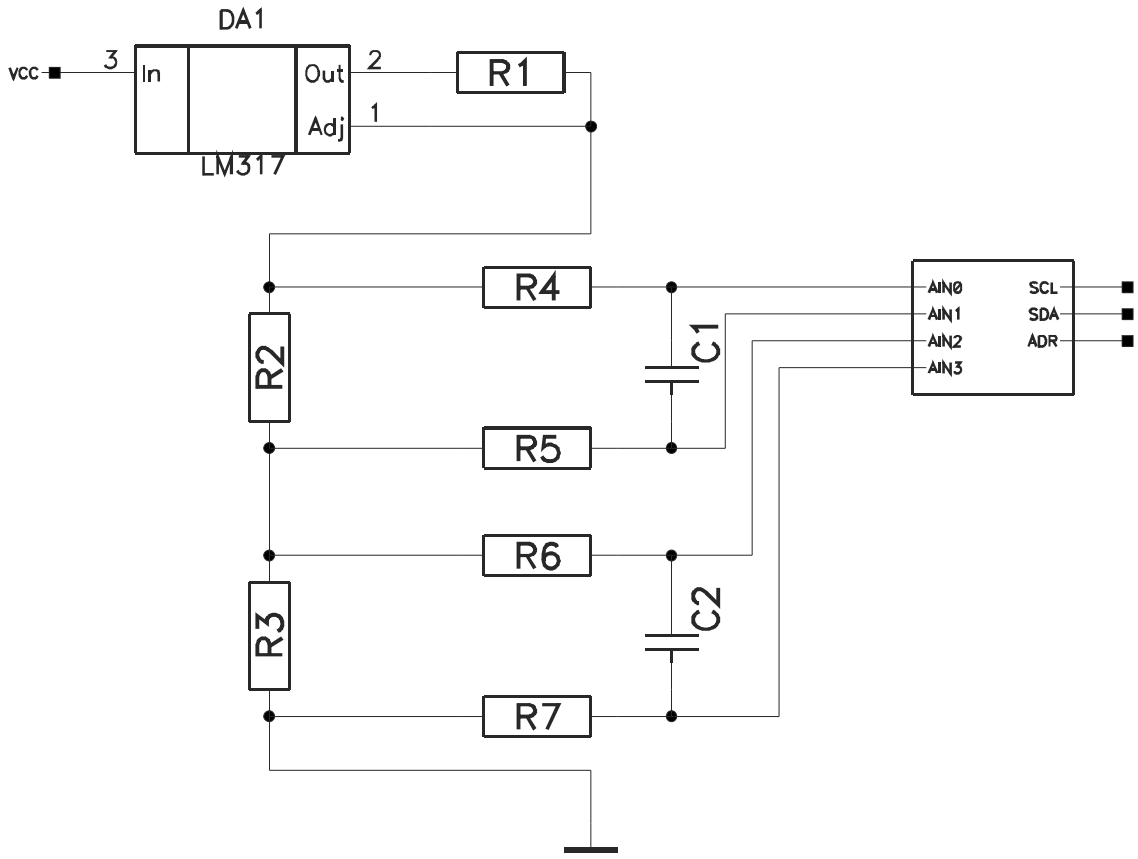
```
    client.subscribe("env/+/status", qos=1)

def on_message(client, userdata, msg):
    # Швидке розміщення у потокобезпечній черзі
    raw_queue.put((msg.topic, msg.payload.decode('utf-8')))

def main():
    db = DBWriter()
    client = mqtt.Client()
    diagnostics = DiagnosticsEngine(db, client)

    # Запуск фонового потоку-обробника
    t = threading.Thread(target=worker_thread, args=(db, diagnostics), daemon=True,
name="WorkerThread")
    t.start()

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

    logger.info("Сервер запущено. Очікування даних...")
    try:
        client.loop_forever()
    except KeyboardInterrupt:
        logger.info("Завершення роботи. Примусовий запис буфера...")
        with db.lock:
            db.flush()

if __name__ == "__main__":
    main()
```



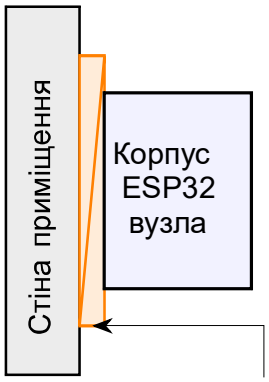
					КРБ.СІ.05.00.00.001			
						Літера	Маса	Мірило
Зм.	Арк.	№докум.	Підпис	Дата				
Розроб.		Лабай			Структура системи збору даних			
Перев.		Волинський						
Т. контр.						Аркуш	Аркушів	
Н. контр.		Возний						
Затв.		Заміховський						



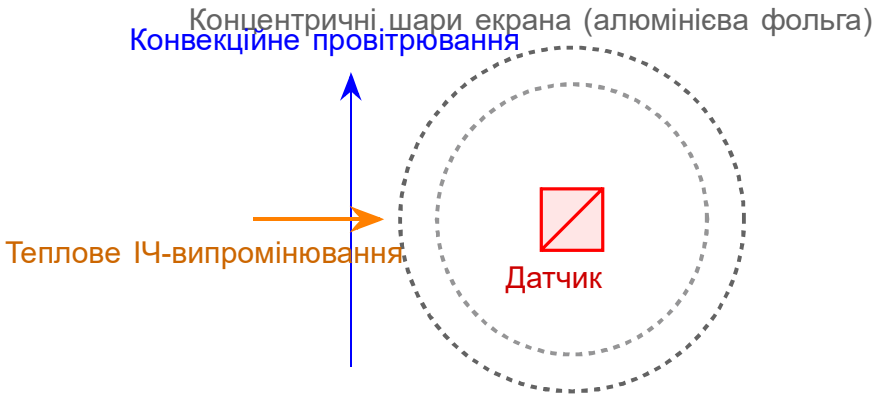
					КРБ.СІ.05.00.00.002		
						Літера	Маса	Мірило
Зм.	Арк.	№докум.	Підпис	Дата	Схема вузла збору даних			
Розроб.		Лабай						
Перев.		Волинський				Аркуш	Аркушів	
Т. контр.								
Н. контр.		Возний						
Затв.		Заміховський						



					КРБ.СІ.05.00.00.003		
						Літера	Маса	Мірило
Зм.	Арк.	№докум.	Підпис	Дата	Схема вузла вимірявання температури термомпорами			
Розроб.		Лабай						
Перев.		Волинський				Аркуш	Аркушів	
Т. контр.								
Н. контр.		Возний						
Затв.		Заміховський						

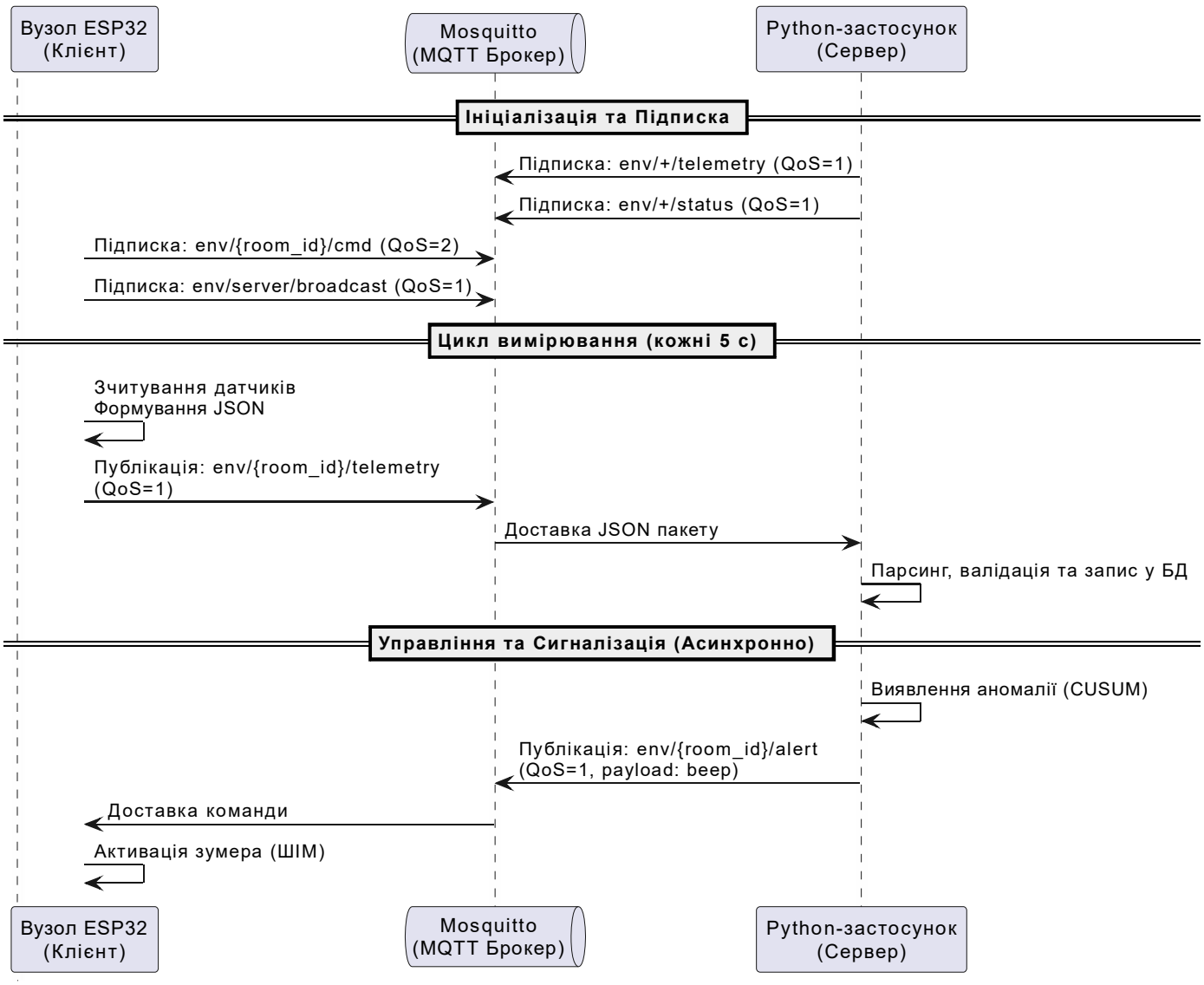


а) Вузол пристінного кріплення

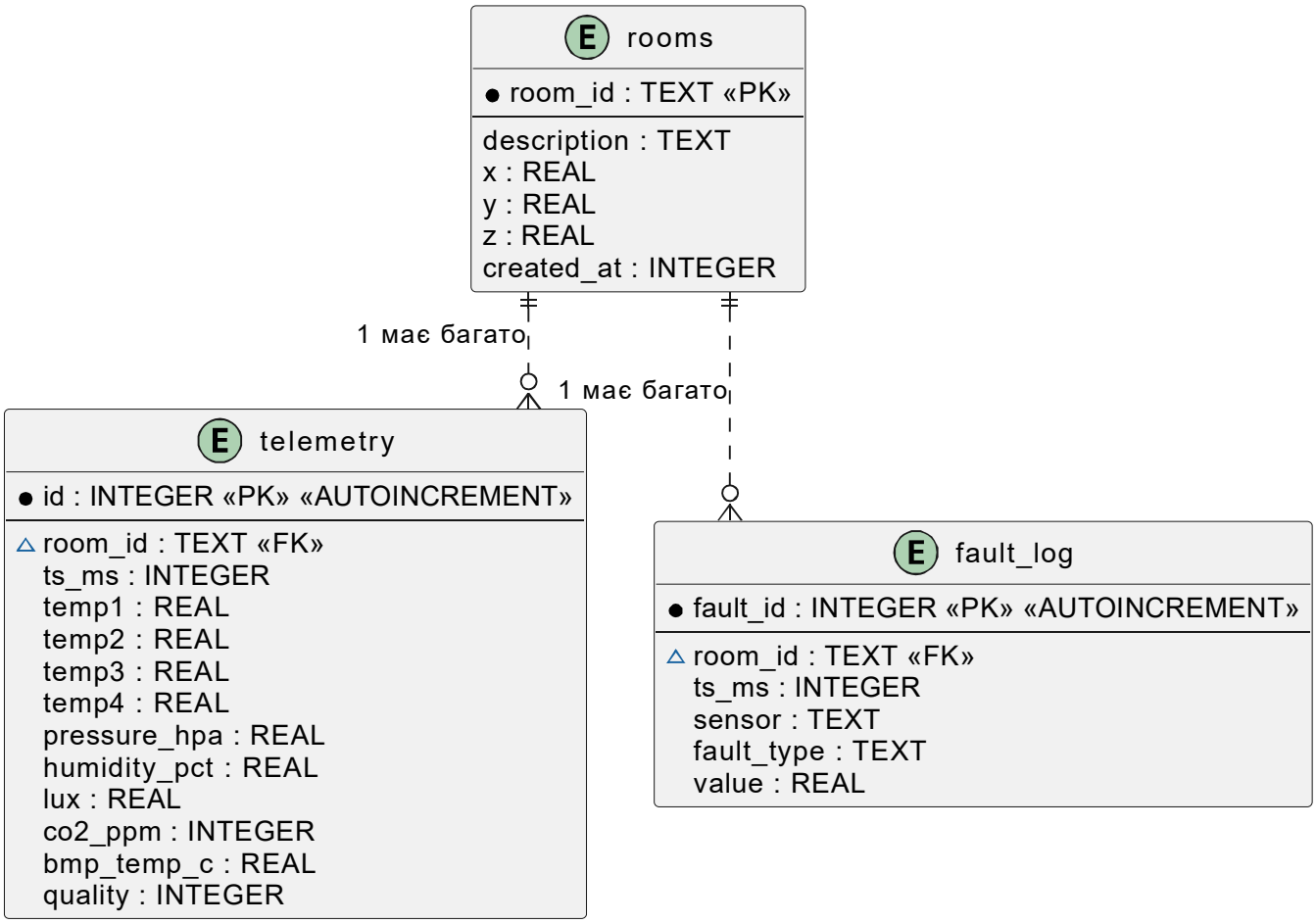


б) Конструкція радіаційного екрана

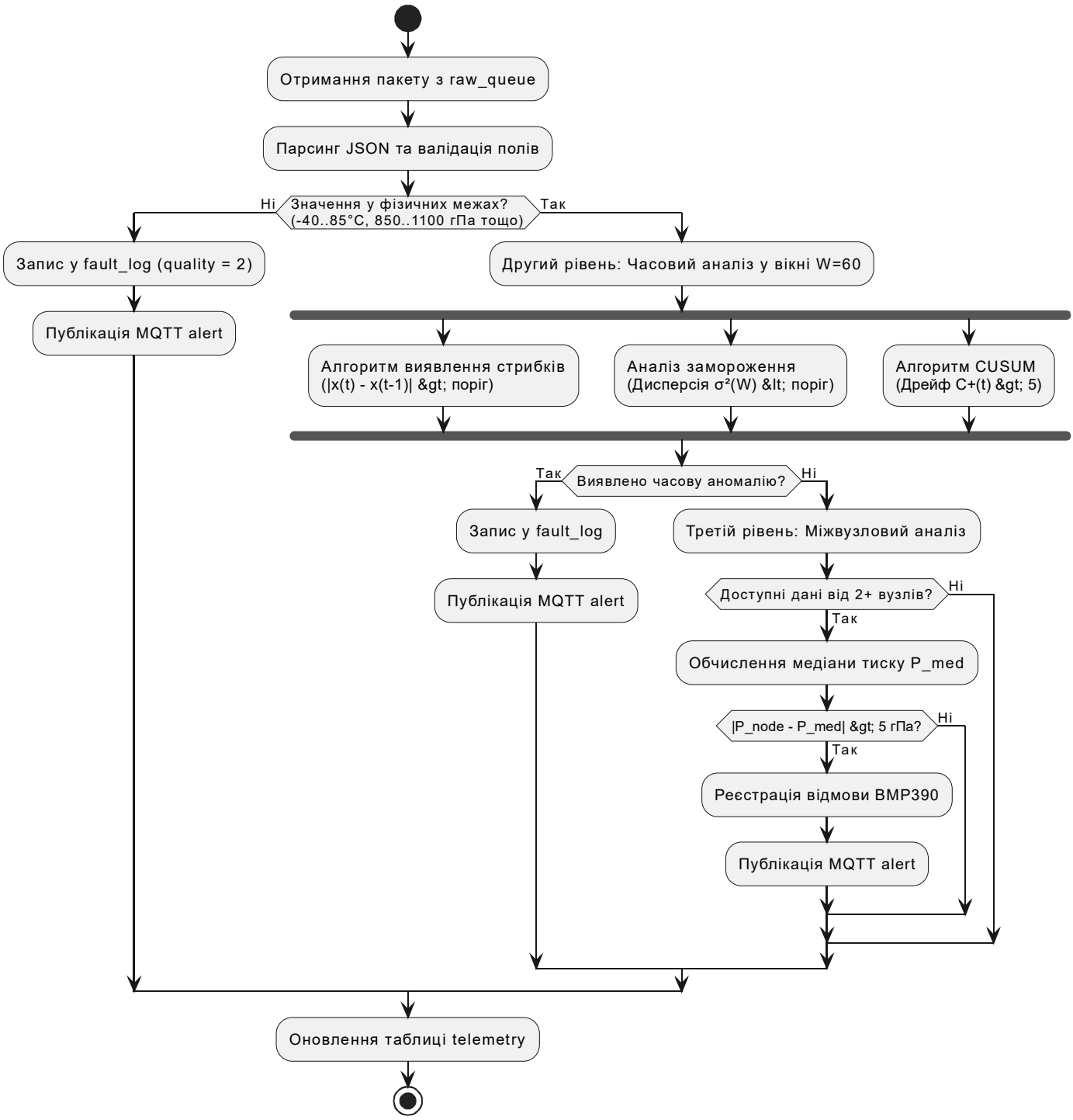
					КРБ.СІ.05.00.00.004		
						Літера	Маса	Мірило
Зм.	Арк.	№докум.	Підпис	Дата				
Розроб.		Лабай			Конструкція температурних датчиків			
Перев.		Волинський						
Т. контр.						Аркуш	Аркушів	
Н. контр.		Возний					
Затв.		Заміховський					



					КРБ.СІ.05.00.00.005		
Зм.	Арк.	№докум.	Підпис	Дата	Діаграма роботи системи	Літера	Маса / Мірило
Розроб.		Лабай					
Перев.		Волинський				Аркуш	Аркушів
Т. контр.							
Н. контр.		Возний					
Затв.		Заміховський					



					КРБ.СІ.05.00.00.006		
						Літера	Маса / Мірило
Зм.	Арк.	№докум.	Підпис	Дата	Структура баз даних		
Розроб.		Лабай					
Перев.		Волинський				Аркуш	Аркушів
Т. контр.							
Н. контр.		Возний					
Затв.		Заміховський					



					КРБ.СІ.05.00.00.007
Зм.	Арк.	№докум.	Підпис	Дата	Алгоритм роботи сервера збору даних
Розроб.		Лабай			
Перев.		Волинський			
Т. контр.					
Н. контр.		Возний			
Затв.		Заміховський			

Літера	Маса	Мірило
Аркуш	Аркушів	

Бібліографічна довідка

Тема роботи: Розроблення апаратно-програмних засобів розподіленої IoT-системи з надлишковими вимірюваннями для виявлення відмов сенсорів та відновлення їх даних

Обсяг бакалаврської роботи 100 сторінки

Перелік графічного матеріалу:

1. Структура системи збору даних
2. Схема вузла зору даних із використанням ESP32
3. Схема вузла вимірювання температури за допмогою термоопорів із 4 дротовим підключенням
4. Конструкція температурних датчиків
5. Діаграма роботи системи
6. Структура бази даних системи
7. Алгоритм роботи системи збору даних

Дата закінчення БР

Студент