Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ І ОБРОБКИ ІОТ-ДАНИХ ТА ВИЯВЛЕННЯ АНОМАЛІЙ ІЗ ВИКОРИСТАННЯМ NODE-RED

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-513сп

Спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ДОЛИННИЙ І.С.
(ПРІЗВИЩЕ та ініціали)

Керівник КУДЕРМЕТОВ Р.К.
(ПРІЗВИЩЕ та ініціали)
Рецензент ХАРИТОНОВ О.Б.
(ПРІЗВИЩЕ та ініціали)

2026

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ
Завідувач кафедри КУДЕРМЕТОВ Р.К.
_____
«14» квітня 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ДОЛИННОГО Ігоря Сергійовича
(ПРІЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка інтелектуальної системи моніторингу і обробки IoT-даних та виявлення аномалій із використанням Node-RED

керівник проєкту (роботи) к.т.н., доцент, КУДЕРМЕТОВ Р.К.,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)
затверджені наказом закладу вищої освіти від « 14 » квітня 2026 року № 160
2. Строк подання студентом проєкту (роботи) 01.06.2026 р.
3. Вихідні дані до проєкту (роботи) опис предметної області, технології розробки клієнтської та серверної частини, технології збору, моніторингу та обробки IoT-даних

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Опис предметної області;
2) Розробка вимог до програмного забезпечення;
3) Моделювання програмного забезпечення;
4) Проєктування архітектури програмного забезпечення;
5) Проєктування інтерфейсу користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	КУДЕРМЕТОВ Р.К.		
Нормоконтроль	ГОЛУБ Т.В.		

7. Дата видачі завдання «14» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	до 18.04.2026	
2	Визначення та аналіз вимог до програмного	до 22.04.2026	
	забезпечення		
3	Розробка специфікації вимог SRS	до 24.04.2026	
4	Розробка діаграми варіантів використання	до 28.04.2026	
5	Розробка діаграми послідовності	до 01.05.2026	
6	Розробка описів прецедентів	до 05.05.2026	
7	Проєктування архітектури програмного	до 12.05.2026	
	забезпечення		
8	Проєктування інтерфейсу користувача	до 22.05.2026	
9	Оформлення пояснювальної записки	до 25.05.2026	
10	Проходження нормоконтролю	до 01.06.2026	
11	Перевірка на наявність академічного плагіату	до 01.06.2026	
12	Проходження рецензування	до 01.06.2026	

Студент(ка) _____ Ігор ДОЛИННИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи) _____ Равіль КУДЕРМЕТОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 81 с., 7 табл., 49 рис., 2 дод., 11 джерел.

ІНТЕРНЕТ РЕЧЕЙ, IOT, NODE-RED, МОНІТОРИНГ ДАНИХ, ОБРОБКА ДАНИХ, ВИЯВЛЕННЯ АНОМАЛІЙ, MQTT, ІНТЕЛЕКТУАЛЬНА СИСТЕМА, ІНФОРМАЦІЙНА ПАНЕЛЬ

Об’єкт розробки – інтелектуальна система моніторингу та обробки IoT-даних.

Мета роботи – розробка системи моніторингу та обробки даних, отриманих від IoT-пристроїв, із можливістю автоматичного виявлення аномалій у потоці даних засобами Node-RED.

У процесі виконання роботи було проведено аналіз предметної області Інтернету речей та сучасних технологій моніторингу IoT-пристроїв. Визначено функціональні та нефункціональні вимоги до програмного забезпечення, виконано моделювання системи та спроєктовано її архітектуру. Для передачі даних використано протокол MQTT, а для їх обробки та візуалізації – платформу Node-RED.

Розроблена система забезпечує збір, обробку та відображення даних у режимі реального часу. Для виявлення аномальних ситуацій застосовується механізм аналізу вхідних даних на основі заданих порогових значень. Інтерфейс користувача надає можливість перегляду поточних показників, графіків зміни параметрів та повідомлень про виявлені відхилення.

Результатом роботи є інтелектуальна система моніторингу IoT-даних, яка може використовуватися для контролю стану обладнання, аналізу показників сенсорів та оперативного виявлення аномалій.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IoT	–	Internet of Things, Інтернет речей
MQTT	–	Message Queuing Telemetry Transport, протокол обміну повідомленнями для IoT-пристроїв
API	–	Application Programming Interface, інтерфейс програмування застосунків
GUI	–	Graphical User Interface, графічний інтерфейс користувача
UI	–	User Interface, інтерфейс користувача
JSON	–	JavaScript Object Notation, формат обміну даними
TCP/IP	–	Transmission Control Protocol / Internet Protocol, стек мережевих протоколів
IDE	–	Integrated Development Environment, інтегроване середовище розробки
TTS	–	Text-to-Speech, технологія синтезу мовлення
Node-RED	–	середовище потокового програмування для обробки даних та інтеграції IoT-пристроїв
Dashboard	–	вебінтерфейс моніторингу та візуалізації даних у Node-RED
AQI	–	Air Quality Index, індекс якості повітря
CSV	–	Comma-Separated Values, формат зберігання табличних даних
HTTP	–	HyperText Transfer Protocol, протокол передачі гіпертексту

ВСТУП

Стрімкий розвиток інформаційних технологій та цифрової трансформації сприяв широкому впровадженню концепції Інтернету речей (IoT, Internet of Things) у різних сферах діяльності людини. Сучасні IoT-системи використовуються в промисловості, енергетиці, транспорті, сільському господарстві, медицині та системах «розумного» будинку. Основною особливістю таких систем є можливість збору великої кількості даних від різноманітних датчиків та пристроїв у режимі реального часу з подальшою їх обробкою, аналізом та використанням для прийняття рішень.

З кожним роком кількість IoT-пристроїв невпинно зростає, що призводить до збільшення обсягів даних, які необхідно обробляти та контролювати. При цьому важливим завданням стає забезпечення безперервного моніторингу стану пристроїв та своєчасне виявлення відхилень від нормального режиму роботи. Несвоєчасне виявлення аномалій може призвести до порушення роботи обладнання, виникнення аварійних ситуацій, втрати даних або зниження ефективності функціонування всієї системи.

Одним із перспективних напрямків розвитку IoT-систем є використання інтелектуальних засобів аналізу даних, які дозволяють автоматизувати процес моніторингу та оперативно реагувати на нештатні ситуації. Інтелектуальні системи моніторингу здатні аналізувати потоки даних у режимі реального часу, визначати критичні значення параметрів та виявляти аномальні події без безпосереднього втручання користувача. Це підвищує надійність функціонування системи та зменшує навантаження на операторів.

Для реалізації подібних рішень широко застосовуються сучасні програмні платформи, які забезпечують інтеграцію різноманітних джерел даних та інструментів їх обробки. Однією з таких платформ є Node-RED – середовище візуального програмування, що дозволяє швидко створювати потоки обробки даних, інтегрувати IoT-пристрої, реалізовувати механізми моніторингу та

будувати інформаційні панелі для відображення результатів роботи системи. Використання Node-RED значно спрощує процес розробки завдяки наявності готових компонентів для роботи з мережевими протоколами, базами даних та сервісами обміну повідомленнями.

Актуальність теми дипломної роботи обумовлена необхідністю створення ефективних засобів моніторингу та аналізу IoT-даних, які забезпечують оперативне виявлення аномалій та підвищують надійність функціонування інформаційних систем. В умовах постійного збільшення кількості підключених пристроїв питання автоматизованої обробки даних та контролю їхнього стану набувають особливої важливості.

Об’єктом дослідження є процеси моніторингу та обробки даних, що надходять від IoT-пристроїв.

Предметом дослідження є методи та засоби побудови інтелектуальних систем моніторингу IoT-даних із використанням платформи Node-RED та механізмів виявлення аномалій.

Метою роботи є розробка інтелектуальної системи моніторингу і обробки IoT-даних та виявлення аномалій із використанням Node-RED.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз предметної області Інтернету речей та існуючих підходів до моніторингу IoT-пристроїв;

– визначити функціональні та нефункціональні вимоги до програмного забезпечення;

– виконати моделювання програмної системи за допомогою UML-діаграм;

– спроєктувати архітектуру системи моніторингу та обробки IoT-даних;

– розробити інтерфейс користувача для відображення поточних показників та повідомлень про аномалії;

– дослідити можливості використання Node-RED для реалізації систем моніторингу та аналізу даних.

Практична цінність роботи полягає у можливості використання розробленого рішення для збору, обробки та аналізу даних від IoT-пристроїв у

режимі реального часу. Запропонована система може застосовуватися для контролю стану обладнання, моніторингу параметрів навколишнього середовища та своєчасного виявлення аномальних ситуацій у різних сферах діяльності.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Розвиток інформаційних технологій характеризується стрімким зростанням кількості підключених пристроїв та широким впровадженням концепції Інтернету речей (IoT – Internet of Things). IoT-системи дозволяють об’єднувати фізичні пристрої, датчики та програмні сервіси в єдину мережу для збору, передачі та обробки даних у режимі реального часу. Такі системи активно застосовуються в промисловості, енергетиці, транспорті, медицині та системах розумного міста.

Основною особливістю IoT-середовища є безперервний потік даних, який генерується великою кількістю пристроїв. Ці дані можуть включати показники температури, вологості, тиску, руху, стану обладнання та інші параметри фізичного середовища. У зв’язку з цим виникає необхідність у створенні ефективних механізмів їх обробки, аналізу та візуалізації.

Однією з ключових проблем IoT-систем є своєчасне виявлення аномальних значень у потоках даних. Аномалії можуть свідчити про несправність пристроїв, критичні зміни середовища або нештатні ситуації в роботі системи. Відсутність автоматизованого механізму їх виявлення може призвести до втрати даних або порушення роботи всієї системи моніторингу.

Для передачі даних у IoT-системах широко використовується протокол обміну повідомленнями MQTT (Message Queuing Telemetry Transport). Даний протокол базується на принципі публікації та підписки (publish/subscribe), що дозволяє ефективно передавати дані між великою кількістю пристроїв навіть у мережах з обмеженою пропускною здатністю. MQTT є одним із стандартів у сфері IoT завдяки своїй простоті, низькому навантаженню та підтримці асинхронної передачі даних.

У даній роботі для реалізації системи передачі даних використовується Python-скрипт, який виконує роль емулятора IoT-пристрою. Він генерує телеметричні дані та надсилає їх до MQTT-брокера, розміщеного у хмарному середовищі emqx.io. Такий підхід дозволяє моделювати роботу реальних

сенсорних пристроїв та тестувати систему без фізичного обладнання.

Для обробки та аналізу отриманих даних використовується платформа Node-RED – інструмент візуального програмування, призначений для побудови потоків обробки даних. Node-RED дозволяє інтегрувати різні сервіси, пристрої та протоколи без необхідності написання великої кількості програмного коду. Завдяки використанню графічного підходу система обробки даних може бути швидко розроблена, налаштована та змінена.

У межах розроблюваної системи Node-RED виконує функції прийому даних через MQTT, їх первинної обробки, розподілу по потоках та аналізу. Отримані дані проходять етап структурування, після чого використовуються як для візуалізації, так і для виявлення аномалій.

Окремою важливою задачею є реалізація механізму виявлення аномалій у режимі реального часу. У даній роботі застосовується підхід порогового контролю, при якому значення параметрів порівнюються із заздалегідь визначеними допустимими межами. У разі перевищення або зниження значень за допустимі межі система фіксує аномалію та формує відповідне повідомлення для користувача. Такий підхід є простим у реалізації та ефективним для базових IoT-систем моніторингу.

Результати обробки даних відображаються у вигляді інформаційної панелі (dashboard), яка реалізована у Node-RED. Вона містить графічні елементи для візуалізації поточних значень, зокрема індикатори (gauge) та графіки (chart), що дозволяють відстежувати динаміку зміни параметрів у часі. Також у системі реалізовано підсистему сповіщень, яка повідомляє користувача про виявлені аномалії за допомогою текстових повідомлень та звукових сигналів.

Таким чином, предметна область даної роботи охоплює технології Інтернету речей, протоколи передачі даних MQTT, інструменти потокової обробки інформації Node-RED, а також методи виявлення аномалій у телеметричних даних. Поєднання цих технологій дозволяє створити інтелектуальну систему моніторингу, яка забезпечує збір, обробку, аналіз та візуалізацію IoT-даних у режимі реального часу.

1.1 Поняття та основні принципи Інтернету речей

Інтернет речей (Internet of Things, IoT) є одним із найбільш перспективних напрямів розвитку сучасних інформаційних технологій. Концепція IoT передбачає об’єднання фізичних об’єктів у єдину інформаційну мережу, що забезпечує автоматичний збір, передачу, обробку та аналіз даних без безпосереднього втручання людини. Завдяки використанню датчиків, виконавчих механізмів, мережевих технологій та програмного забезпечення створюються інтелектуальні системи, здатні контролювати стан об’єктів і навколишнього середовища в режимі реального часу.

Основною особливістю Інтернету речей є можливість інтеграції фізичних пристроїв із цифровим середовищем. Кожний пристрій може бути оснащений одним або декількома датчиками, які здійснюють вимірювання певних параметрів, таких як температура, вологість повітря, атмосферний тиск, якість повітря, рівень освітлення або інші показники. Отримані дані передаються через мережу до спеціалізованих систем обробки, де виконуються їх аналіз, зберігання та візуалізація.

Сучасні IoT-системи складаються з декількох основних компонентів. Першим компонентом є рівень збору даних, який представлений різноманітними датчиками та вимірювальними пристроями. Саме на цьому рівні відбувається формування первинної інформації про стан контрольованого об’єкта або середовища. Другим компонентом є комунікаційний рівень, що забезпечує передачу даних між пристроями та серверними системами. Для цього можуть використовуватися як дротові, так і бездротові технології зв’язку. Третім компонентом виступає рівень обробки та зберігання інформації, де здійснюється аналіз отриманих даних, виявлення закономірностей та прийняття рішень. Останнім компонентом є рівень представлення інформації користувачу, який забезпечує візуалізацію даних за допомогою графіків, індикаторів, таблиць та інших засобів інтерфейсу.

Важливим аспектом функціонування IoT-систем є можливість роботи в режимі реального часу. Дані, які надходять від сенсорів, повинні оперативно оброблятися та відображатися користувачеві. Це дозволяє своєчасно реагувати на зміни контрольованих параметрів та виявляти потенційно небезпечні ситуації. Особливо актуальним це є для промислових систем моніторингу, систем безпеки, транспортної інфраструктури та об’єктів критичної інфраструктури.

Однією з ключових переваг технології Інтернету речей є автоматизація процесів моніторингу та контролю. Використання датчиків дозволяє значно зменшити вплив людського фактору та підвищити точність отриманих даних. Крім того, централізоване накопичення інформації відкриває можливості для подальшого аналізу, прогнозування та оптимізації роботи різних систем.

З кожним роком кількість підключених до мережі IoT-пристроїв постійно зростає. Це пов’язано зі зменшенням вартості електронних компонентів, розвитком бездротових мереж та збільшенням обчислювальних можливостей сучасних комп’ютерних систем. За оцінками аналітичних компаній, кількість IoT-пристроїв у світі вже обчислюється десятками мільярдів одиниць, а сфери їх застосування охоплюють практично всі галузі людської діяльності.

Серед основних напрямків використання Інтернету речей можна виділити системи розумного будинку, моніторинг навколишнього середовища, промислову автоматизацію, транспортні системи, сільське господарство, медицину та енергетику. У кожній із цих сфер IoT-технології дозволяють підвищити ефективність роботи обладнання, покращити якість обслуговування та забезпечити більш раціональне використання ресурсів.

Таким чином, Інтернет речей є важливою складовою сучасного цифрового суспільства та створює основу для побудови інтелектуальних систем моніторингу й управління. Використання IoT-технологій дозволяє отримувати актуальну інформацію про стан об’єктів у режимі реального часу, автоматизувати процеси збору та обробки даних, а також забезпечувати своєчасне виявлення відхилень від нормального режиму функціонування систем.

1.2 Технології передачі даних в IoT-системах

Одним із ключових елементів будь-якої системи Інтернету речей є механізм передачі даних між пристроями, датчиками та програмними компонентами. Ефективність функціонування IoT-систем значною мірою залежить від швидкості обміну повідомленнями, надійності доставки даних, рівня використання мережевих ресурсів та можливості роботи в умовах нестабільного з'єднання. Саме тому вибір технології передачі даних є важливим етапом проєктування систем моніторингу та керування.

У сучасних IoT-рішеннях використовуються різні мережеві протоколи та технології обміну даними. Найбільш поширеними серед них є HTTP, MQTT та CoAP. Кожен із зазначених протоколів має власні особливості, переваги та сфери застосування.

Протокол HTTP (HyperText Transfer Protocol) є одним із найпоширеніших протоколів мережі Інтернет. Його основною перевагою є універсальність та широка підтримка практично всіма сучасними програмними платформами. Однак використання HTTP у системах Інтернету речей має певні обмеження. Передача кожного повідомлення супроводжується значною кількістю службової інформації, що збільшує навантаження на мережу та знижує ефективність роботи пристроїв із обмеженими обчислювальними ресурсами.

Для усунення зазначених недоліків у сфері IoT широкого поширення набув протокол MQTT (Message Queuing Telemetry Transport). Він був розроблений спеціально для передачі телеметричних даних у мережах із низькою пропускною здатністю та нестабільним з'єднанням. MQTT використовує модель взаємодії Publish/Subscribe, яка передбачає наявність проміжного сервера — брокера повідомлень. Пристрої-публікатори передають дані брокеру, а клієнти-підписники отримують повідомлення відповідно до тем підписки.

Використання MQTT дозволяє мінімізувати обсяг службового трафіку та забезпечити ефективну передачу повідомлень навіть при обмежених ресурсах

мережі. Завдяки цьому протокол широко використовується в системах моніторингу навколишнього середовища, промисловій автоматизації, розумних будинках та інших IoT-рішеннях.

Ще одним популярним протоколом є CoAP (Constrained Application Protocol), який розроблений для пристроїв із дуже обмеженими ресурсами. Він використовує принципи REST-архітектури та дозволяє здійснювати обмін повідомленнями з мінімальними витратами мережевого трафіку. Проте через меншу поширеність та складнішу інтеграцію CoAP використовується рідше, ніж MQTT.

Для порівняння характеристик основних протоколів передачі даних в IoT-системах наведено таблицю 1.2.

Таблиця 1.2 – Порівняння протоколів передачі даних для IoT-систем

Характеристика	HTTP	MQTT	CoAP
Модель взаємодії	Request/Response	Publish/Subscribe	Request/Response
Обсяг службових даних	Високий	Низький	Дуже низький
Підтримка реального часу	Середня	Висока	Висока
Складність реалізації	Низька	Низька	Середня
Використання в IoT	Обмежене	Широке	Спеціалізоване

Особливу увагу при проєктуванні систем моніторингу необхідно приділяти надійності доставки повідомлень. Для цього протокол MQTT підтримує декілька рівнів якості обслуговування (QoS). Рівень QoS 0 забезпечує доставку повідомлення без підтвердження отримання. Рівень QoS 1 гарантує доставку повідомлення хоча б один раз, а рівень QoS 2 забезпечує гарантовану одноразову доставку. Завдяки цьому розробник може обирати оптимальний баланс між швидкістю передачі та надійністю роботи системи.

Важливим компонентом MQTT-архітектури є брокер повідомлень. Саме він відповідає за приймання повідомлень від пристроїв та їх подальше розповсюдження серед підписників. У межах даної роботи використовується

публічний брокер MQTT, який забезпечує взаємодію між програмою генерації даних та системою моніторингу, реалізованою у середовищі Node-RED.

Для передачі даних від IoT-пристроїв можуть використовуватися як локальні мережі, так і глобальна мережа Інтернет. Сучасні рішення підтримують використання Wi-Fi, Ethernet, мобільних мереж зв’язку та інших технологій комунікації. Вибір конкретного способу підключення визначається вимогами до швидкості передачі даних, енергоспоживання та доступності мережевої інфраструктури.

У рамках розроблюваної системи моніторингу використання MQTT є найбільш доцільним рішенням. Даний протокол забезпечує швидку передачу телеметричних даних, підтримує роботу в режимі реального часу та легко інтегрується із середовищем Node-RED. Це дозволяє реалізувати ефективний механізм збору, обробки та візуалізації інформації від IoT-пристроїв із мінімальними витратами мережевих ресурсів.

Таким чином, сучасні технології передачі даних відіграють важливу роль у функціонуванні систем Інтернету речей. Серед існуючих рішень протокол MQTT є одним із найбільш придатних для побудови систем моніторингу в реальному часі завдяки своїй простоті, надійності та ефективності використання мережевих ресурсів.

1.3 Системи моніторингу та візуалізації даних

Одним із найважливіших компонентів сучасних IoT-систем є підсистема моніторингу та візуалізації даних. Вона забезпечує збір інформації від датчиків, її обробку, збереження та подальше відображення у зрозумілому для користувача вигляді. Завдяки використанню сучасних засобів моніторингу оператор може в режимі реального часу контролювати стан обладнання, аналізувати зміни параметрів та своєчасно реагувати на можливі відхилення.

Традиційно для моніторингу технологічних процесів використовувалися SCADA-системи (Supervisory Control and Data Acquisition). Такі системи широко застосовуються у промисловості для збору даних від контролерів, датчиків та виконавчих механізмів. Основною перевагою SCADA є можливість централізованого контролю великої кількості пристроїв, візуалізації процесів та ведення архівів даних. Разом із тим впровадження класичних SCADA-рішень часто потребує значних фінансових витрат та складної інфраструктури.

З розвитком вебтехнологій почали активно використовуватися сучасні вебпанелі моніторингу. Такі рішення працюють через браузер і дозволяють отримувати доступ до даних з будь-якого пристрою, підключеного до мережі Інтернет. Вебпанелі забезпечують високу гнучкість налаштування, підтримують інтерактивні елементи керування та можуть легко інтегруватися з різними джерелами даних.

Особливого поширення серед розробників IoT-рішень набула платформа Node-RED. Вона є середовищем візуального програмування, яке дозволяє створювати складні потоки обробки даних за допомогою графічних блоків без необхідності написання великої кількості програмного коду. Node-RED підтримує широкий спектр мережевих протоколів, серед яких MQTT, HTTP, WebSocket та інші.

Однією з ключових переваг Node-RED є наявність вбудованого Dashboard-модуля, який дозволяє створювати інформаційні панелі моніторингу. Dashboard надає набір готових елементів інтерфейсу, зокрема графіки, індикатори, таблиці, текстові повідомлення та кнопки керування. Завдяки цьому розробник може швидко створити вебінтерфейс для візуалізації телеметричних даних без використання додаткових фреймворків.

Для порівняння засобів моніторингу даних наведено таблицю 1.3.

Таблиця 1.3 – Порівняння засобів моніторингу даних

Система	Переваги	Недоліки
SCADA	Висока надійність, підтримка промислових стандартів	Висока вартість впровадження

Продовження таблиці 1.3

Система	Переваги	Недоліки
Вебпанелі моніторингу	Доступність через браузер, гнучкість налаштування	Залежність від мережі
Node-RED Dashboard	Простота створення інтерфейсів, інтеграція з MQTT	Обмежені можливості складної аналітики

Важливим аспектом моніторингу є правильне відображення телеметричних даних. Телеметрія являє собою інформацію, що надходить від датчиків та інших пристроїв у вигляді числових значень, повідомлень або подій. Для підвищення зручності сприйняття інформації використовуються різні засоби візуалізації.

Найбільш поширеними елементами відображення телеметричних даних є графіки зміни параметрів у часі, цифрові індикатори та кругові шкали (Gauge). Графіки дозволяють аналізувати тенденції зміни показників та виявляти закономірності в роботі системи. Цифрові індикатори забезпечують відображення поточних значень параметрів, а шкали дозволяють швидко оцінювати рівень показника відносно допустимих меж.

У розробленій системі для моніторингу використовуються шість основних параметрів навколишнього середовища: температура, вологість повітря, атмосферний тиск, швидкість вітру, кількість опадів та якість повітря. Для кожного параметра створено окремі графіки та індикатори, що дозволяє користувачу оперативно оцінювати поточний стан контрольованого об'єкта.

Таким чином, сучасні системи моніторингу та візуалізації даних є невід'ємною складовою IoT-рішень. Використання платформи Node-RED та інформаційних панелей Dashboard забезпечує ефективне представлення телеметричної інформації в режимі реального часу та створює зручні умови для аналізу даних і прийняття рішень оператором системи.

1.4 Методи виявлення аномалій у потоках даних

У сучасних інформаційних системах моніторингу одним із найважливіших завдань є своєчасне виявлення аномалій у потоці даних. Зростання кількості IoT-пристроїв призводить до безперервного надходження великого обсягу інформації від різноманітних сенсорів та вимірювальних приладів. Ручний аналіз таких даних є складним та малоефективним, тому актуальним стає використання автоматизованих методів виявлення відхилень від нормального режиму роботи системи.

Під аномалією розуміють значення параметра або сукупність подій, які суттєво відрізняються від очікуваної поведінки системи. Аномалії можуть свідчити про несправність обладнання, помилки передачі даних, вплив зовнішніх факторів або виникнення аварійних ситуацій. Наприклад, різке підвищення температури датчика, раптове зростання рівня забруднення повітря чи нестандартні зміни атмосферного тиску можуть сигналізувати про необхідність негайного реагування оператора.

Залежно від характеру даних та складності системи застосовуються різні підходи до виявлення аномалій. Найпростішими та найбільш поширеними є порогові методи аналізу. Їх принцип роботи полягає у визначенні допустимого діапазону значень для кожного контрольованого параметра. Якщо отримане значення виходить за встановлені межі, система формує повідомлення про аномалію.

Пороговий підхід є особливо ефективним для моніторингу фізичних параметрів навколишнього середовища, оскільки більшість датчиків працює в межах відомих нормативних значень. Наприклад, температура повітря може контролюватися в межах від –20 °C до +50 °C, відносна вологість — від 0 до 100 %, а атмосферний тиск — у визначеному діапазоні залежно від місцевості.

У таблиці 1.4 наведено приклади порогових значень для контролю параметрів.

Таблиця 1.4 – Приклади порогових значень для контролю параметрів

Параметр	Нижня межа	Верхня межа
Температура	-20°C	+0°C
Вологість	0%	100%
Атмосферний тиск	950 гПа	1050 гПа
Швидкість вітру	0 м/с	30 м/с
Якість повітря	0	200
Опади	0 мм	100 мм

Перевагами порогового аналізу є простота реалізації, мінімальні обчислювальні витрати та висока швидкість обробки інформації. Саме тому даний метод широко використовується в системах реального часу та вбудованих рішеннях, де обчислювальні ресурси є обмеженими.

Більш складним підходом є статистичний аналіз даних. У цьому випадку аномалія визначається як значення, що істотно відхиляється від середнього рівня або статистичного розподілу даних. Для цього можуть використовуватися середнє арифметичне значення, медіана, дисперсія, стандартне відхилення та інші статистичні показники.

Статистичні методи дозволяють враховувати особливості поведінки параметрів у часі та зменшувати кількість хибних спрацьовувань. Наприклад, короткочасне коливання температури може не вважатися аномалією, якщо воно знаходиться в межах нормального статистичного розподілу. Водночас реалізація таких методів потребує накопичення історичних даних та додаткових обчислювальних ресурсів.

Останніми роками значного розвитку набули методи машинного навчання для виявлення аномалій. Вони базуються на побудові математичних моделей, які здатні самостійно визначати закономірності в даних та знаходити нестандартні ситуації. До найбільш поширених підходів належать нейронні мережі, дерева рішень, метод опорних векторів, кластеризація та алгоритми глибокого навчання.

Основною перевагою машинного навчання є можливість виявлення складних аномалій, які неможливо описати простими правилами або пороговими

значеннями. Такі системи можуть враховувати взаємозв'язок між декількома параметрами одночасно та адаптуватися до змін умов роботи. Однак їх використання пов'язане з необхідністю наявності великого обсягу навчальних даних, значних обчислювальних ресурсів та складнішого процесу налаштування.

У таблиці 1.5 наведено порівняння основних методів виявлення аномалій.

Таблиця 1.5 – Порівняння основних методів виявлення аномалій

Метод	Переваги	Недоліки
Пороговий аналіз	Простота реалізації, швидкодія	Не враховує складні закономірності
Статистичний аналіз	Менше хибних спрацьовувань	Потребує історичних даних
Машинне навчання	Висока точність, адаптивність	Складність реалізації та навчання

У рамках даної роботи для виявлення аномалій обрано пороговий метод аналізу. Таке рішення пояснюється особливостями проєктованої системи моніторингу IoT-даних. Контрольовані параметри мають чітко визначені допустимі межі, а сама система повинна забезпечувати швидке реагування на відхилення в режимі реального часу.

Крім того, пороговий аналіз легко реалізується засобами середовища Node-RED за допомогою функціональних вузлів та умовної логіки. Це дозволяє мінімізувати навантаження на систему та забезпечити стабільну роботу навіть при безперервному надходженні великого потоку телеметричних даних.

Таким чином, існує значна кількість методів виявлення аномалій у потоках даних, які відрізняються рівнем складності, точністю та вимогами до ресурсів. Для невеликих IoT-систем моніторингу найбільш доцільним є використання порогового аналізу, який забезпечує просту реалізацію, високу швидкодію та достатню ефективність при контролі параметрів навколишнього середовища та технічного обладнання.

1.5 Сучасний стан розвитку систем моніторингу IoT-даних

Стрімкий розвиток інформаційних технологій та концепції Інтернету речей призвів до появи великої кількості систем моніторингу, здатних здійснювати безперервний контроль фізичних об'єктів, технологічних процесів та навколишнього середовища. Сучасні IoT-системи активно використовуються у промисловості, транспортній сфері, енергетиці, медицині, аграрному секторі та побуті. Зростання кількості підключених пристроїв сприяє формуванню нових підходів до збору, обробки та аналізу даних, що робить моніторинг одним із ключових напрямів розвитку цифрових технологій.

Однією з найважливіших концепцій цифрової трансформації виробництва є Industry 4.0. Вона передбачає інтеграцію кіберфізичних систем, автоматизованого обладнання, датчиків та програмного забезпечення в єдине інформаційне середовище. Основною метою такого підходу є підвищення ефективності виробничих процесів шляхом постійного контролю параметрів обладнання та оперативного реагування на зміни його стану.

У межах концепції Industry 4.0 системи моніторингу виконують завдання збору телеметричних даних від виробничого обладнання, контролю технологічних параметрів та прогнозування можливих несправностей. Використання інтелектуального аналізу даних дозволяє переходити від планового технічного обслуговування до прогнозного, що значно знижує витрати підприємств та підвищує надійність роботи обладнання.

Ще одним перспективним напрямом застосування IoT-технологій є концепція Smart City. Вона передбачає використання мережі взаємопов'язаних пристроїв для управління міською інфраструктурою. До основних сфер застосування належать моніторинг якості повітря, контроль транспортних потоків, управління системами освітлення, енергоспоживанням та безпекою населення.

У розумних містах тисячі датчиків безперервно генерують великі обсяги

інформації, яка повинна оперативно оброблятися та аналізуватися. Системи моніторингу дозволяють отримувати актуальні дані про стан міського середовища та забезпечують підтримку прийняття управлінських рішень на основі реальних показників.

Широкого поширення набули також системи розумного будинку (Smart Home). Такі рішення дозволяють автоматизувати управління освітленням, опаленням, вентиляцією, системами безпеки та іншими побутовими пристроями. Центральним елементом розумного будинку є система моніторингу, яка збирає дані від датчиків температури, вологості, руху, освітленості та інших параметрів.

Перевагою використання IoT у житлових приміщеннях є можливість постійного контролю умов навколишнього середовища та автоматичного реагування на зміни показників. Наприклад, система може самостійно активувати вентиляцію при погіршенні якості повітря або надсилати повідомлення користувачу про виявлення аварійної ситуації.

Значний розвиток отримали системи промислового моніторингу. Сучасні підприємства активно використовують мережі датчиків для контролю температури, тиску, вологості, рівня вібрацій та інших технологічних параметрів. Отримані дані дозволяють здійснювати постійний контроль стану обладнання та своєчасно виявляти потенційні несправності.

Особливого значення промисловий моніторинг набуває на об'єктах підвищеної небезпеки, де навіть незначні відхилення параметрів можуть призвести до серйозних наслідків. Саме тому сучасні системи повинні не лише відображати поточні значення показників, але й виконувати автоматичний аналіз інформації та формувати попередження про можливі загрози.

Таблиця 1.6 – Основні сфери застосування систем моніторингу IoT

Сфера застосування	Контрольовані параметри
Industry 4.0	Температура, тиск, вібрація, навантаження
Smart City	Якість повітря, транспорт, освітлення
Smart Home	Температура, вологість, безпека

Продовження таблиці 1.6

Енергетика	Споживання енергії, стан обладнання
Сільське господарство	Вологість ґрунту, температура, опади

Важливою тенденцією останніх років є використання хмарних IoT-платформ. Такі рішення забезпечують централізоване зберігання даних, віддалений доступ до інформації та масштабованість систем моніторингу. Серед найбільш відомих платформ можна виділити AWS IoT Core, Microsoft Azure IoT Hub та Google Cloud IoT.

Використання хмарних технологій дозволяє організувати збір та аналіз даних від великої кількості пристроїв незалежно від їхнього географічного розташування. Крім того, хмарні платформи надають інструменти для побудови аналітичних моделей, автоматизації процесів та інтеграції з іншими інформаційними системами.

Попри значний розвиток технологій, сучасні системи моніторингу стикаються з низкою проблем. Однією з основних є постійне зростання обсягів даних, що генеруються IoT-пристроями. Велика кількість інформації ускладнює її обробку та підвищує вимоги до обчислювальних ресурсів системи.

Іншою актуальною проблемою є забезпечення достовірності та цілісності даних. Помилки датчиків, збої мережевого обладнання або втрати пакетів під час передачі можуть призводити до появи некоректної інформації, що негативно впливає на результати аналізу.

Серйозною проблемою також залишається своєчасне виявлення аномалій. У багатьох випадках оператор фізично не може контролювати всі параметри системи одночасно, особливо якщо кількість датчиків становить сотні або тисячі одиниць. Це створює необхідність використання автоматизованих механізмів аналізу та формування сповіщень.

Крім того, важливим питанням є забезпечення безперервності роботи систем моніторингу в режимі реального часу. Навіть незначна затримка обробки інформації може призвести до втрати критично важливих даних або

несвоєчасного реагування на аварійну ситуацію.

Сучасний розвиток Інтернету речей демонструє тенденцію до переходу від простого збору інформації до створення інтелектуальних систем підтримки прийняття рішень. Такі системи повинні не лише відображати поточні показники, але й аналізувати їх, виявляти небезпечні тенденції та автоматично інформувати користувачів про відхилення від нормального режиму роботи.

Отже, аналіз сучасного стану розвитку систем моніторингу IoT-даних свідчить про постійне зростання ролі автоматизованих засобів контролю та обробки інформації. Використання концепцій Industry 4.0, Smart City та Smart Home підтверджує актуальність впровадження інтелектуальних систем моніторингу. Збільшення обсягів телеметричних даних та потреба в оперативному реагуванні на небезпечні ситуації обумовлюють необхідність створення систем, здатних автоматично виявляти аномалії, аналізувати потоки даних та забезпечувати підтримку прийняття рішень у режимі реального часу. Саме вирішенню цих завдань присвячено розробку інтелектуальної системи моніторингу та обробки IoT-даних, що розглядається в даній роботі.

2 РОЗРОБКА ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Загальна характеристика системи

На етапі розробки програмного забезпечення одним із найважливіших завдань є визначення функціональних та нефункціональних вимог до системи. Вимоги описують очікувану поведінку програмного продукту, його можливості, обмеження та умови експлуатації. Якісно сформовані вимоги забезпечують правильне проєктування архітектури системи та дозволяють створити програмний продукт, який повністю відповідає поставленим завданням.

Об'єктом розробки є програмна система моніторингу параметрів навколишнього середовища, що забезпечує отримання, обробку, візуалізацію та

аналіз даних від датчиків у режимі реального часу. Передача даних здійснюється за допомогою протоколу MQTT через брокер повідомлень EMQX. Для генерації та передачі даних використовується програмний модуль, реалізований мовою Python. Обробка отриманої інформації, виявлення аномальних значень та відображення результатів виконуються у середовищі Node-RED.

Основною метою створення системи є автоматизація процесу контролю параметрів навколишнього середовища, оперативне виявлення відхилень від нормальних значень та інформування користувача про потенційно небезпечні ситуації. Система повинна забезпечувати безперервний моніторинг даних та надавати користувачу зручний інтерфейс для аналізу поточного стану контрольованих показників.

У процесі аналізу предметної області було визначено основні категорії користувачів системи та їхні потреби. Основним користувачем є оператор системи моніторингу, який здійснює спостереження за показниками датчиків, аналізує отриману інформацію та реагує на повідомлення про аномальні ситуації.

Основні вимоги користувачів до програмного забезпечення наведено в таблиці 2.1.

Таблиця 2.1 – Вимоги до програмного забезпечення

№ п/п	Користувач	Вимоги користувача
1	Оператор системи	Отримання даних від датчиків у режимі реального часу
2	Оператор системи	Відображення поточних значень параметрів за допомогою індикаторів (Gauge)
3	Оператор системи	Побудова графіків зміни параметрів у часі
4	Оператор системи	Автоматичне виявлення аномальних значень
5	Оператор системи	Отримання текстових повідомлень про виникнення аномалій
6	Оператор системи	Отримання звукового сповіщення при виявленні критичних відхилень
7	Адміністратор системи	Забезпечення стабільного функціонування програмного забезпечення
8	Адміністратор системи	Можливість зміни параметрів моніторингу та налаштування системи

Під час розробки системи були сформульовані такі функціональні вимоги:

– прийом повідомлень від MQTT-брокера;

– обробка отриманих даних у середовищі Node-RED;

– фільтрація та аналіз інформації;

– виявлення аномальних значень параметрів;

– формування повідомлень про аномальні події;

– візуалізація даних за допомогою графіків та індикаторів;

– відтворення звукових сигналів тривоги;

– забезпечення взаємодії між програмними компонентами системи.

Крім функціональних вимог, було визначено низку нефункціональних вимог:

– висока швидкодія обробки повідомлень;

– надійність передачі даних через MQTT-протокол;

– зручність користувацького інтерфейсу;

– можливість масштабування системи;

– підтримка безперервної роботи протягом тривалого часу;

– простота налаштування та супроводу програмного забезпечення.

Для реалізації програмного забезпечення було обрано сучасний стек технологій, до якого входять мова програмування Python, платформа Node-RED, протокол MQTT та веб-інтерфейс Dashboard. Такий вибір забезпечує простоту інтеграції окремих компонентів системи, високу швидкість розробки та можливість подальшого розширення функціональних можливостей.

Отже, проведений аналіз вимог дозволив сформувати перелік функціональних та нефункціональних характеристик програмного забезпечення, які стали основою для подальшого проєктування архітектури системи моніторингу параметрів навколишнього середовища та реалізації її програмних компонентів.

2.1.1 Функціональні вимоги

Функціональні вимоги визначають основні можливості програмного забезпечення та описують дії, які система повинна виконувати для досягнення

поставленої мети. Для системи моніторингу параметрів навколишнього середовища функціональні вимоги пов’язані з отриманням даних, їх обробкою, аналізом та відображенням результатів користувачу.

Розроблювана система повинна забезпечувати автоматичний прийом даних від джерел інформації через протокол MQTT. Передача повідомлень здійснюється за допомогою MQTT-брокера EMQX, який забезпечує взаємодію між програмним модулем генерації даних та системою моніторингу.

Програмне забезпечення повинно виконувати обробку отриманих повідомлень у режимі реального часу. Для цього використовуються функціональні вузли Node-RED, які здійснюють аналіз структури повідомлень, виділення необхідних параметрів та їх подальше передавання до модулів візуалізації.

Однією з основних функцій системи є моніторинг контрольованих параметрів навколишнього середовища. Отримані значення повинні відображатися на інформаційній панелі за допомогою цифрових індикаторів та графіків, що дозволяє користувачу оперативно оцінювати поточний стан об’єкта спостереження.

Важливою функцією є автоматичне виявлення аномалій. Система повинна аналізувати вхідні дані та визначати випадки виходу параметрів за встановлені межі. У разі виявлення аномального значення користувач повинен отримати відповідне повідомлення.

Для оперативного реагування на небезпечні ситуації програмне забезпечення повинно підтримувати механізм сповіщення користувача. Передбачено формування текстових повідомлень на панелі керування та відтворення звукового сигналу тривоги.

Основні функціональні вимоги системи наведено нижче:

– отримання даних від MQTT-брокера;

– обробка повідомлень у середовищі Node-RED;

– аналіз параметрів навколишнього середовища;

– виявлення аномальних значень;

– формування повідомлень про аномалії;

– відображення даних у вигляді графіків;

– відображення поточних значень за допомогою індикаторів Gauge;

– звукове оповіщення користувача;

– забезпечення безперервного моніторингу даних;

– підтримка роботи в режимі реального часу.

Таким чином, визначені функціональні вимоги формують основу роботи програмного забезпечення та забезпечують виконання всіх необхідних задач моніторингу та контролю параметрів навколишнього середовища.

2.1.2 Нефункціональні вимоги

Нефункціональні вимоги визначають характеристики якості програмного забезпечення та встановлюють критерії ефективності його роботи. На відміну від функціональних вимог, вони описують не конкретні функції системи, а властивості, якими вона повинна володіти під час експлуатації.

Однією з найважливіших вимог є продуктивність системи. Програмне забезпечення повинно забезпечувати швидке приймання та обробку повідомлень без значних затримок. Час відображення нових даних на інформаційній панелі повинен бути мінімальним для забезпечення актуальності інформації.

Важливою характеристикою є надійність роботи системи. Програмне забезпечення повинно коректно функціонувати протягом тривалого часу без збоїв та втрати даних. У разі короткочасного розриву з’єднання з MQTT-брокером система повинна відновлювати роботу після повторного підключення.

Система повинна забезпечувати зручність використання для операторів різного рівня підготовки. Інтерфейс користувача має бути інтуїтивно зрозумілим та не вимагати спеціального навчання для роботи з основними функціями.

Окрему увагу необхідно приділити масштабованості програмного забезпечення. Архітектура системи повинна дозволяти додавання нових датчиків, інформаційних панелей та алгоритмів аналізу без суттєвої зміни існуючих компонентів.

До основних нефункціональних вимог належать:

– висока швидкодія обробки даних;

– стабільна робота у режимі реального часу;

– надійність передачі повідомлень;

– зручність користування інтерфейсом;

– можливість масштабування системи;

– простота налаштування та адміністрування;

– сумісність із сучасними веббраузерами;

– можливість безперервної роботи протягом тривалого часу;

– підтримка подальшого розвитку та модернізації системи.

Зазначені нефункціональні вимоги забезпечують ефективне використання програмного забезпечення та підвищують якість функціонування системи моніторингу в цілому.

2.1.3 Вимоги до інтерфейсу користувача

Інтерфейс користувача є одним із ключових компонентів програмної системи, оскільки забезпечує взаємодію оператора з функціональними можливостями програмного забезпечення. Основною метою проєктування інтерфейсу є забезпечення швидкого доступу до інформації про поточний стан контрольованих параметрів та оперативного реагування на аномальні ситуації.

Інтерфейс розроблюваної системи реалізовано за допомогою Dashboard-панелі середовища Node-RED. Вебінтерфейс забезпечує відображення даних у режимі реального часу та доступний через стандартний веббраузер.

На головній сторінці повинні розміщуватися елементи візуалізації, які дозволяють користувачу отримувати актуальну інформацію про контрольовані параметри. Для цього використовуються графічні індикатори типу Gauge та часові графіки типу Chart.

Інтерфейс користувача повинен забезпечувати:

– відображення поточних значень усіх контрольованих параметрів;

– графічне представлення зміни параметрів у часі;

– автоматичне оновлення даних без перезавантаження сторінки;

– відображення повідомлень про аномальні події;

– звукове сповіщення про критичні відхилення;

– зручне розташування елементів управління та візуалізації;

– можливість перегляду інформації з різних пристроїв.

Для покращення сприйняття інформації інтерфейс повинен використовувати логічне групування елементів. Показники датчиків доцільно розміщувати у вигляді окремих інформаційних блоків, а повідомлення про аномалії — у спеціально виділеній області екрана.

Важливою вимогою є адаптивність інтерфейсу. Dashboard повинен коректно відображатися як на персональних комп’ютерах, так і на ноутбуках, планшетах та мобільних пристроях. Це дозволить оператору контролювати стан системи незалежно від місця перебування.

Отже, вимоги до інтерфейсу користувача спрямовані на забезпечення максимальної наочності, зручності та оперативності роботи із системою моніторингу параметрів навколишнього середовища.

2.2 Розробка специфікації вимог SRS

Специфікація вимог до програмного забезпечення (Software Requirements Specification, SRS) є одним із основних документів під час розробки програмних систем. Вона визначає функціональні можливості програмного продукту, його характеристики, обмеження та умови експлуатації. Наявність SRS дозволяє формалізувати вимоги до системи, забезпечити однакове розуміння поставлених завдань усіма учасниками процесу розробки та мінімізувати ризики виникнення помилок під час реалізації програмного забезпечення.

Для розроблюваної інтелектуальної системи моніторингу та обробки IoT-даних специфікація вимог визначає перелік функцій, які повинна виконувати система, вимоги до продуктивності, надійності та зручності використання. Документ також описує взаємодію між окремими компонентами системи, зокрема

джерелами даних, MQTT-брокером, середовищем Node-RED та інформаційною панеллю користувача.

Метою розробки специфікації вимог є формування чіткого опису функціональних можливостей системи ще до початку етапу реалізації. Це дозволяє визначити межі проєкту, оцінити необхідні ресурси та забезпечити контроль відповідності кінцевого результату поставленим завданням.

Структура специфікації вимог для розроблюваної системи включає три основні розділи: вступ, загальний опис та специфічні вимоги. У вступі визначається призначення документа, область застосування системи та перелік основних термінів і скорочень. Розділ загального опису містить інформацію про архітектуру системи, її взаємодію із зовнішніми компонентами та характеристиками потенційних користувачів. Специфічні вимоги деталізують функціональність програмного забезпечення та визначають критерії його якості.

Призначенням розроблюваної системи є забезпечення збору, обробки та візуалізації даних, отриманих від IoT-пристроїв. Система повинна підтримувати прийом повідомлень через протокол MQTT, виконувати аналіз отриманої інформації, відображати показники в режимі реального часу та автоматично виявляти аномальні значення контрольованих параметрів.

До основних функцій системи належать прийом телеметричних даних, обробка інформації, збереження поточних значень параметрів, формування графіків та індикаторів, а також генерація повідомлень про виявлені аномалії. Крім того, система повинна забезпечувати зручний доступ до інформації через веб-інтерфейс, створений засобами Node-RED Dashboard.

Під час формування специфікації вимог також враховуються нефункціональні характеристики програмного забезпечення. Зокрема, система повинна забезпечувати високу швидкість обробки повідомлень, стабільність роботи при тривалому функціонуванні, можливість масштабування та простоту супроводу. Важливими є також вимоги до зручності користування, оскільки результати моніторингу повинні бути представлені у зрозумілому та наочному вигляді.

Таким чином, специфікація вимог SRS виступає основою для подальшого моделювання, проєктування та реалізації інтелектуальної системи моніторингу IoT-даних. Вона забезпечує формалізацію вимог до програмного забезпечення та створює необхідні умови для розробки системи, яка відповідатиме поставленим функціональним і технічним завданням.

2.3 Структура інтелектуальної системи

У цьому підрозділі описується загальна структура інтелектуальної системи моніторингу та обробки IoT-даних, яка реалізує збір, передачу, обробку та візуалізацію інформації в режимі реального часу. Архітектура системи побудована за модульним принципом, що дозволяє розділити функціональні блоки та забезпечити простоту масштабування і супроводу.

Розроблювана система складається з кількох основних компонентів, які взаємодіють між собою через мережеві протоколи передачі даних. Джерелом інформації виступає Python-скрипт, який імітує роботу IoT-датчиків та генерує телеметричні дані. Згенеровані значення передаються у вигляді повідомлень до MQTT-брокера broker.emqx.io, який забезпечує централізовану передачу даних між учасниками системи.

На стороні серверної обробки використовується платформа Node-RED, яка підключається до MQTT-брокера через вузол MQTT In. Отримані повідомлення надходять у функціональний блок SensorData, де здійснюється первинна обробка та структуризація даних. На цьому етапі дані розділяються за відповідними параметрами та готуються для подальшої обробки.

Оброблені дані розподіляються на кілька паралельних потоків, кожен з яких відповідає за окремий параметр системи: temperature, humidity, pressure, wind, rain та air quality. Кожен потік проходить через вузли Change, після чого дані передаються до компонентів візуалізації — Gauge та Chart, які відображають

поточні значення та їх зміну в часі.

Окремим функціональним блоком системи є підсистема виявлення аномалій. Вона реалізована у вигляді функції, яка аналізує вхідні дані та порівнює їх із заданими пороговими значеннями. У випадку виявлення відхилень формується спеціальне повідомлення, яке передається до підсистеми сповіщень.

Підсистема сповіщень включає два основних елементи: Notification, яка відповідає за текстові повідомлення про аномалії, та Alarm, яка забезпечує звукове сповіщення користувача. Це дозволяє оперативно реагувати на критичні зміни параметрів системи.

На рівні користувацького інтерфейсу всі дані відображаються у Node-RED Dashboard, який забезпечує інтерактивну інформаційну панель з графіками, індикаторами та повідомленнями про стан системи.
Таким чином, запропонована структура системи забезпечує повний цикл роботи з IoT-даними: від їх генерації та передачі до аналізу, візуалізації та реагування на аномальні ситуації.

2.4 Функціонування системи

Принципи функціонування інтелектуальної системи моніторингу IoT-даних базуються на концепції безперервної обробки потокової інформації та поділу системи на незалежні етапи: генерація даних, передача, прийом, обробка, аналіз та візуалізація. Такий підхід забезпечує високу гнучкість, масштабованість та можливість роботи системи в режимі реального часу.

Основним принципом роботи системи є подієва модель обробки даних (event-driven architecture). Це означає, що всі дії в системі ініціюються надходженням нових повідомлень від IoT-пристроїв. Кожне повідомлення розглядається як окрема подія, яка проходить послідовність обробки без необхідності періодичного опитування джерел даних.

Передача інформації між компонентами системи здійснюється за допомогою протоколу MQTT, який реалізує принцип публікації та підписки (publish/subscribe). IoT-пристрої виступають у ролі публікаторів даних, а Node-RED-система — у ролі підписника, який отримує повідомлення через MQTT-брокер. Такий підхід дозволяє зменшити зв’язність компонентів системи та спростити її масштабування.

Обробка даних у системі виконується потоково. Це означає, що кожне нове повідомлення проходить через послідовність обробних блоків одразу після його отримання. Дані не накопичуються для пакетної обробки, що дозволяє забезпечити мінімальну затримку між вимірюванням та відображенням результату.

Принцип розподілу даних за параметрами дозволяє обробляти кожен показник окремо. Такий підхід забезпечує незалежність потоків даних для температури, вологості, тиску, швидкості вітру, рівня опадів та якості повітря. Завдяки цьому система може легко масштабуватися шляхом додавання нових каналів без зміни основної логіки.

Окремим принципом роботи є безперервна візуалізація даних у режимі реального часу. Інформаційна панель оновлюється автоматично при надходженні нових значень, що дозволяє користувачеві спостерігати актуальний стан системи без необхідності ручного оновлення сторінки або запитів до сервера.

Принцип виявлення аномалій базується на пороговому аналізі значень. Для кожного параметра визначено допустимий діапазон. У випадку виходу значення за межі цього діапазону система формує подію аномалії, яка передається до підсистеми сповіщень. Такий механізм забезпечує швидке реагування на критичні зміни показників.

Окремо слід виділити принцип незалежності модулів системи. Кожен функціональний блок (прийом даних, обробка, візуалізація, аналіз аномалій) працює автономно та взаємодіє з іншими через стандартизовані повідомлення MQTT. Це дозволяє змінювати або модернізувати окремі частини системи без впливу на загальну архітектуру.

Таким чином, принципи функціонування системи забезпечують стабільну роботу інтелектуальної платформи моніторингу IoT-даних, її масштабованість, гнучкість та здатність до обробки потокової інформації в режимі реального часу.

3 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

На етапі моделювання програмного забезпечення виконується формалізація функціональних вимог та опис майбутньої системи за допомогою спеціалізованих моделей. Моделювання дозволяє визначити основні компоненти системи, їх взаємодію та послідовність виконання операцій. Використання UML-діаграм сприяє кращому розумінню структури програмного забезпечення та спрощує процес його подальшої реалізації.

Для розроблюваної системи моніторингу параметрів навколишнього середовища було використано об'єктно-орієнтований підхід до проєктування. У процесі моделювання визначено основних учасників системи, варіанти використання, сценарії взаємодії між компонентами та послідовність обробки даних.

Система включає декілька взаємопов'язаних компонентів: програмний модуль генерації даних на мові Python, MQTT-брокер для передачі повідомлень, середовище Node-RED для обробки інформації та вебінтерфейс Dashboard для відображення результатів моніторингу. Кожен із зазначених компонентів виконує власні функції та бере участь у загальному процесі збору, аналізу та візуалізації даних.

У результаті моделювання були побудовані UML-діаграми, які відображають функціональні можливості системи та взаємодію між її компонентами. Побудовані моделі використовуються як основа для подальшого проєктування архітектури програмного забезпечення та реалізації його програмних модулів.

3.1 Діаграма варіантів використання

Діаграма варіантів використання (Use Case Diagram) дозволяє визначити взаємодію користувача із системою та відобразити основні функції програмного забезпечення. Основним актором розроблюваної системи є оператор моніторингу, який здійснює контроль параметрів навколишнього середовища та аналізує результати роботи системи.

До основних варіантів використання належать:

– перегляд поточних значень параметрів;

– перегляд графіків зміни показників;

– отримання повідомлень про аномалії;

– отримання звукових сповіщень;

– аналіз історії змін параметрів;

– контроль стану системи моніторингу.

На рисунку 3.1 наведено діаграму варіантів використання системи.

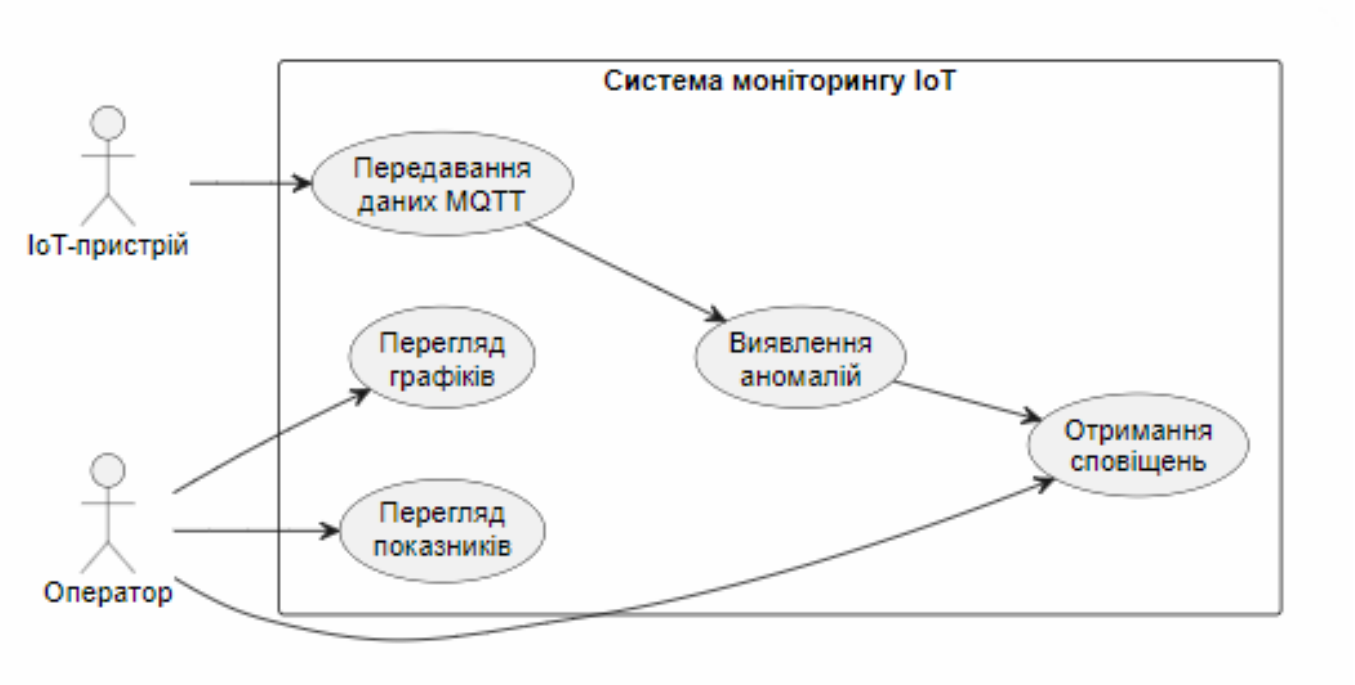

Рисунок 3.1 – Діаграма варіантів використання системи моніторингу параметрів навколишнього середовища

3.2 Діаграма послідовності

Для відображення процесу обробки даних у системі було побудовано діаграму послідовності (Sequence Diagram). Вона демонструє порядок взаємодії між компонентами програмного забезпечення під час передачі та обробки інформації.

Послідовність роботи системи складається з таких етапів: Python-модуль генерує значення параметрів середовища; дані передаються до MQTT-брокера EMQX; вузол MQTT In у Node-RED отримує повідомлення; функція SensorData виконує обробку отриманих даних. Далі оброблені значення передаються до компонентів візуалізації Dashboard; дані одночасно надходять до модуля визначення аномалій, у разі виявлення відхилень формується повідомлення про аномалію, система активує звукове сповіщення та відображає повідомлення користувачу.

Побудована діаграма дозволяє наочно продемонструвати взаємодію між програмними компонентами та послідовність виконання основних операцій.

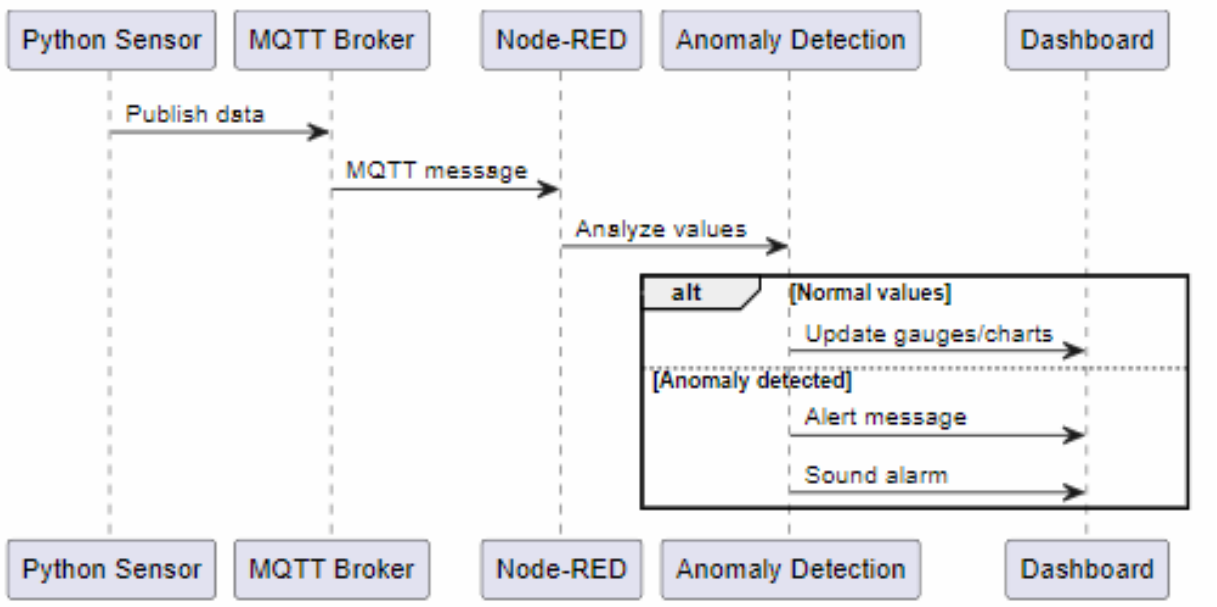

Рисунок 3.2 – Діаграма послідовності для використання системи моніторингу параметрів навколишнього середовища

3.3 Розгорнутий опис прецедентів

У даному підрозділі наведено детальний опис основних прецедентів використання розробленої системи моніторингу та аналізу даних IoT-пристроїв. Опис прецедентів дозволяє визначити послідовність взаємодії користувача із системою та реакцію системи на різні події.

3.3.1 Прецедент «Перегляд показників датчиків»

Мета: отримання актуальних значень параметрів датчиків у режимі реального часу.

Основний актор: оператор системи.

Передумови:

- система запущена;
- встановлено з'єднання з MQTT-брокером;
- датчики передають дані.

Основний сценарій:

- оператор відкриває веб-інтерфейс системи;
- система підключається до потоку даних MQTT;
- дані надходять до Node-RED;
- функція обробки SensorData аналізує отримані повідомлення;
- значення параметрів відображаються на індикаторах Gauge;
- історія змін відображається на графіках Chart.

Результат: оператор отримує актуальну інформацію про стан контрольованого об'єкта.

3.3.2 Прецедент «Виявлення аномалії»

Мета: автоматичне визначення виходу параметрів за допустимі межі.

Основний актор: система моніторингу.

Передумови:

- отримуються дані від датчиків;
- встановлені порогові значення параметрів.

Основний сценарій:

- дані надходять до функціонального блоку SensorData;
- оброблені дані передаються до функції Anomaly Detection;
- система порівнює отримані значення з допустимими межами;
- при виявленні відхилення формується повідомлення про аномалію;
- дані про аномалію передаються до модуля сповіщень.

Альтернативний сценарій:

- значення параметрів знаходяться в межах норми;
- аномалія не фіксується;
- система продовжує моніторинг.

Результат: аномальна подія виявлена та зафіксована системою.

3.3.3 Прецедент «Отримання повідомлення про аномалію»

Мета: інформування оператора про виникнення аварійної або нестандартної ситуації.

Основний актор: оператор системи.

Передумови:

- виявлена аномалія;
- система моніторингу працює коректно.

Основний сценарій:

- модуль Anomaly Detection формує повідомлення;
- повідомлення передається до вузла Notification;
- у веб-інтерфейсі відображається текстове попередження;
- активується звуковий сигнал Alarm;
- оператор аналізує отриману інформацію.

Результат: оператор своєчасно отримує повідомлення про проблему.

3.3.4 Прецедент «Передавання даних через MQTT»

Мета: доставка телеметричних даних від пристрою до системи моніторингу.

Основний актор: IoT-пристрій.

Передумови:

- MQTT-брокер доступний;
- пристрій підключений до мережі.

Основний сценарій:

- python-застосунок формує пакет даних;
- дані публікуються у MQTT Topic;
- MQTT Broker приймає повідомлення;
- Node-RED отримує повідомлення через MQTT In;
- дані передаються на подальшу обробку.

Результат: дані успішно доставлені до системи моніторингу.

3.4 Сценарії використання системи

Сценарії використання описують послідовність дій користувача та реакцію системи на ці дії. Для розробленої системи були визначені основний та альтернативний сценарії роботи.

Основний сценарій:

- система запускається та встановлює з'єднання з MQTT-брокером;
- python-модуль починає надсилати дані;
- Node-RED отримує повідомлення та виконує їх обробку;
- дані відображаються на Dashboard у вигляді графіків та індикаторів;
- користувач переглядає поточний стан параметрів;
- система продовжує моніторинг у режимі реального часу.

Альтернативний сценарій:

- система отримує значення, що перевищує допустимий поріг;
- модуль аналізу визначає аномалію;
- формується повідомлення про небезпечну подію;
- на інформаційній панелі відображається попередження.

– активується звуковий сигнал тривоги;

– користувач отримує інформацію про виникнення аномальної ситуації.

Невдалий сценарій:

– втрачається з'єднання з MQTT-брокером;

– передача даних тимчасово припиняється;

– система фіксує відсутність нових повідомлень;

– після відновлення з'єднання прийом даних продовжується у штатному режимі.

Таким чином, виконане моделювання дозволило визначити основні процеси функціонування програмного забезпечення, встановити взаємозв'язки між його компонентами та підготувати основу для подальшого проєктування архітектури системи моніторингу параметрів навколишнього середовища.

4 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

На етапі проєктування архітектури програмного забезпечення було визначено структуру взаємодії компонентів системи моніторингу параметрів IoT-пристроїв та виявлення аномалій. Архітектура системи побудована за модульним принципом, що забезпечує простоту масштабування, підтримки та подальшого розвитку програмного забезпечення.

Основною задачею архітектурного проєктування стало забезпечення безперервного збору телеметричних даних, їх обробки в режимі реального часу, візуалізації показників та своєчасного інформування користувача про виникнення аномальних ситуацій.

Під час проєктування враховувалися вимоги до продуктивності, надійності передачі повідомлень, зручності користування системою та можливості інтеграції з іншими сервісами моніторингу. Для передачі даних було використано протокол MQTT, який є одним із найпоширеніших рішень у сфері Інтернету речей. Обробка

даних реалізована засобами середовища Node-RED, що дозволяє швидко будувати потоки даних та інтегрувати різні програмні компоненти без значних витрат часу на розробку.

4.1 Загальна архітектура системи

Архітектура розробленої системи складається з чотирьох основних рівнів:

- рівень генерації даних;
- рівень передачі повідомлень;
- рівень обробки та аналізу;
- рівень відображення інформації.

На першому рівні працює програмний модуль, розроблений мовою Python. Його основним завданням є формування та передавання телеметричних даних, що імітують роботу набору датчиків. Передавання інформації здійснюється через MQTT-протокол.

Другий рівень представлений MQTT-брокером broker.emqx.io, який виконує функції прийому, зберігання та маршрутизації повідомлень між джерелами даних і системою моніторингу.

На третьому рівні знаходиться середовище Node-RED, яке реалізує основну бізнес-логіку системи. Саме тут відбувається прийом повідомлень, їх обробка, аналіз параметрів та визначення аномальних значень.

Четвертий рівень представлений інформаційною панеллю Dashboard, де оператор може спостерігати поточні значення параметрів, переглядати графіки зміни показників та отримувати повідомлення про аномальні ситуації.

Запропонована архітектура забезпечує високу гнучкість системи та дозволяє додавати нові типи датчиків або алгоритми аналізу без суттєвої зміни існуючих компонентів.

На рисунку 4.1 наведено діаграму розміщення компонентів системи моніторингу IoT-пристроїв.

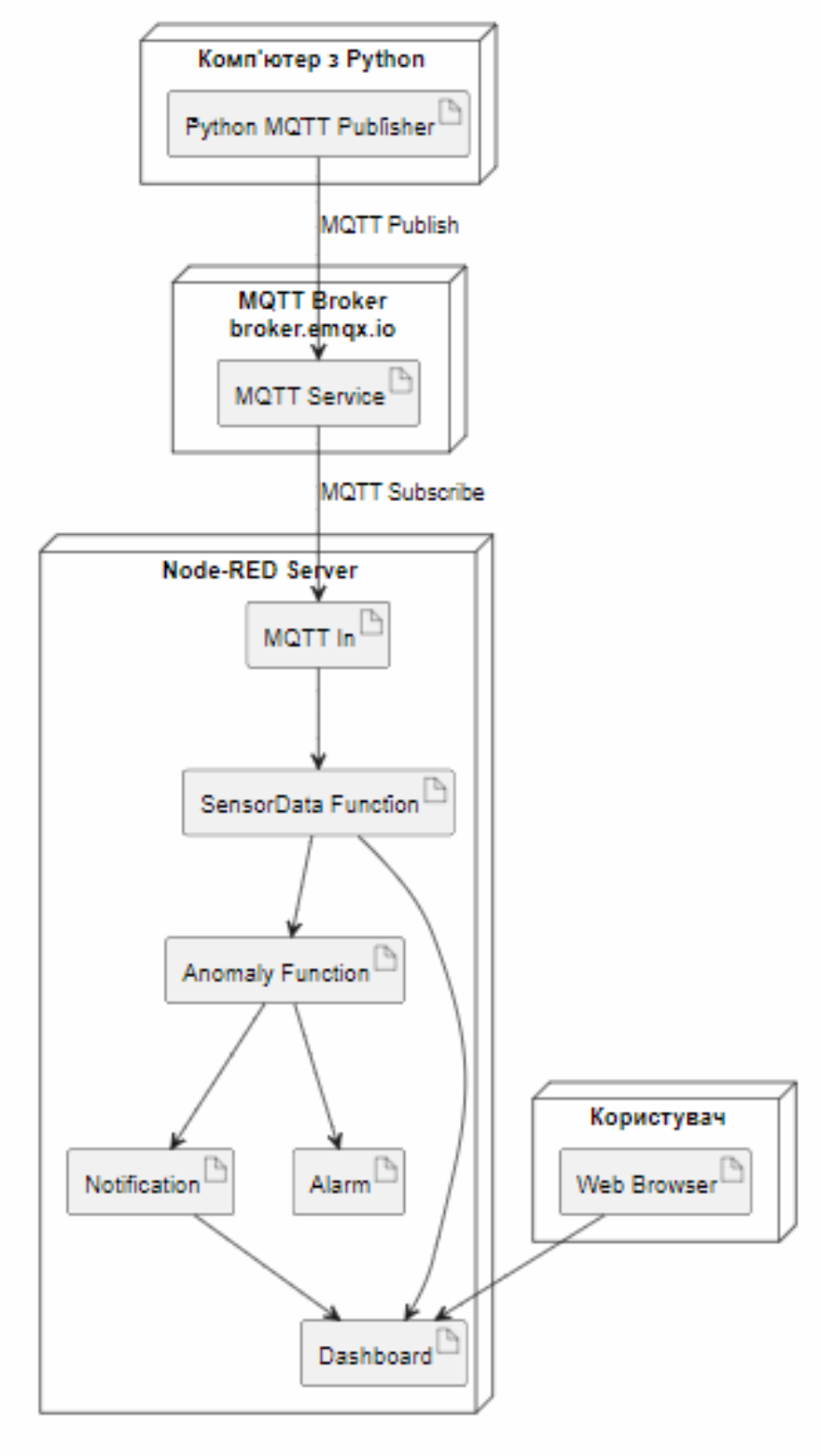

Рисунок 4.1 – Діаграма розміщення компонентів системи моніторингу IoT-пристроїв

4.2 Апаратне забезпечення системи

Незважаючи на те, що основна увага в роботі приділяється програмній реалізації інтелектуальної системи моніторингу IoT-даних, важливим елементом будь-якої IoT-системи є апаратне забезпечення, яке забезпечує збір первинної інформації про стан навколишнього середовища або об’єкта спостереження.

У рамках розробленої системи передбачено використання мікроконтролера ESP32 як пристрою збору та передавання даних. Мікроконтролер ESP32 має вбудований модуль Wi-Fi та забезпечує підтримку сучасних мережевих протоколів, що дозволяє використовувати його як універсальну платформу для побудови IoT-рішень.

Рисунок 4.2 – Мікроконтролер ESP32

Для контролю параметрів навколишнього середовища можуть використовуватись різні типи сенсорів. У даній роботі під час моделювання даних враховувались характеристики датчиків температури, вологості, атмосферного тиску та якості повітря. Через відсутність фізичного обладнання на етапі розробки було використано програмне моделювання показників датчиків із генерацією значень, близьких до реальних умов експлуатації.

Для контролю температури та вологості повітря може використовуватись цифровий датчик DHT22. Даний сенсор забезпечує вимірювання температури в діапазоні від –40 °C до +80 °C та вологості від 0 до 100 %. Типова похибка вимірювання температури становить ±0,5 °C, а вологості — ±2–5 %.

На рисунку 4.2.1 наведено схему датчика DHT22, на рисунку 4.2.2 – модуль датчика DHT22.

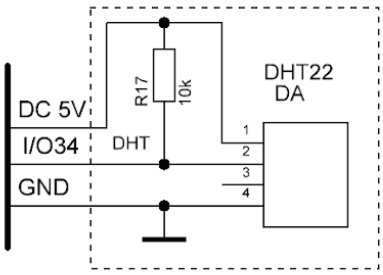

Рисунок 4.2.1 – Схема датчика DHT22

Рисунок 4.2.2 – Модуль датчика DHT22

Для контролю атмосферного тиску може використовуватись датчик BMP280. Сенсор дозволяє визначати атмосферний тиск у широкому діапазоні та може застосовуватись для моніторингу погодних умов і розрахунку висоти над рівнем моря.

На рисунку 4.2.3 наведено модуль датчика BMP280.

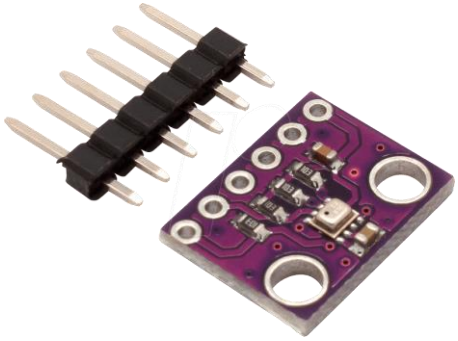

Рисунок 4.2.3 – Модуль датчика BMP280

Для оцінки якості повітря можуть застосовуватись газові сенсори серії MQ, які здатні визначати концентрацію різних газів та рівень забруднення повітря.

На рисунку 4.2.4 наведено газові сенсори серії MQ.

Рисунок 4.2.4 – Газовий сенсор MQ-135

У таблиці 4.1 наведено основні характеристики датчиків.

Таблиця 4.1 – Основні характеристики датчиків

Датчик	Параметр	Діапазон
DHT22	Температура	-40…+80 °C
DHT22	Вологість	0…100%
BMP280	Тиск	300…1100hPa
MQ-135	Якість повітря	AQI

Моделювання показників зазначених датчиків дозволило відтворити умови функціонування реальної IoT-системи без використання фізичного обладнання. Згенеровані дані передавались до системи моніторингу через MQTT-брокер та використовувались для тестування алгоритмів обробки інформації, візуалізації даних і виявлення аномалій.

Запропонований підхід забезпечує можливість подальшого переходу від програмного моделювання до використання реальних IoT-пристроїв без необхідності суттєвої зміни архітектури системи.

4.3 Організація потоків обробки даних

Обробка даних у системі реалізована у вигляді послідовного потоку повідомлень.

Після запуску Python-додатка генеруються значення контрольованих параметрів та публікуються у відповідний MQTT Topic. MQTT-брокер приймає повідомлення та передає їх до вузла MQTT In середовища Node-RED.

Отримані повідомлення надходять до функціонального вузла SensorData, який виконує первинну обробку інформації. На цьому етапі відбувається розбір структури повідомлення та підготовка даних для подальшого використання.

Далі сформований потік розгалужується на декілька незалежних напрямків. Кожен параметр надходить до окремого вузла Change, після чого відображається за допомогою елементів Gauge та Chart на інформаційній панелі.

Паралельно всі отримані дані передаються до підсистеми аналізу аномалій, яка контролює відповідність значень встановленим допустимим межам.

Такий підхід дозволяє виконувати візуалізацію та аналіз інформації одночасно, що забезпечує оперативне реагування на зміну стану контрольованого об'єкта.

На рисунку 4.3 наведено схему потоків обробки даних у Node-RED.

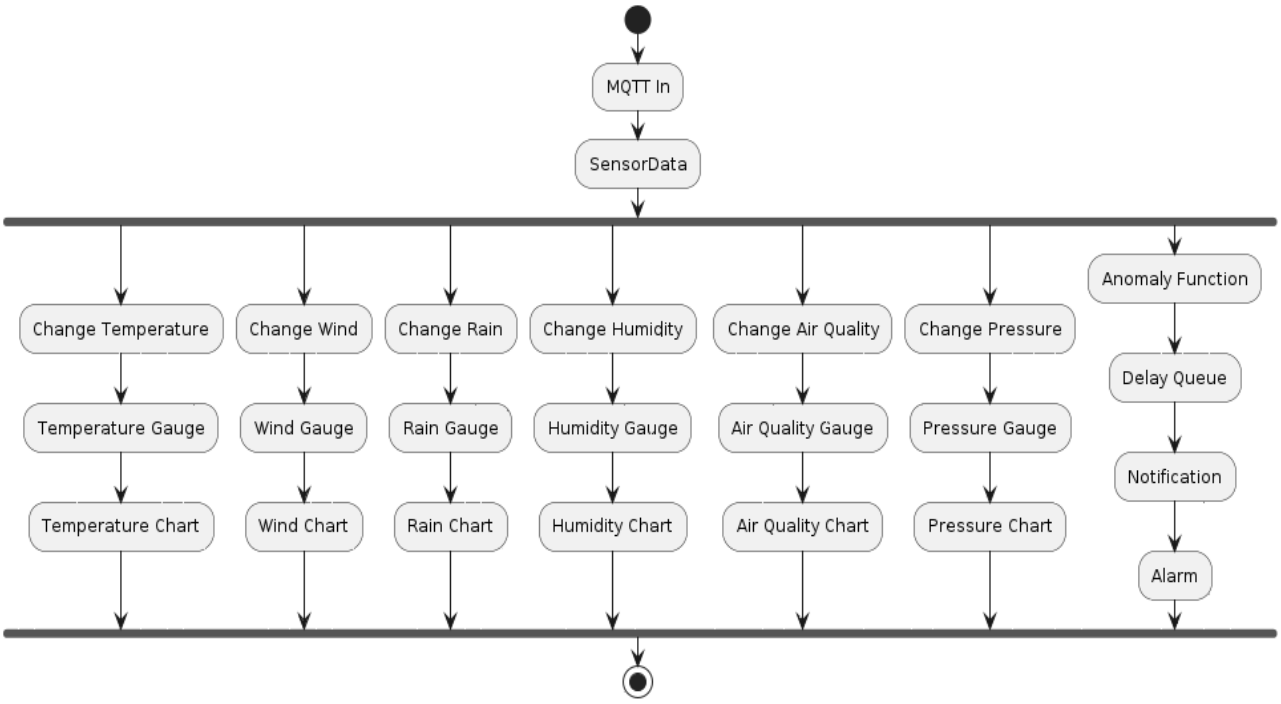

Рисунок 4.3 – Схема потоків обробки даних у Node-RED

4.4 Механізм виявлення аномалій

Одним із ключових компонентів системи є підсистема автоматичного виявлення аномалій.

Для реалізації даного механізму використовується окремий функціональний вузол Node-RED під назвою Function Anomaly. Його завданням є аналіз отриманих значень та порівняння їх із заздалегідь встановленими пороговими межами.

Під час надходження нового повідомлення система перевіряє всі контрольовані параметри. Якщо хоча б один із них перевищує допустиме значення або виходить за встановлений діапазон, формується повідомлення про аномальну подію.

Після виявлення аномалії інформація передається до вузла Delay Queue, який забезпечує коректну обробку повідомлень та запобігає надмірному дублюванню сповіщень.

Використання такого підходу дозволяє своєчасно виявляти потенційно небезпечні ситуації та знижувати ризик втрати важливої інформації.

На рисунку 4.4 наведено схему алгоритму виявлення аномалій у Node-RED.

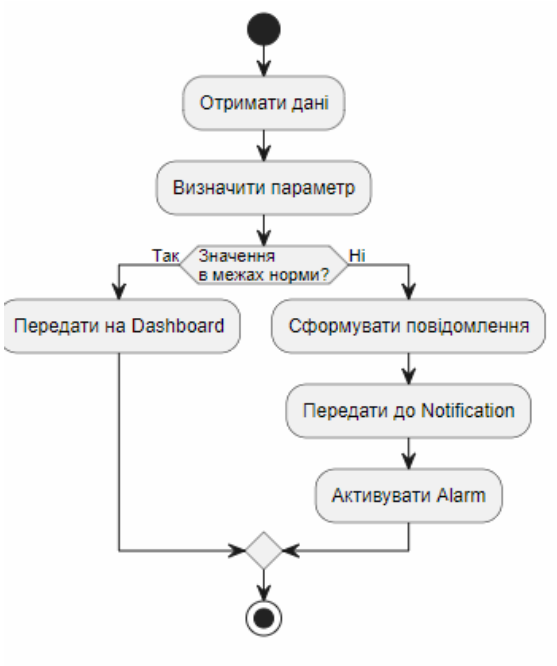

Рисунок 4.4 – Схема алгоритму виявлення аномалій у Node-RED

4.5 Підсистема сповіщень

Підсистема сповіщень призначена для оперативного інформування користувача про виявлені аномалії.

Після отримання повідомлення від модуля аналізу аномалій система формує текстове попередження та передає його до вузла Notification. Сформоване повідомлення відображається на інформаційній панелі Dashboard.

Для підвищення ефективності реагування додатково використовується звукове оповіщення Alarm. При виникненні аномальної ситуації оператор отримує не лише текстове повідомлення, але й звуковий сигнал, що привертає увагу навіть за відсутності постійного візуального контролю панелі моніторингу.

Реалізована підсистема сповіщень забезпечує швидке інформування користувача та підвищує надійність роботи всієї системи моніторингу.

На рисунку 4.5 наведено схему формування сповіщень у Node-RED.

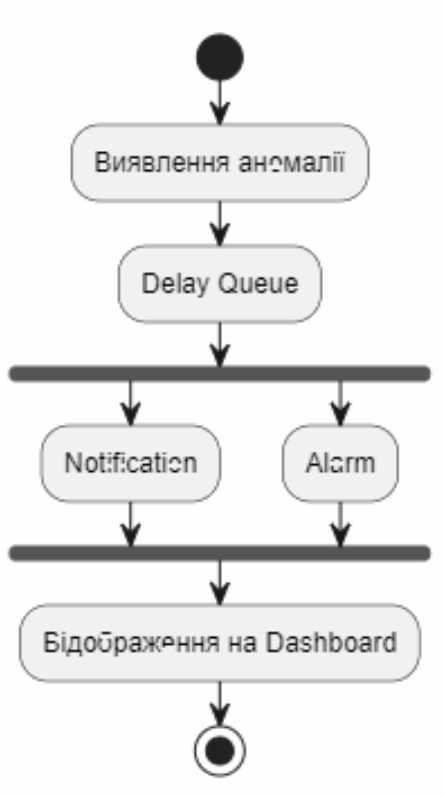

Рисунок 4.5 – Схема формування сповіщень у Node-RED

5 ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

Інтерфейс користувача є важливою складовою інформаційної системи моніторингу параметрів навколишнього середовища, оскільки саме через нього здійснюється взаємодія користувача із системою, перегляд поточних показників

та отримання інформації про можливі відхилення від нормальних умов функціонування. При проєктуванні інтерфейсу основна увага приділялася забезпеченню наочності відображення даних, швидкості сприйняття інформації та зручності використання системи в режимі реального часу.

Для реалізації користувацького інтерфейсу було використано Dashboard-панель середовища Node-RED. Дане рішення дозволяє створювати інтерактивні веб-інтерфейси без необхідності розробки окремого клієнтського застосунку. Усі елементи відображення даних автоматично оновлюються під час надходження нових повідомлень через MQTT-брокер, що забезпечує актуальність представленої інформації.

Проєктування інтерфейсу виконувалося відповідно до принципів простоти, функціональності та мінімізації навантаження на користувача. Основні параметри моніторингу розміщені у вигляді окремих інформаційних блоків, що забезпечує швидкий доступ до необхідних даних та спрощує аналіз поточного стану контрольованого середовища.

5.1 Структура інформаційної панелі

Інформаційна панель системи являє собою єдиний веб-інтерфейс, у межах якого розташовані засоби відображення даних від усіх підключених датчиків. Структура панелі організована таким чином, щоб користувач міг одночасно контролювати всі ключові параметри середовища.

До складу інформаційної панелі входять наступні блоки:

- блок відображення температури повітря (рис.5.1.1);
- блок відображення швидкості вітру (рис.5.1.2);
- блок відображення рівня опадів (рис.5.1.3);
- блок відображення вологості повітря (рис.5.1.4);
- блок відображення якості повітря (рис.5.1.5);

– блок відображення атмосферного тиску (рис.5.1.6);

– блок повідомлень про аномальні події (рис.5.1.7);

– блок звукового оповіщення (рис.5.1.8).

Кожний блок містить цифровий індикатор поточного значення та графічне представлення зміни параметра в часі. Такий підхід дозволяє одночасно аналізувати як поточний стан середовища, так і динаміку його зміни.

Для забезпечення зручності користування всі елементи панелі мають однаковий стиль оформлення та логічне розташування на екрані. Найважливіша інформація розміщується у верхній частині інтерфейсу, що дозволяє оперативно оцінити ситуацію без необхідності перегляду всієї сторінки.

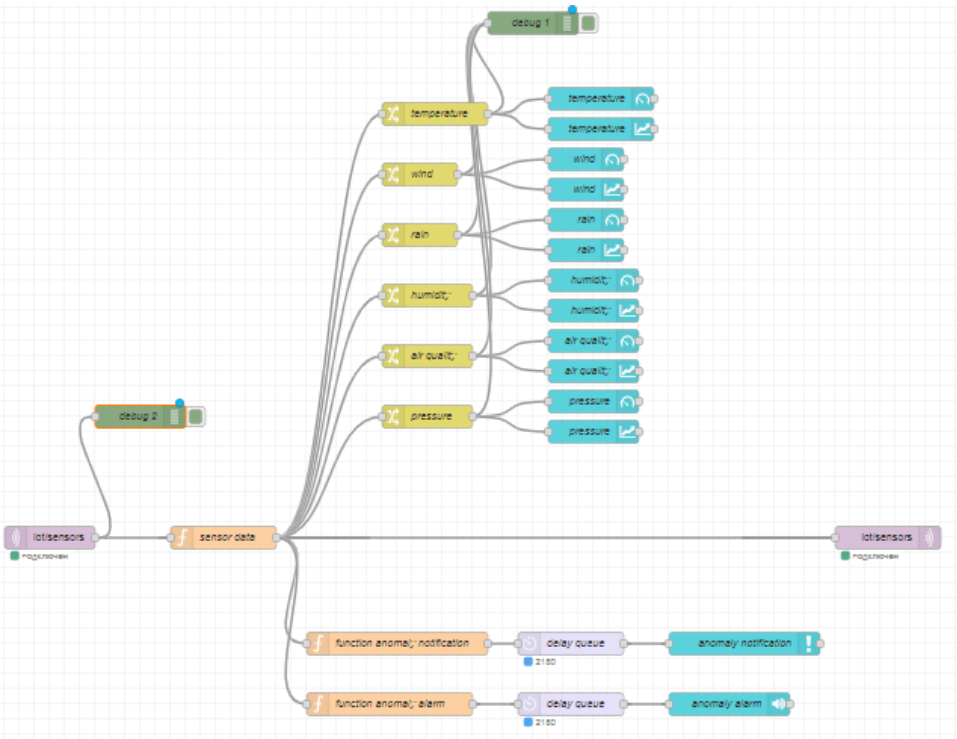

Рисунок 5.1 – Загальна структура інформаційної панелі системи моніторингу

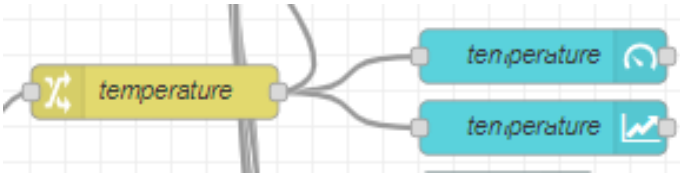

Рисунок 5.1.1 – Блок відображення температури повітря

Рисунок 5.1.2 – Блок відображення швидкості вітру

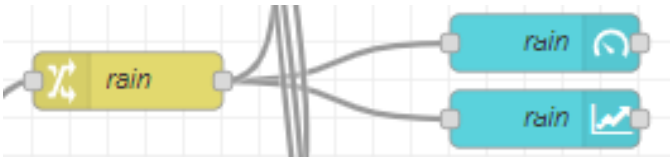

Рисунок 5.1.3 – Блок відображення рівня опадів

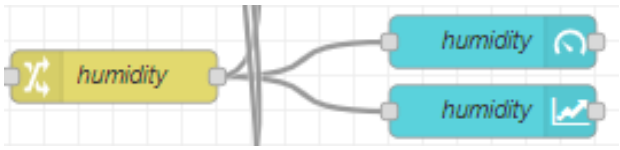

Рисунок 5.1.4 – Блок відображення вологості повітря

Рисунок 5.1.5 – Блок відображення якості повітря

Рисунок 5.1.6 – Блок відображення атмосферного тиску

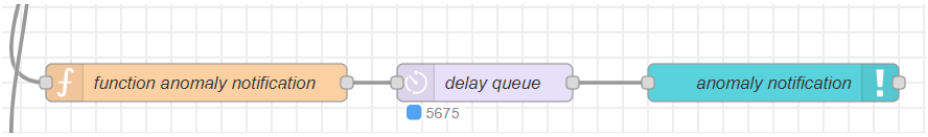

Рисунок 5.1.7 – Блок повідомлень про аномальні події

Рисунок 5.1.8 – Блок звукового оповіщення

5.2 Засоби візуалізації даних

Одним із головних завдань інтерфейсу є забезпечення наочного відображення отриманих даних. Для цього в системі використовуються графічні елементи Dashboard-панелі Node-RED.

Поточні значення параметрів відображаються за допомогою компонентів типу Gauge. Дані елементи дозволяють швидко визначити стан контрольованого показника та оцінити його наближення до критичних значень. Для кожного параметра створено окремий індикатор.

Аналіз змін параметрів у часі реалізовано за допомогою компонентів Chart. Графіки автоматично оновлюються при надходженні нових повідомлень та відображають історію вимірювань. Це дозволяє виявляти тенденції зміни показників та аналізувати поведінку системи протягом певного проміжку часу.

У системі реалізовано візуалізацію наступних параметрів:

- температура повітря;
- відносна вологість;
- атмосферний тиск;
- швидкість вітру;
- рівень опадів;
- показник якості повітря.

Застосування графічних інструментів дозволяє значно підвищити інформативність інтерфейсу порівняно з текстовим відображенням даних.

Блоком інформаційної панелі для відображення показників датчиків візуально обрано елементи Node-RED – Chart.

Блок складається з 6 графіків (Chart) що відображаються на панелі користувача (Dashboard).

На рисунку 5.2 наведено початковий стан графіків до моменту отримання даних.

Рисунок 5.2 – Блок інформаційної панелі для відображення показників датчиків за допомогою компонентів Chart

Блоком інформаційної панелі для відображення показників датчиків графічно обрано елементи Node-RED – Gauge.

Блок складається з 6 панелей значень (Gauge) що відображаються на панелі користувача (Dashboard).

На рисунку 5.2.1 наведено початковий стан панелей значень до моменту отримання даних.

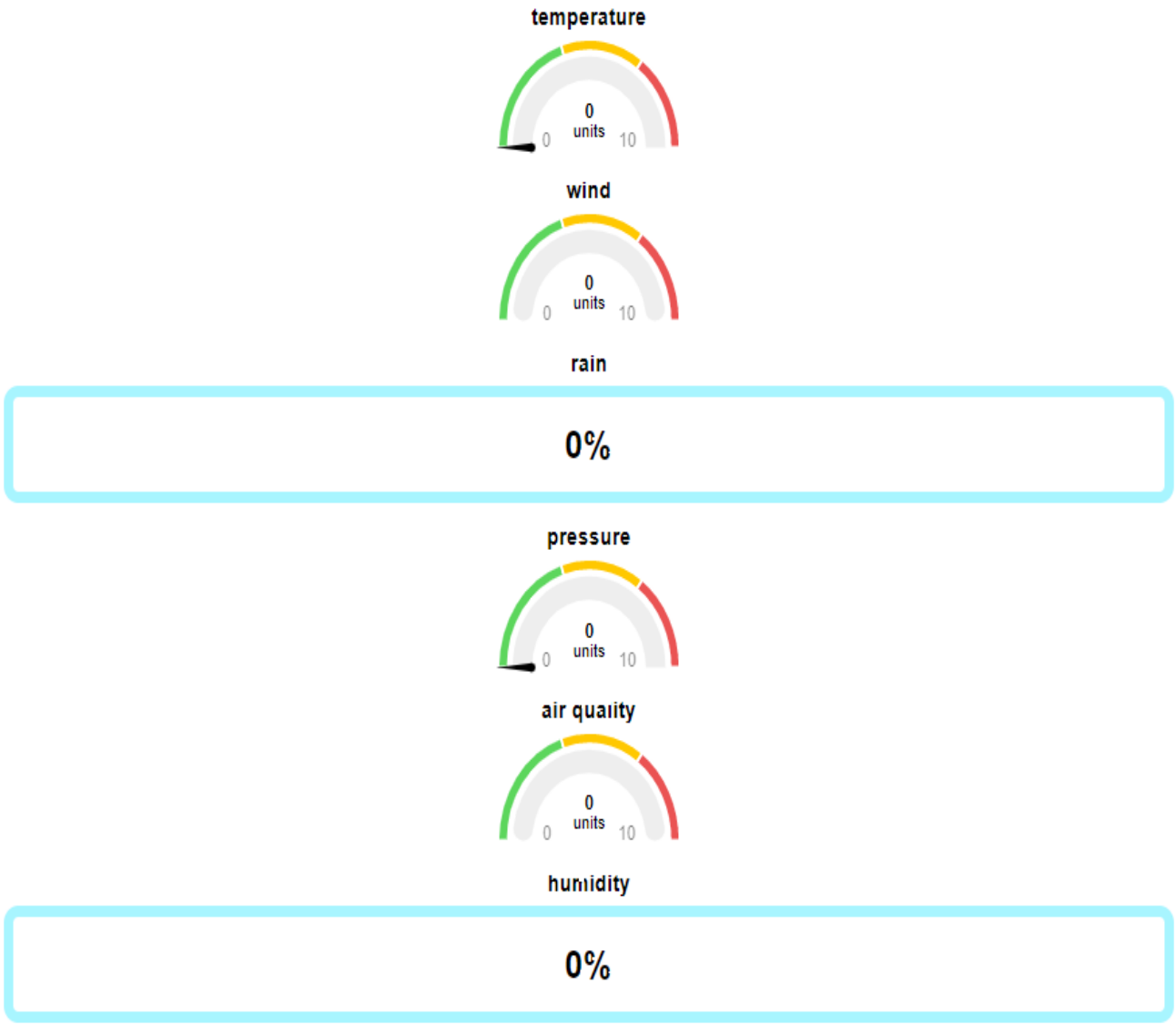

Рисунок 5.2.1 – Блок інформаційної панелі для відображення показників датчиків за допомогою компонентів Gauge

Наступним кроком запущено генерацію даних для тестування роботи системи та коректності відображення показників.

В результаті роботи системи було перевірено правильність та коректність її роботи та відображення отриманих даних.

На наступних рисунках наведено результати роботи системи, зокрема:

– відображення показників датчиків температури повітря та швидкості вітру

за допомогою компонентів Chart (рис. 5.2.2);

– відображення показників датчиків кількості опадів та якості повітря за допомогою компонентів Chart (рис. 5.2.3);

– відображення показників датчиків вологості повітря та атмосферного тиску за допомогою компонентів Chart (рис. 5.2.4);

– відображення показників датчиків температури повітря та швидкості вітру за допомогою компонентів Gauge (рис. 5.2.5);

– відображення показників датчиків кількості опадів та атмосферного тиску за допомогою компонентів Gauge (рис. 5.2.6);

– відображення показників датчиків датчиків якості повітря та вологості повітря за допомогою компонентів Gauge (рис. 5.2.7);

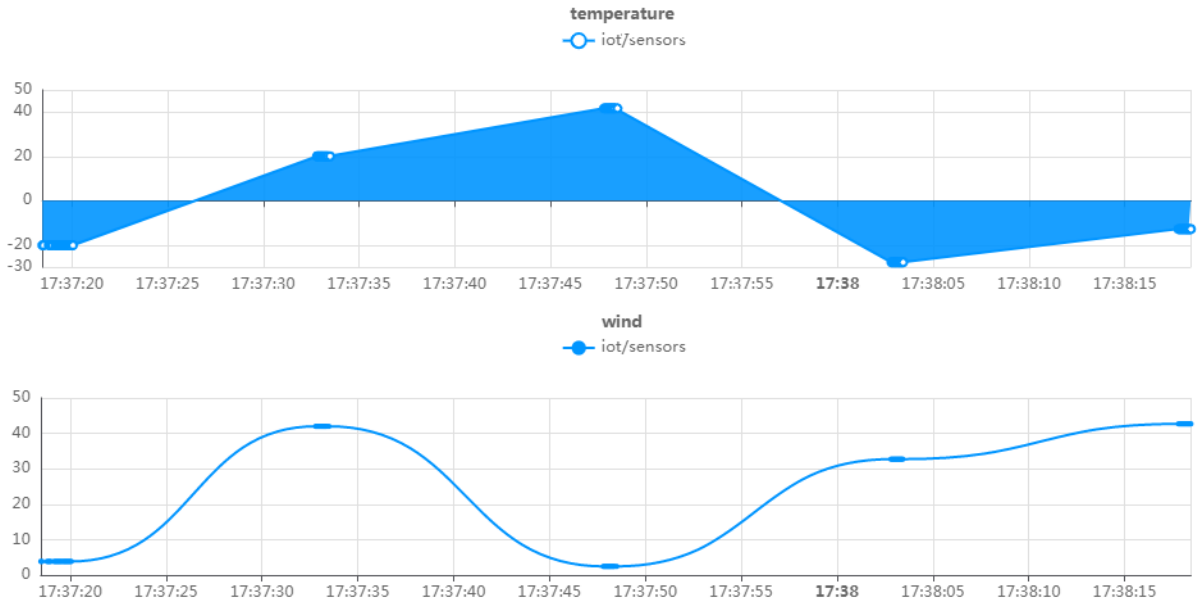

Рисунок 5.2.2 – Відображення показників датчиків температури повітря та швидкості вітру за допомогою компонентів Chart

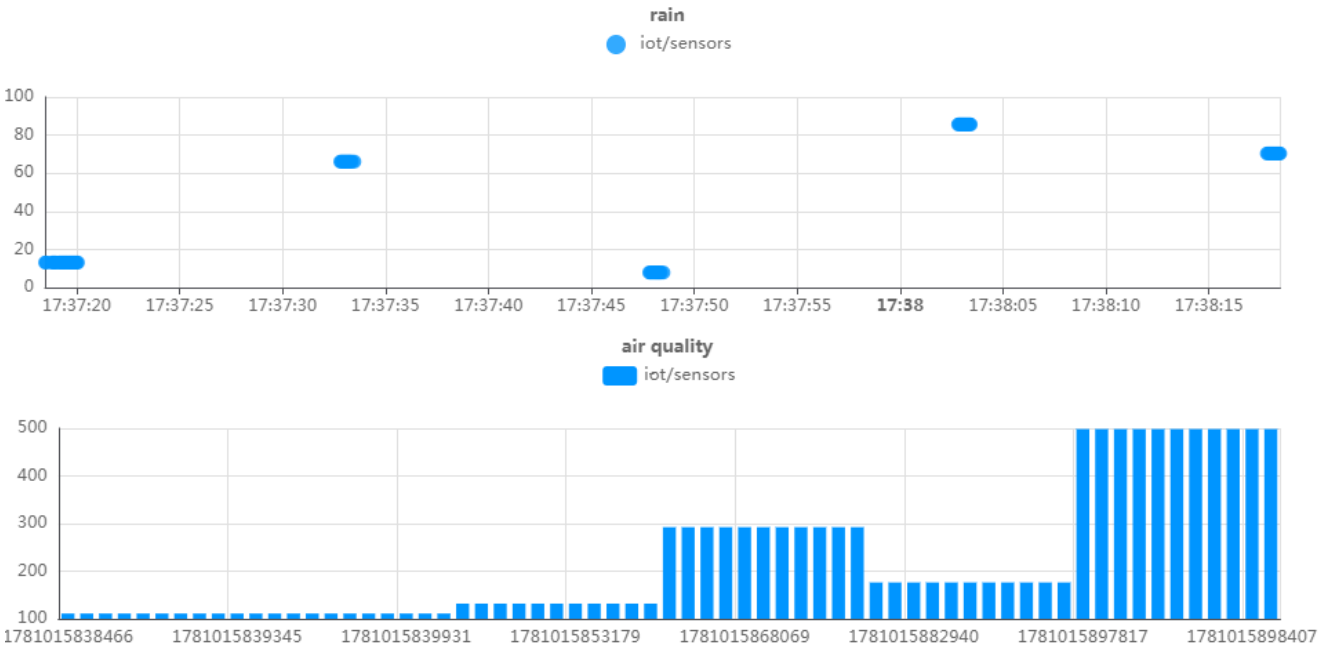

Рисунок 5.2.3 – Відображення показників датчиків кількості опадів та якості повітря за допомогою компонентів Chart

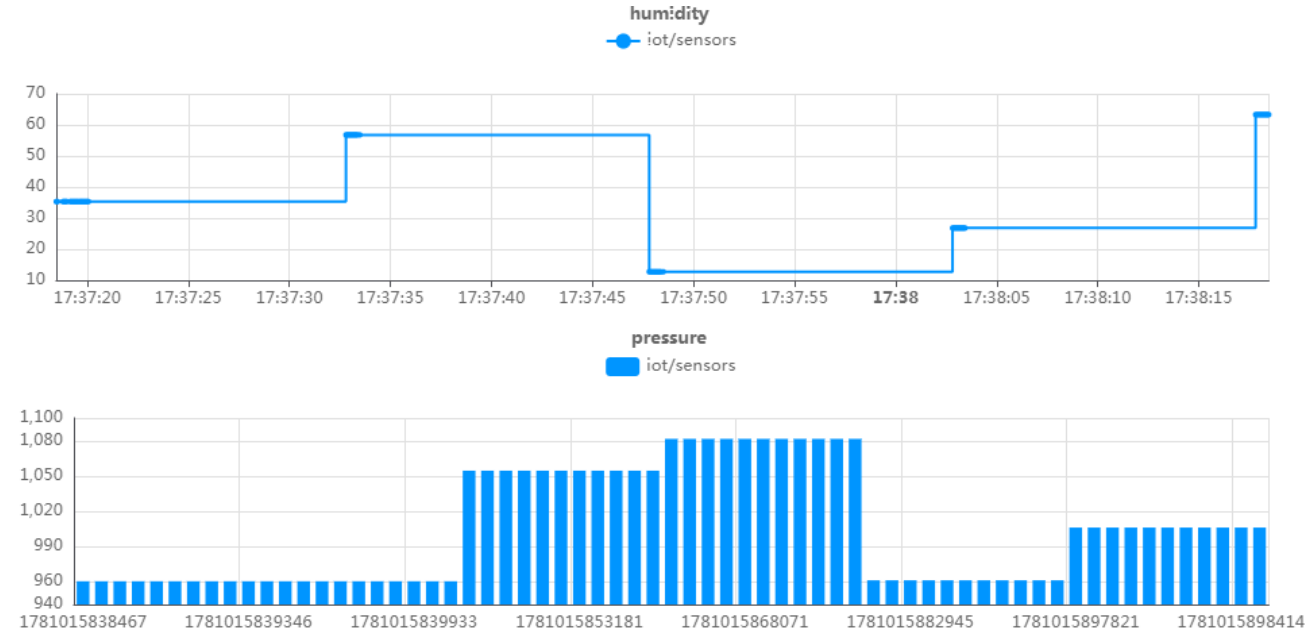

Рисунок 5.2.4 – Відображення показників датчиків вологості повітря та атмосферного тиску за допомогою компонентів Chart

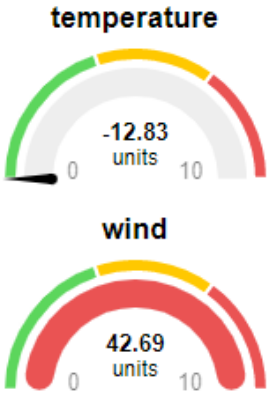

Рисунок 5.2.5 – Відображення показників датчиків температури повітря та швидкості вітру за допомогою компонентів Gauge

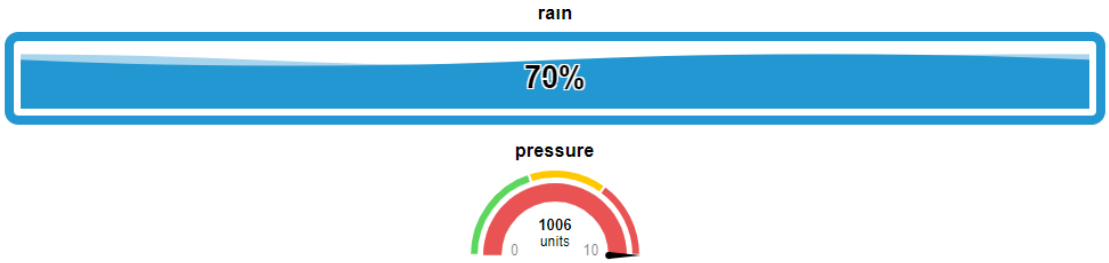

Рисунок 5.2.6 – Відображення показників датчиків кількості опадів та атмосферного тиску за допомогою компонентів Gauge

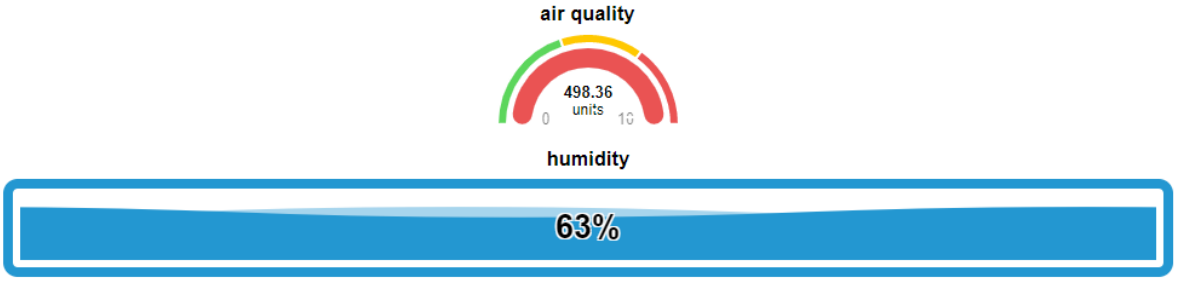

Рисунок 5.2.7 – Відображення показників датчиків якості повітря та вологості повітря за допомогою компонентів Gauge

5.3 Відображення аномалій та повідомлень

Для підвищення ефективності моніторингу в системі реалізовано механізм автоматичного виявлення аномальних ситуацій. Після отримання даних вони передаються до спеціального функціонального вузла аналізу, який виконує перевірку отриманих значень на відповідність допустимим межам.

У випадку виявлення відхилення формується повідомлення про аномалію, яке відображається у відповідному інформаційному блоці Dashboard-панелі. Повідомлення містить інформацію про параметр, що вийшов за допустимий діапазон, та його поточне значення.

Для оперативного реагування додатково використовується механізм звукового сповіщення. При надходженні сигналу про аномалію користувач отримує аудіоповідомлення, що дозволяє звернути увагу на проблему навіть без постійного спостереження за екраном.

Основними перевагами реалізованого механізму є:

- автоматичне виявлення небезпечних ситуацій;
- миттєве інформування користувача;
- зменшення ризику пропуску критичних подій;
- можливість своєчасного прийняття рішень.

Далі було проведено генерацію даних та повноцінне тестування роботи системи щоб впевнитися в коректності її роботи та правильності відображення показників, а також перевірити точність реагування на аномальні показники для запобігання хибних спрацювань або пропуску аномалій.

На рисунках 5.3 – 5.3.5 наведено результат роботи системи, зокрема: процес генерації, надсилання та отримання показників системою та спрацювання і надсилання повідомлень та сповіщень при аномальних показниках.

Результатом тестування системи та механізму виявлення аномалій і сповіщення про них стало підтвердженням валідності та правильності її роботи.

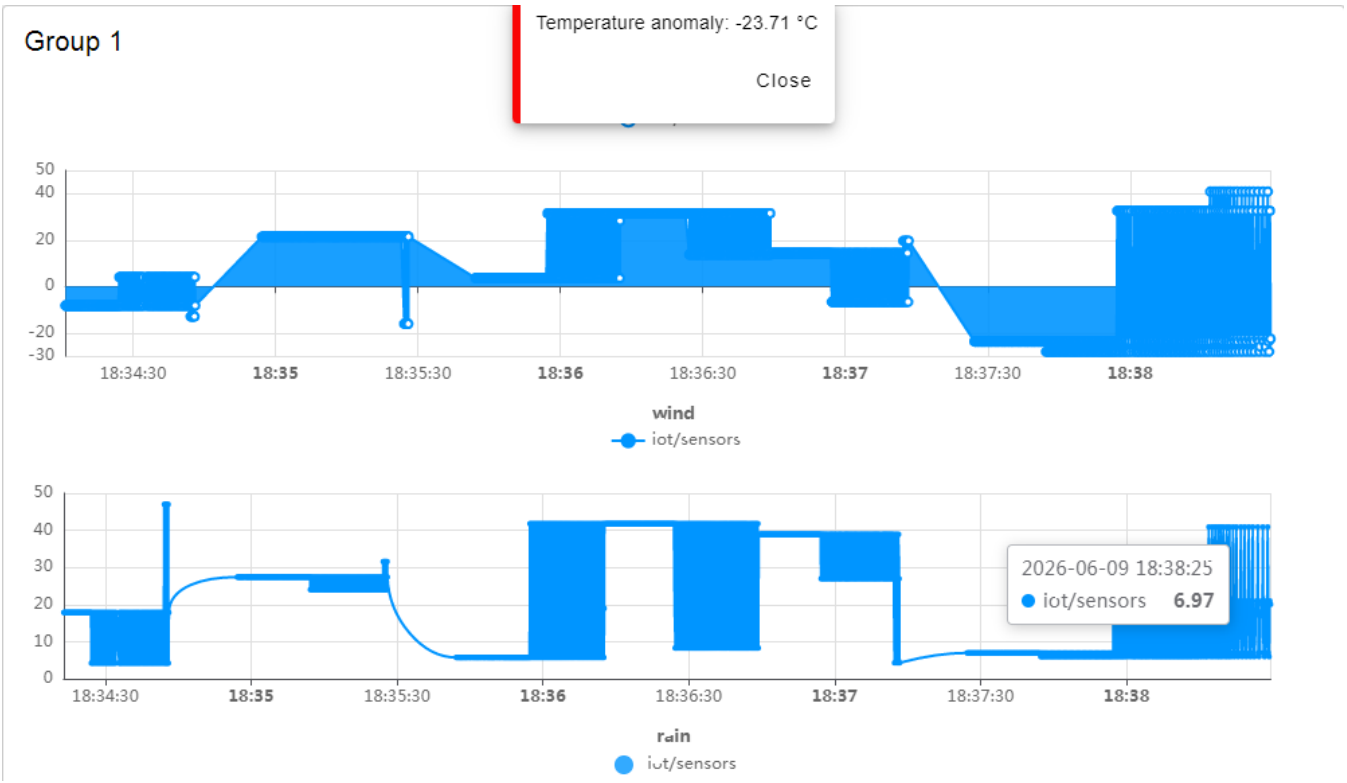

Рисунок 5.3 – Приклад відображення повідомлення про аномальну температуру повітря у Dashboard-панелі

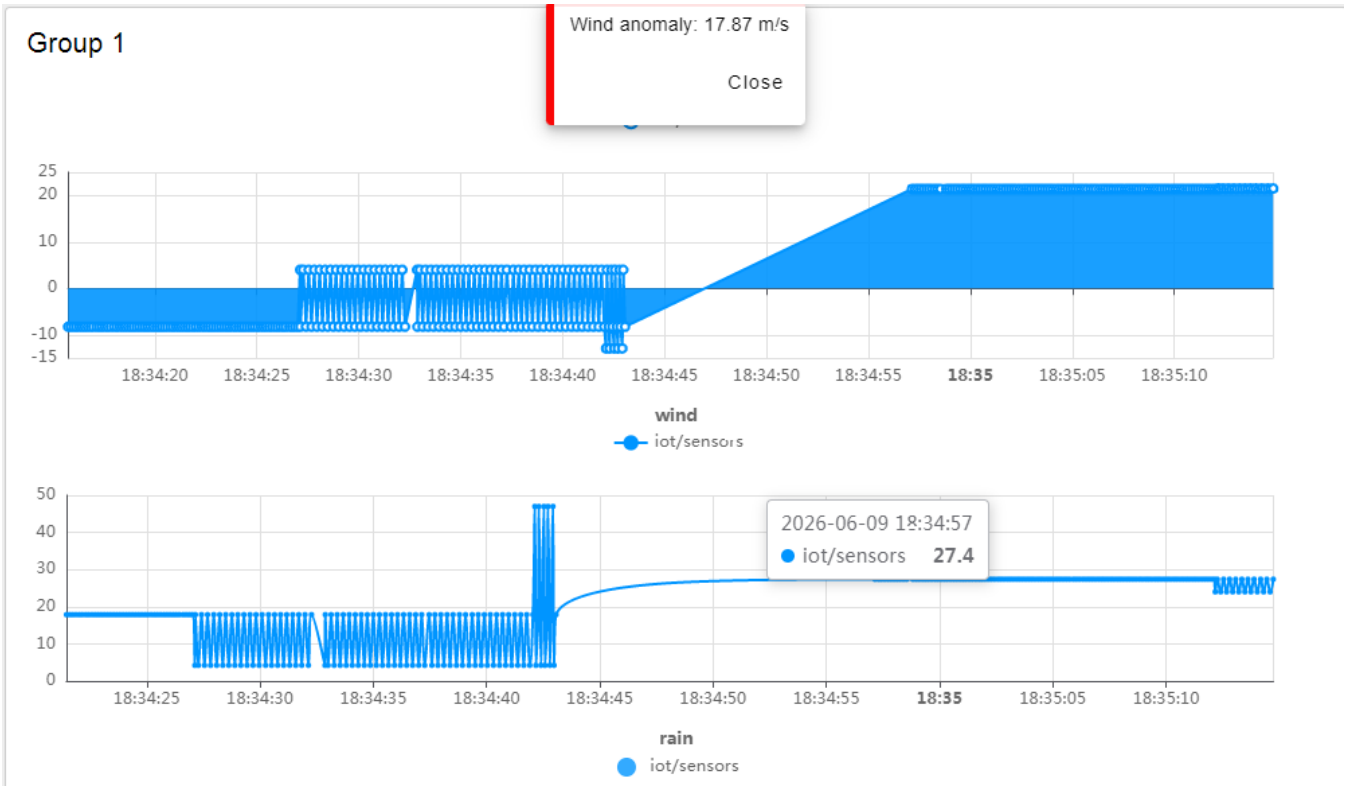

Рисунок 5.3.1 – Приклад відображення повідомлення про аномальну швидкість вітру у Dashboard-панелі

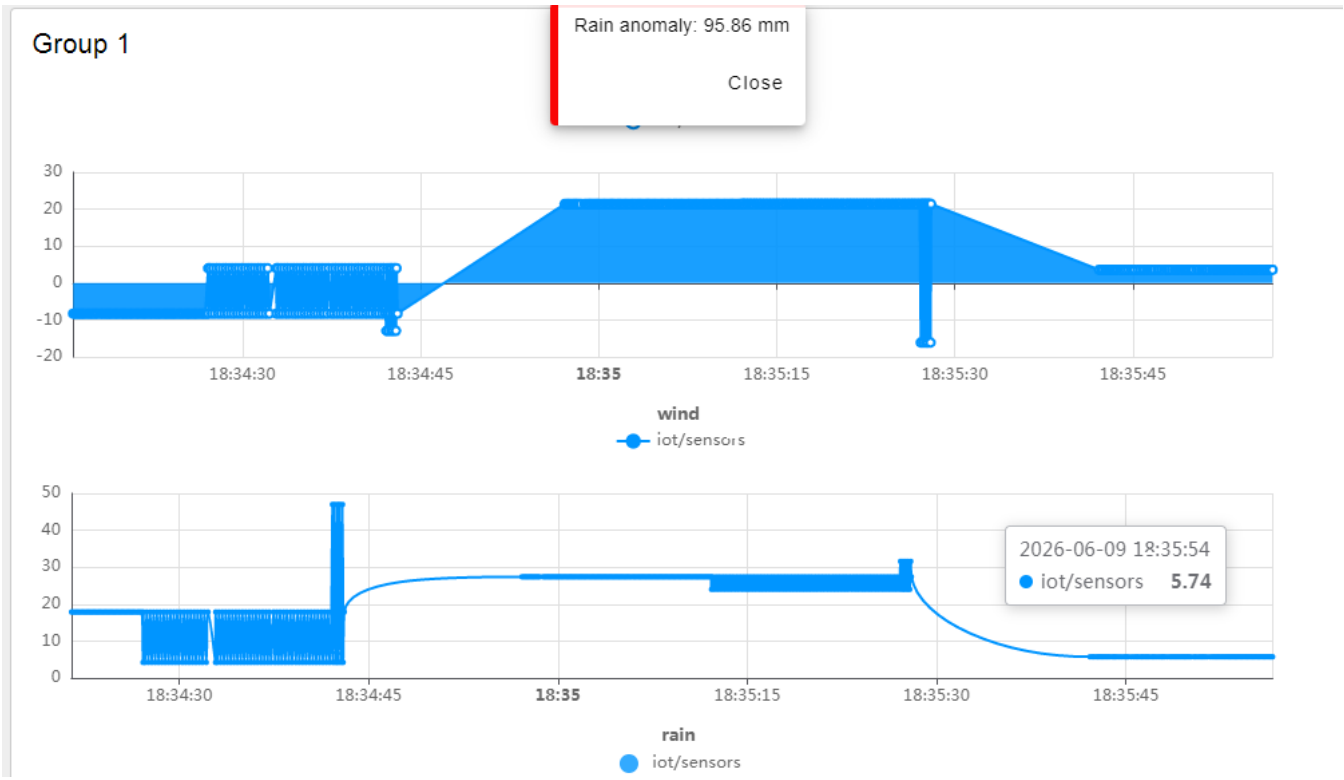

Рисунок 5.3.2 – Приклад відображення повідомлення про аномальну кількість опадів у Dashboard-панелі

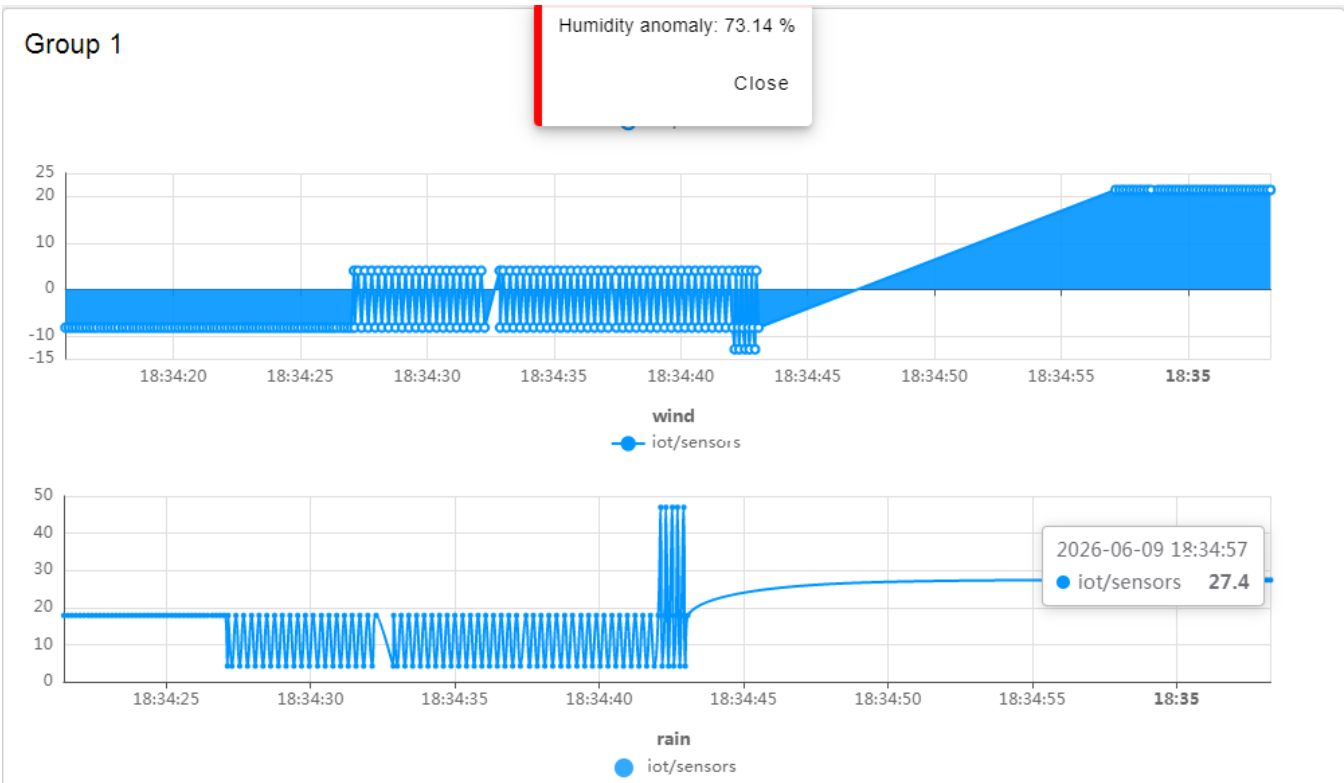

Рисунок 5.3.3 – Приклад відображення повідомлення про аномальну вологість повітря у Dashboard-панелі

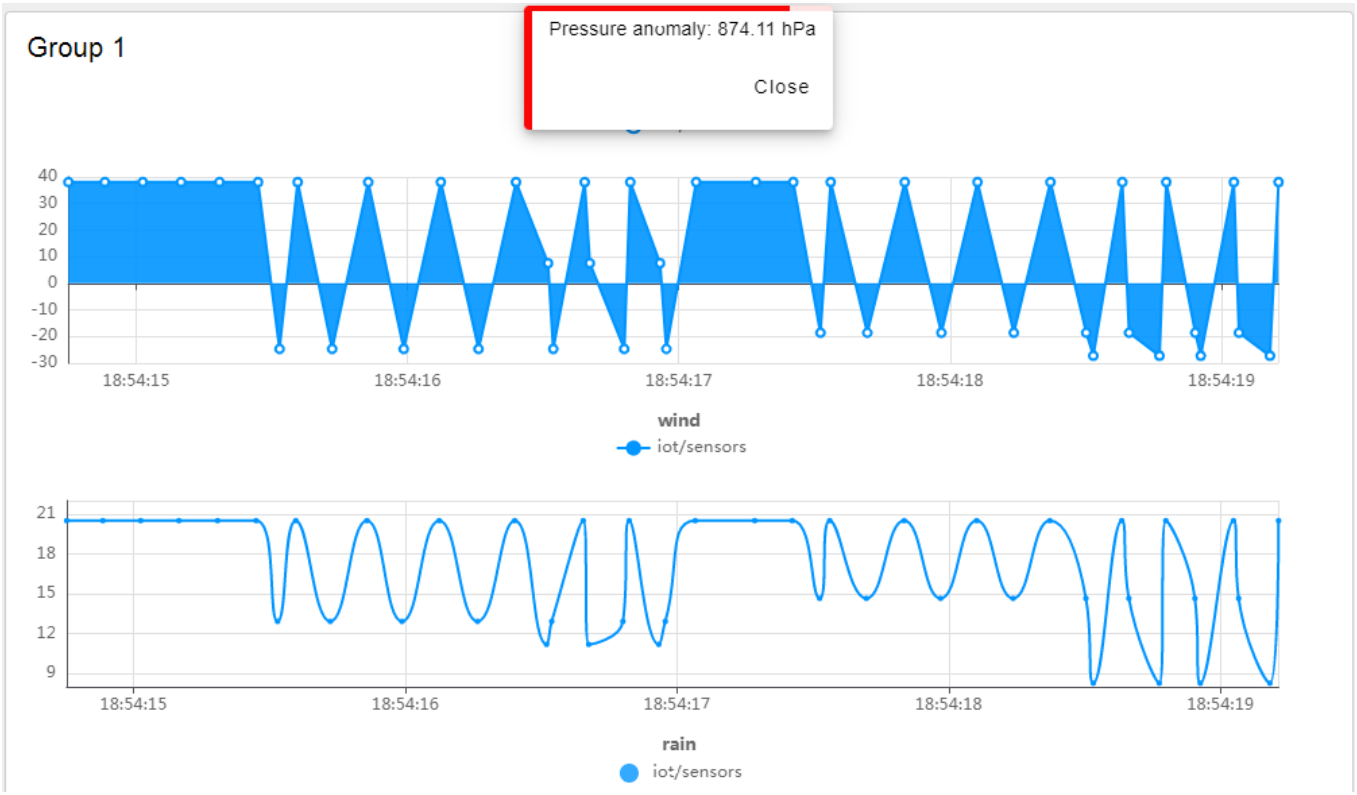

Рисунок 5.3.4 – Приклад відображення повідомлення про аномальний атмосферний тиск у Dashboard-панелі

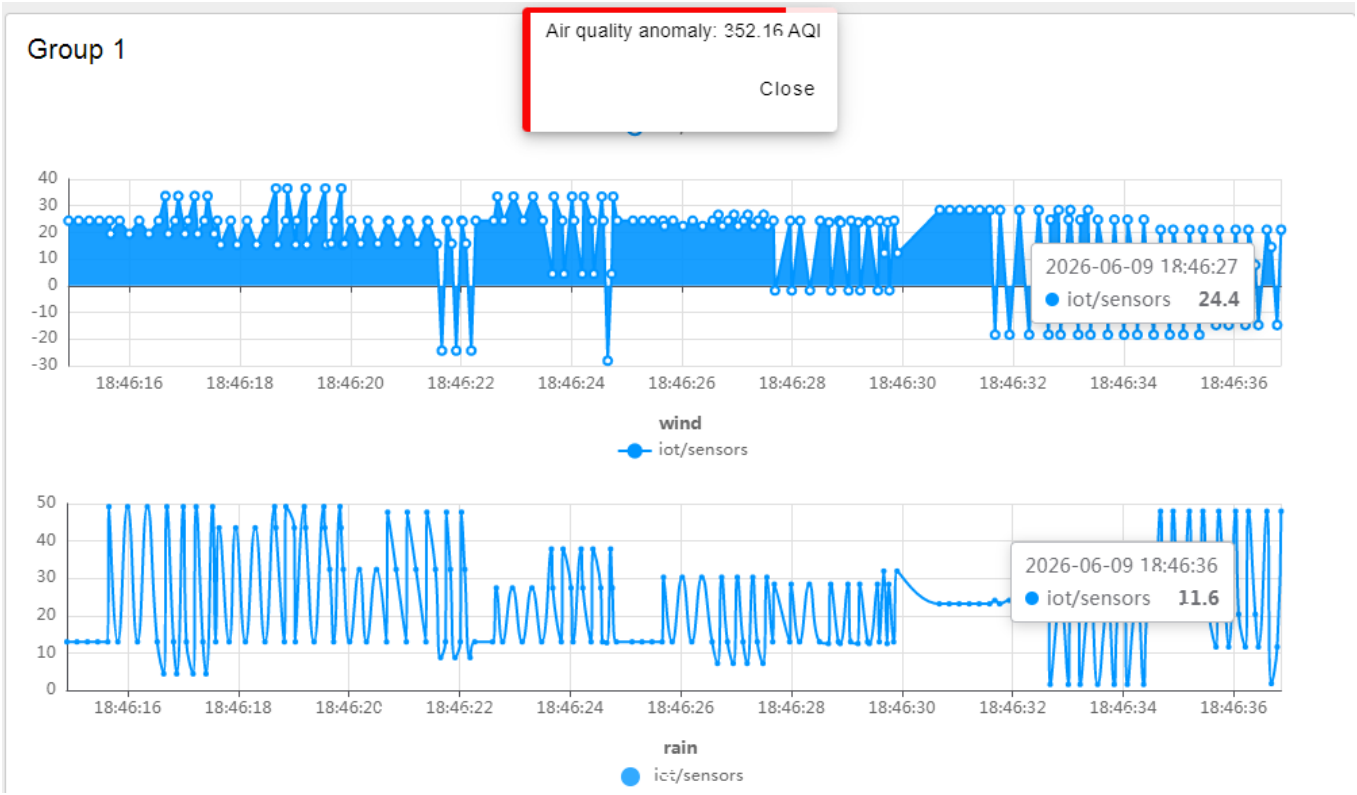

Рисунок 5.3.5 – Приклад відображення повідомлення про аномальну якість повітря у Dashboard-панелі

5.4 Сценарії взаємодії користувача із системою

Взаємодія користувача із системою здійснюється через веб-браузер. Після відкриття Dashboard-панелі користувач отримує доступ до всіх функцій моніторингу в режимі реального часу.

Основний сценарій використання системи передбачає такі етапи:

- відкриття інформаційної панелі моніторингу;
- отримання актуальних показників від датчиків;
- перегляд поточних значень параметрів;
- аналіз графіків зміни показників;
- отримання повідомлень про аномальні ситуації;
- реагування на отримані попередження.

Альтернативний сценарій виникає у випадку появи аномалії. Система автоматично відображає відповідне повідомлення та активує звукове оповіщення. Після цього користувач може проаналізувати причину відхилення та прийняти необхідні заходи.

Запропонований інтерфейс забезпечує просту та інтуїтивно зрозумілу взаємодію із системою. Завдяки використанню сучасних засобів візуалізації та автоматичного сповіщення користувач отримує повний контроль над процесом моніторингу стану навколишнього середовища.

На рисунку 5.4 наведено загальний вигляд користувацького інтерфейсу системи моніторингу у Node-RED, що включає у себе раніше обрані, спроєктовані та запрограмовані компоненти що забезпечують генерацію, отримання, обробку та аналіз даних.

Результатом проєктування користувацького інтерфейсу та перевірки його роботи стало підтвердження роботи системи при навантаженні (обробці та аналізу великих масивів даних), відсутність хибних спрацювань, а також швидкість роботи системи загалом.

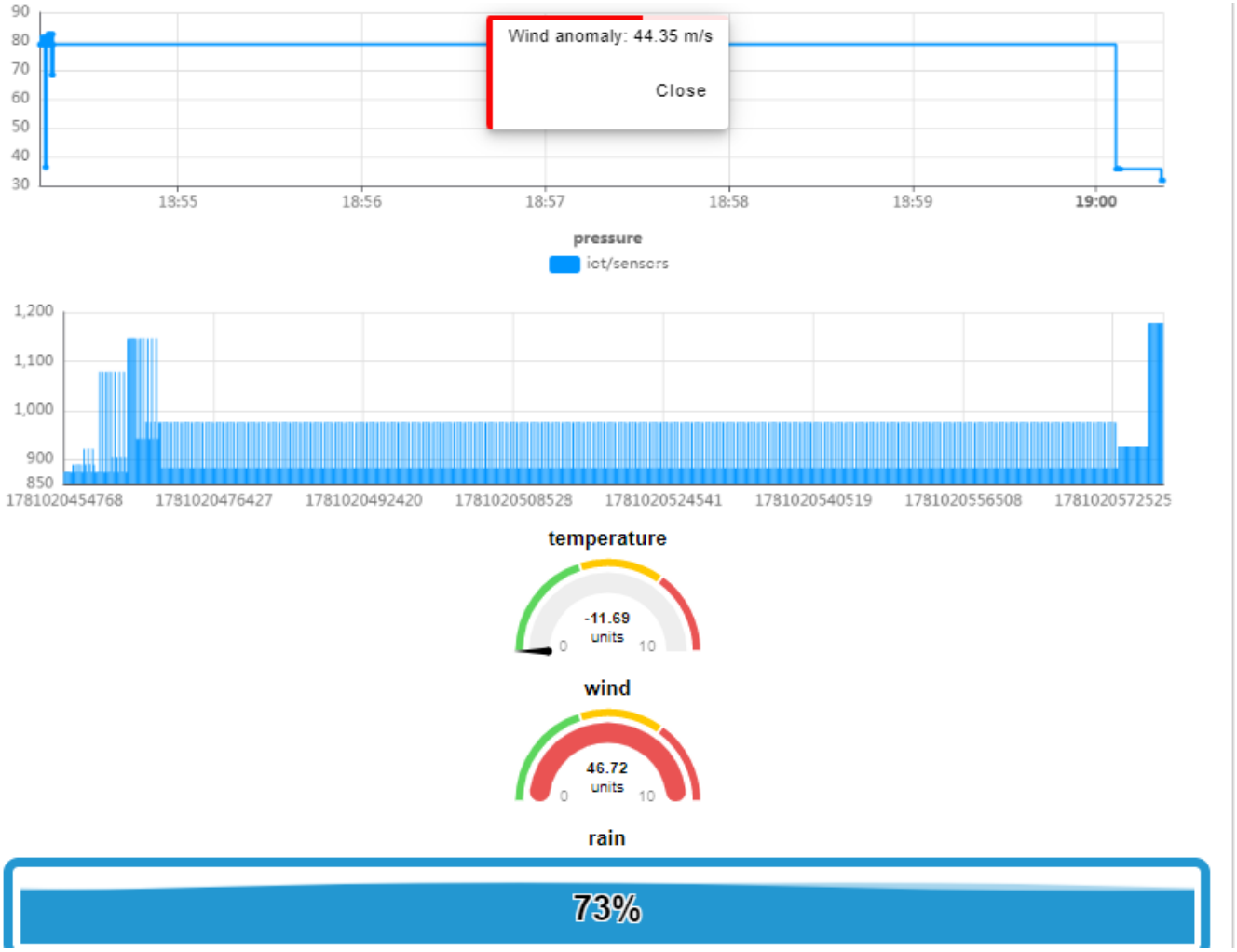

Рисунок 5.4 – Загальний вигляд користувацького інтерфейсу системи моніторингу у Node-RED

ВИСНОВКИ

У ході виконання дипломного проєкту було розроблено програмну систему моніторингу параметрів навколишнього середовища на основі технологій Python, MQTT та Node-RED. Основною метою роботи було створення програмного забезпечення, здатного здійснювати збір, передачу, обробку, візуалізацію та аналіз даних у режимі реального часу з автоматичним виявленням аномальних ситуацій. Поставлена мета була повністю досягнута.

Під час виконання роботи було проведено аналіз предметної області та досліджено сучасні підходи до побудови систем моніторингу на базі технологій Інтернету речей. Результати аналізу показали актуальність використання MQTT-протоколу для передачі телеметричних даних та платформи Node-RED для реалізації механізмів обробки інформації й побудови користувацьких інтерфейсів.

На етапі формування вимог були визначені функціональні та нефункціональні характеристики системи. До основних функціональних можливостей належать приймання даних від датчиків, їх обробка, відображення на інформаційній панелі, зберігання поточного стану параметрів та автоматичне виявлення аномалій. Також було визначено вимоги щодо продуктивності, надійності, масштабованості та зручності використання системи.

У процесі моделювання програмного забезпечення були розроблені UML-діаграми, що описують логіку роботи системи, сценарії взаємодії користувача та послідовність виконання основних операцій. Отримані результати дозволили сформувати цілісне уявлення про структуру програмного забезпечення та взаємозв’язки між його компонентами.

У межах проєктування архітектури було реалізовано систему, яка складається з модуля генерації даних на мові Python, MQTT-брокера для передачі повідомлень, середовища Node-RED для обробки інформації та веб-інтерфейсу Dashboard для відображення результатів моніторингу. Передача даних

здійснюється через MQTT-брокер broker.emqx.io, що забезпечує швидкий та надійний обмін повідомленнями між компонентами системи.

Розроблена система здійснює моніторинг шести основних параметрів навколишнього середовища: температури повітря, вологості повітря, атмосферного тиску, швидкості вітру, рівня опадів та якості повітря. Для кожного параметра реалізовано окремі елементи візуалізації у вигляді індикаторів та графіків, що дозволяє користувачу отримувати актуальну інформацію про стан контрольованого середовища та аналізувати зміну показників у часі.

Особливу увагу було приділено реалізації механізму виявлення аномалій. У системі створено окремий модуль аналізу отриманих даних, який автоматично визначає відхилення від встановлених допустимих значень. У разі виявлення аномальної ситуації система формує повідомлення для користувача та активує звукове сповіщення, що дозволяє оперативно реагувати на потенційно небезпечні події.

Розроблений користувацький інтерфейс забезпечує наочне відображення інформації та просту взаємодію із системою. Використання Dashboard-панелі Node-RED дозволило реалізувати сучасний веб-інтерфейс із можливістю перегляду поточних показників, аналізу графіків та отримання повідомлень про аномальні ситуації в режимі реального часу.

Отримані результати повністю відповідають поставленому завданню та висунутим вимогам. Розроблена система може бути використана для моніторингу параметрів навколишнього середовища, елементів інфраструктури Інтернету речей, промислових об’єктів, аграрних комплексів та інших сфер, де необхідний оперативний контроль стану контрольованих параметрів. Подальший розвиток проєкту може бути пов’язаний із підключенням реальних датчиків, інтеграцією баз даних для довгострокового зберігання інформації, використанням алгоритмів машинного навчання для прогнозування аномалій та розширенням функціональних можливостей системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Node-RED Documentation. OpenJS Foundation. URL: https://nodered.org/docs/ (дата звернення: 01.05.2026)

2. MQTT Essentials – A Lightweight IoT Protocol. HiveMQ. URL: https://www.hivemq.com/mqtt-essentials/ (дата звернення: 10.05.2026).

3. EMQX MQTT Broker Documentation. EMQ Technologies. URL: https://www.emqx.io/docs/en/latest/ (дата звернення: 10.05.2026).

4. Python Documentation. Python Software Foundation. URL: https://docs.python.org/3/ (дата звернення: 20.04.2026).

5. Node.js Documentation. OpenJS Foundation. URL: https://nodejs.org/en/docs (дата звернення: 25.04.2026).

6. Express.js Guide. Express.js Foundation. URL: https://expressjs.com/ (дата звернення: 25.04.2026).

7. MQTT Protocol Specification. OASIS Standard. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (дата звернення: 22.04.2026).

8. Grafana / Dashboard Visualization Concepts. Grafana Labs. URL: https://grafana.com/docs/ (дата звернення: 05.05.2026).

9. IBM. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.ibm.com/topics/internet-of-things (дата звернення: 15.05.2026).

10. Tutorialspoint. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.tutorialspoint.com/internet_of_things/internet_of_things_hardware.html

(дата звернення: 25.05.2026).

11. HiveMQ. [Електронний ресурс]. – Режим доступу до ресурсу: https://www.hivemq.com/blog/mqtt-essentials-part-8-retained-messages/.

(дата звернення: 20.05.2026).

ДОДАТОК А
ЛІСТИНГИ ПРОГРАМ

Лістинг А.1 – Файл send_data_mqtt.py

```
import paho.mqtt.client as mqtt
import json
import random
import time

# MQTT broker
BROKER = "broker.emqx.io"
PORT = 1883
TOPIC = "iot/sensors"

client = mqtt.Client()
client.connect(BROKER, PORT, 60)

def publish_data():
    data = {
        "temperature": round(random.uniform(-30.0, 45.0), 2),
        "wind": round(random.uniform(0.0, 50.0), 2),
        "rain": round(random.uniform(0.0, 100.0), 2),
        "humidity": round(random.uniform(10.0, 100.0), 2),
        "air_quality": round(random.uniform(0.0, 500.0), 2),
        "pressure": round(random.uniform(800.0, 1200.0), 2)
    }

    payload = json.dumps(data)
    client.publish(TOPIC, payload)
    print("Sent:", payload)

try:
    while True:
        publish_data()
```

```
        time.sleep(15)

except KeyboardInterrupt:
    print("Stopped")

finally:
    client.disconnect()
```

Лістинг А.2 – Файл function sensor_data.java

```
let d = msg.payload;

if (!d) return null;

if (typeof d === "string") {
    try {
        d = JSON.parse(d);
    } catch (e) {
        node.warn("Bad JSON: " + d);
        return null;
    }
}

msg.payload = d;

return msg;
```

Лістинг А.3 – Файл function function_anomaly_notification.java

```
let d = msg.payload;

if (typeof d === "string") {
    try {
        d = JSON.parse(d);
    } catch (e) {
        return null;
    }
}

let alerts = [];

if (d.temperature < -10 || d.temperature > 10) {
    alerts.push("Temperature anomaly: " + d.temperature + " °C");
}

if (d.wind < 0 || d.wind > 10) {
    alerts.push("Wind anomaly: " + d.wind + " m/s");
```

```
}

if (d.rain < 0 || d.rain > 10) {
    alerts.push("Rain anomaly: " + d.rain + " mm");
}

if (d.humidity < 10 || d.humidity > 50) {
    alerts.push("Humidity anomaly: " + d.humidity + " %");
}

if (d.air_quality < 0 || d.air_quality > 300) {
    alerts.push("Air quality anomaly: " + d.air_quality + " AQI");
}

if (d.pressure < 900 || d.pressure > 1100) {
    alerts.push("Pressure anomaly: " + d.pressure + " hPa");
}

if (alerts.length === 0) {
    return null;
}

return [alerts.map(a => ({ payload: a }))];
```

Лістинг А.4 – Файл function function_anomaly_alarm.java

```
let d = msg.payload;

if (typeof d === "string") {
    try {
        d = JSON.parse(d);
    } catch (e) {
        return null;
    }
}

let alerts = [];

if (d.temperature < -10 || d.temperature > 10) {
    alerts.push("Warning. Temperature anomaly detected. Value " + d.temperature + "
degrees Celsius");
}

if (d.wind < 0 || d.wind > 10) {
    alerts.push("Warning. Wind anomaly detected. Value " + d.wind + " meters per
second");
}

if (d.rain < 0 || d.rain > 10) {
    alerts.push("Warning. Rain anomaly detected. Value " + d.rain + " millimeters");
```

```
}

if (d.humidity < 10 || d.humidity > 50) {
    alerts.push("Warning. Humidity anomaly detected. Value " + d.humidity + "
percent");
}

if (d.air_quality < 0 || d.air_quality > 300) {
    alerts.push("Warning. Air quality anomaly detected. Value " + d.air_quality + "
air quality index");
}

if (d.pressure < 900 || d.pressure > 1100) {
    alerts.push("Warning. Pressure anomaly detected. Value " + d.pressure + "
hectopascals");
}

if (alerts.length === 0) {
    return null;
}

return [alerts.map(a => ({ payload: a }))];
```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

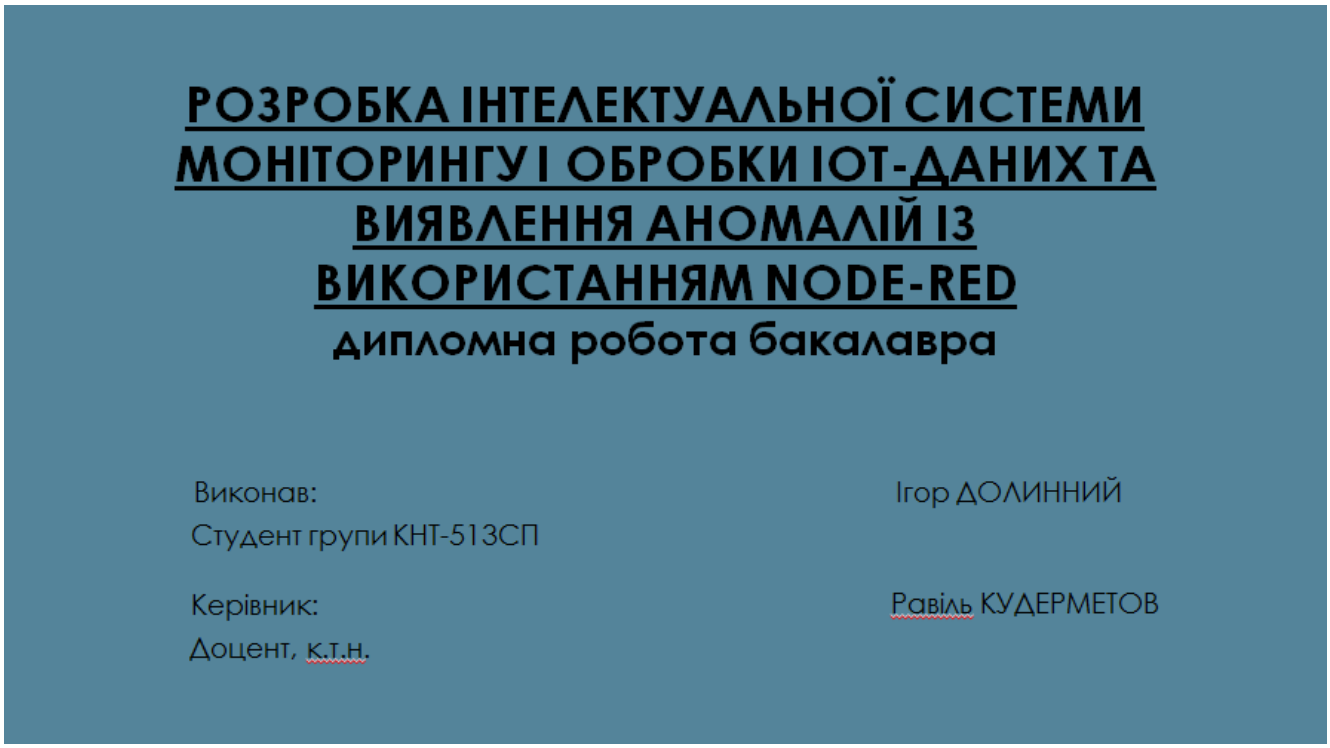

Рисунок Б.1 – Слайд 1

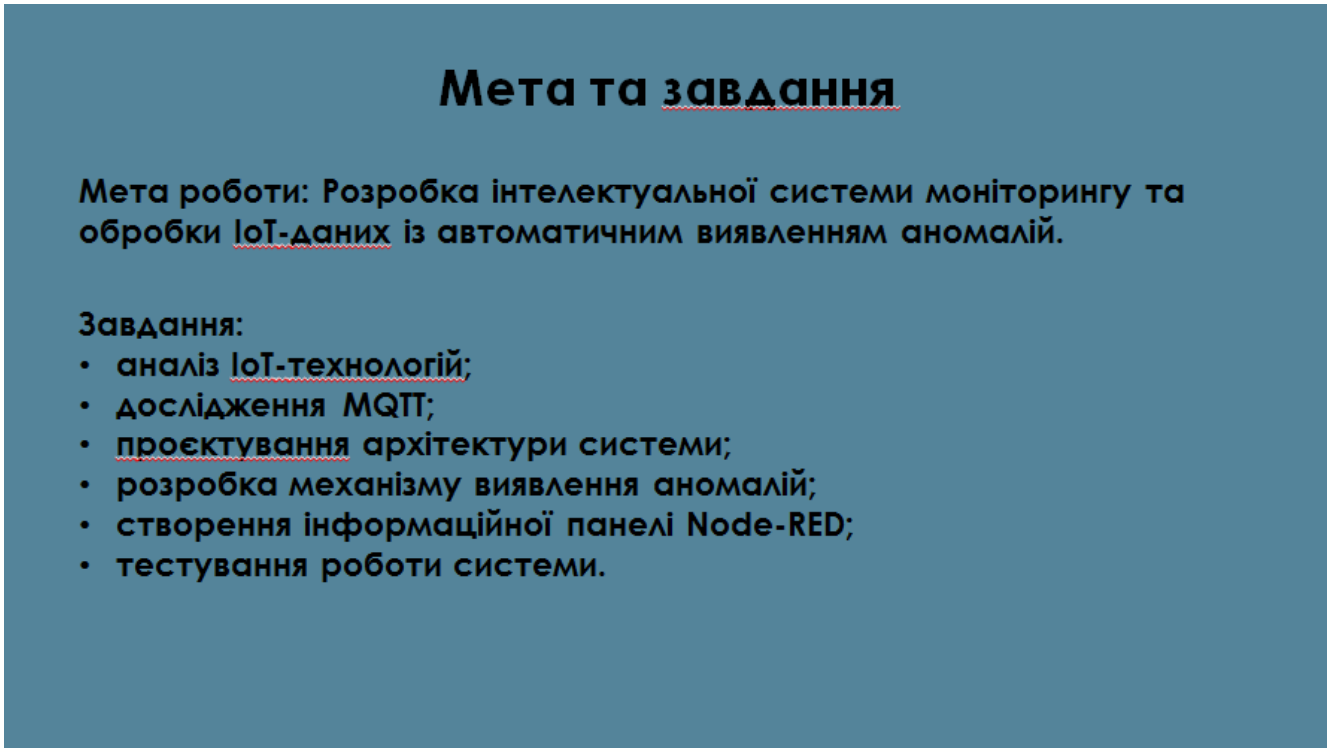

Рисунок Б.2 – Слайд 2

Актуальність роботи

Актуальність роботи тісно пов'язана з наступними чинниками:

- стрімке зростання кількості IoT-пристроїв;
- необхідність контролю стану обладнання;
- потреба в оперативному реагуванні на відхилення;
- використання технологій реального часу.

Рисунок Б.3 – Слайд 3

Апаратна складова системи

У роботі використовувалося програмне моделювання показників датчиків. Згенеровані дані відповідають характеристикам сенсорів DHT22, BMP280 та MQ-135.

Рисунок 1 – Сенсор DHT22

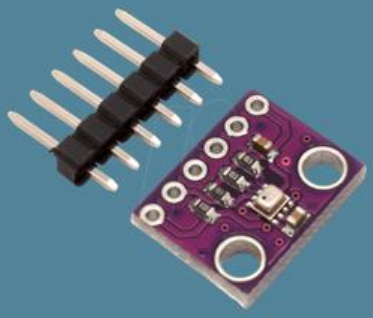

Рисунок 2 – Сенсор BMP280

Рисунок 3 – Сенсор MQ-135

Рисунок Б.4 – Слайд 4

Загальна архітектура системи

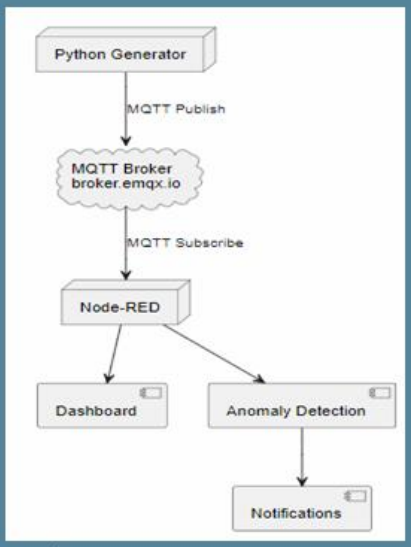

Рисунок 4 – Архітектура інтелектуальної системи моніторингу IoT-даних

Рисунок Б.5 – Слайд 5

Передача даних через MQTT

У системі реалізовано:

- ❑ протокол MQTT;
- ❑ модель Publisher/Subscriber;
- ❑ мінімальне мережеве навантаження;
- ❑ підтримка роботи в режимі реального часу.

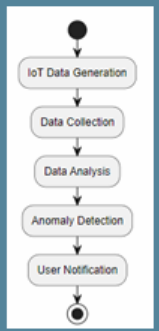

Рисунок 5 – Загальна схема функціонування системи

Рисунок Б.6 – Слайд 6

Діаграма прецедентів (UML Use Case)

Актор:
- Користувач

Прецеденти:
- перегляд показників;
- перегляд графіків;
- отримання сповіщень;
- контроль аномалій.

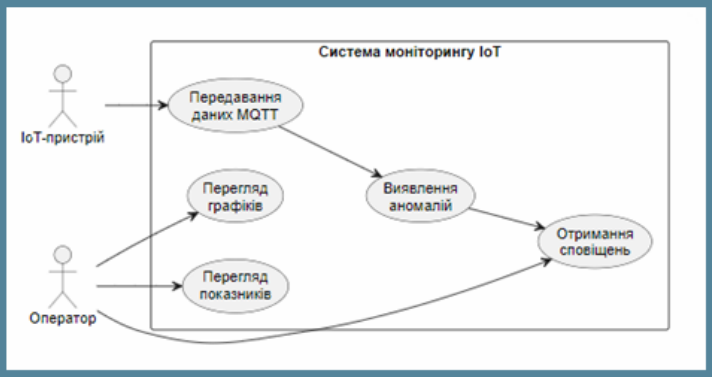

Рисунок 6 – Діаграма варіантів використання системи моніторингу

Рисунок Б.7 – Слайд 7

Алгоритм обробки даних

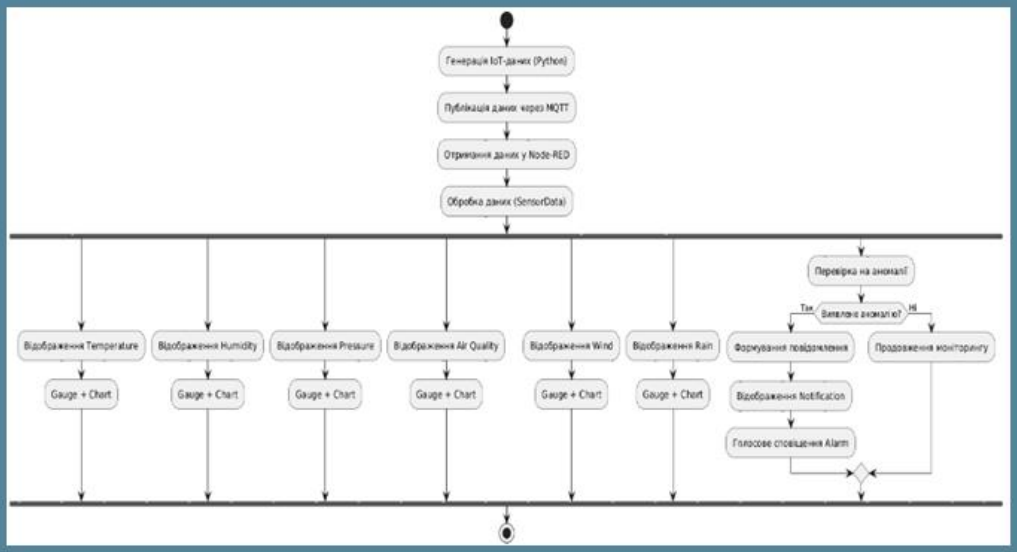

Рисунок 7 – Діаграма діяльності системи

Рисунок Б.8 – Слайд 8

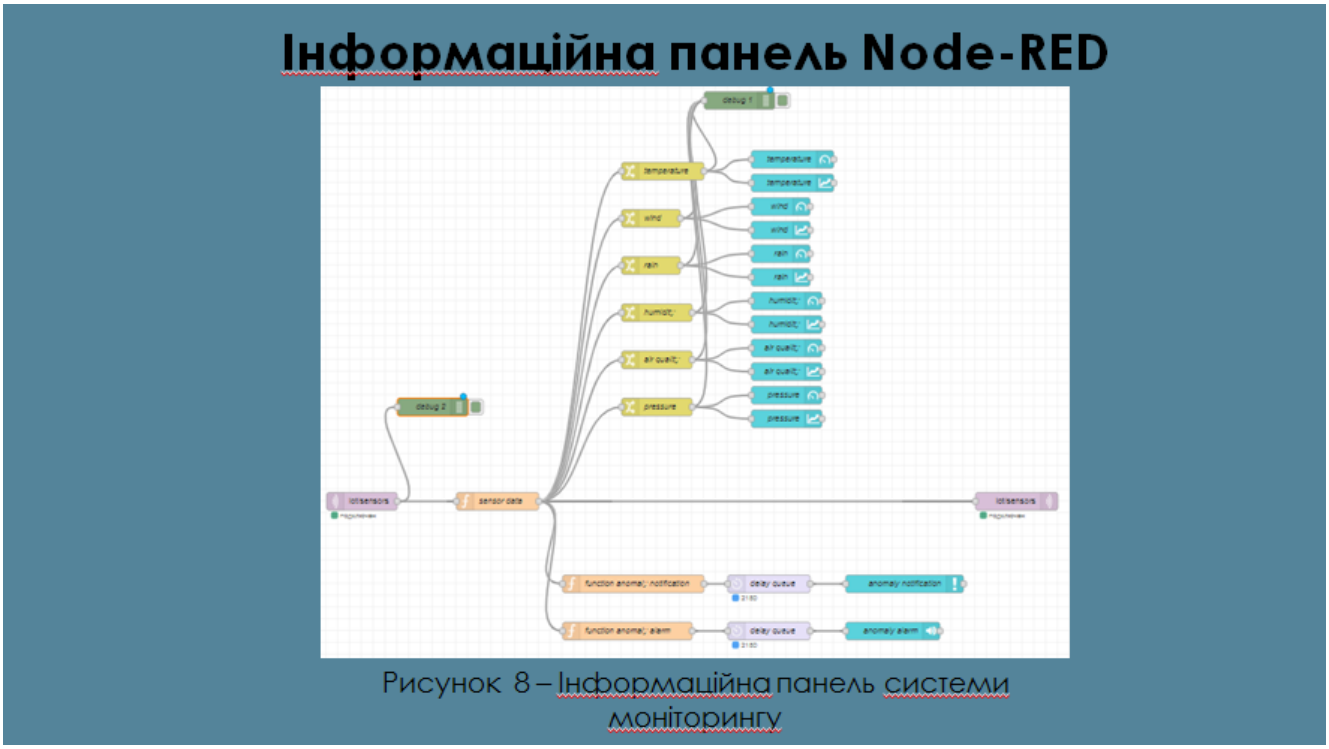

Рисунок Б.9 – Слайд 9

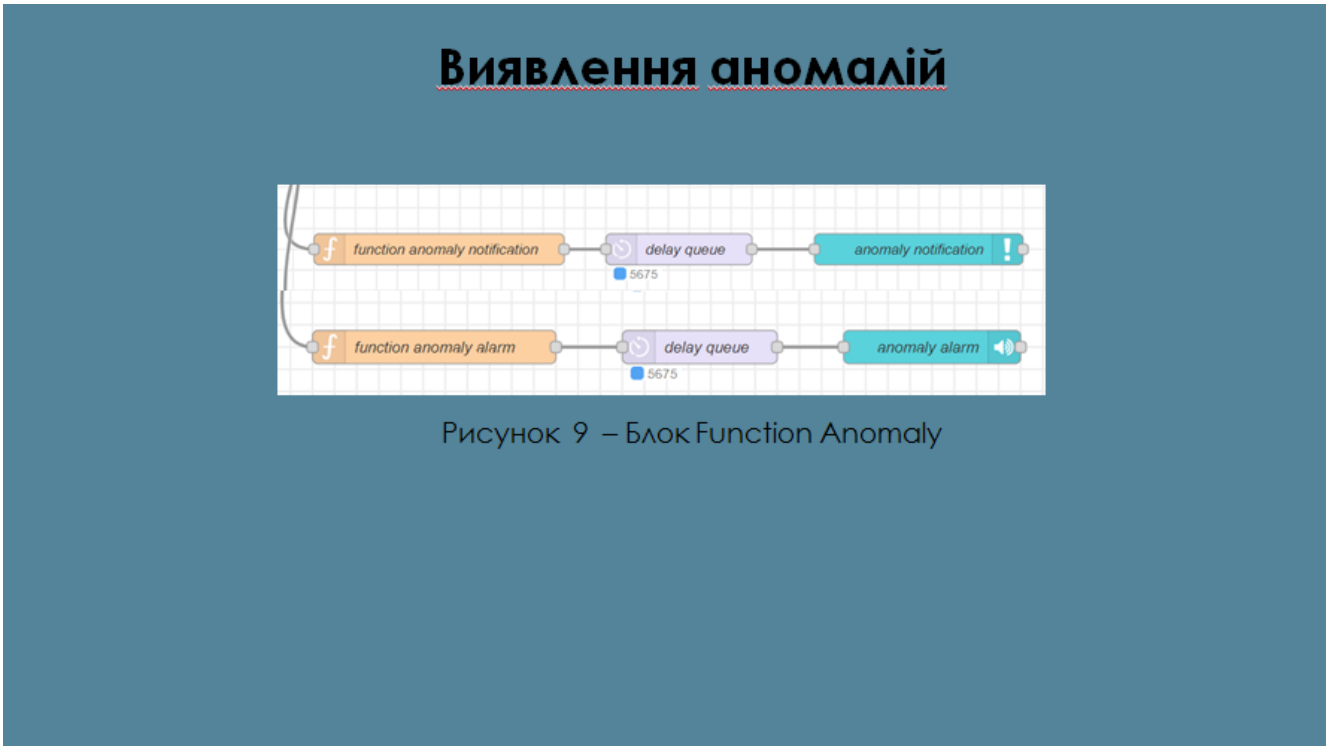

Рисунок Б.10 – Слайд 10

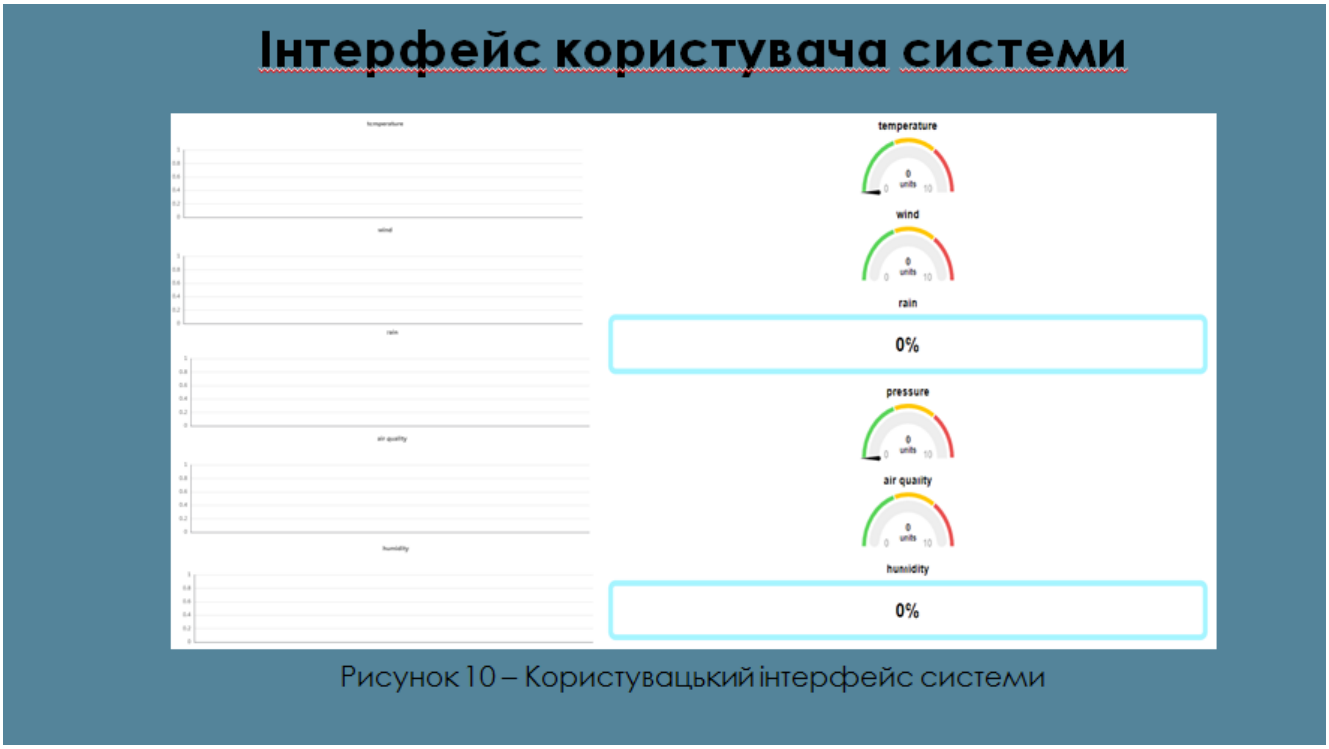

Рисунок Б.11 – Слайд 11

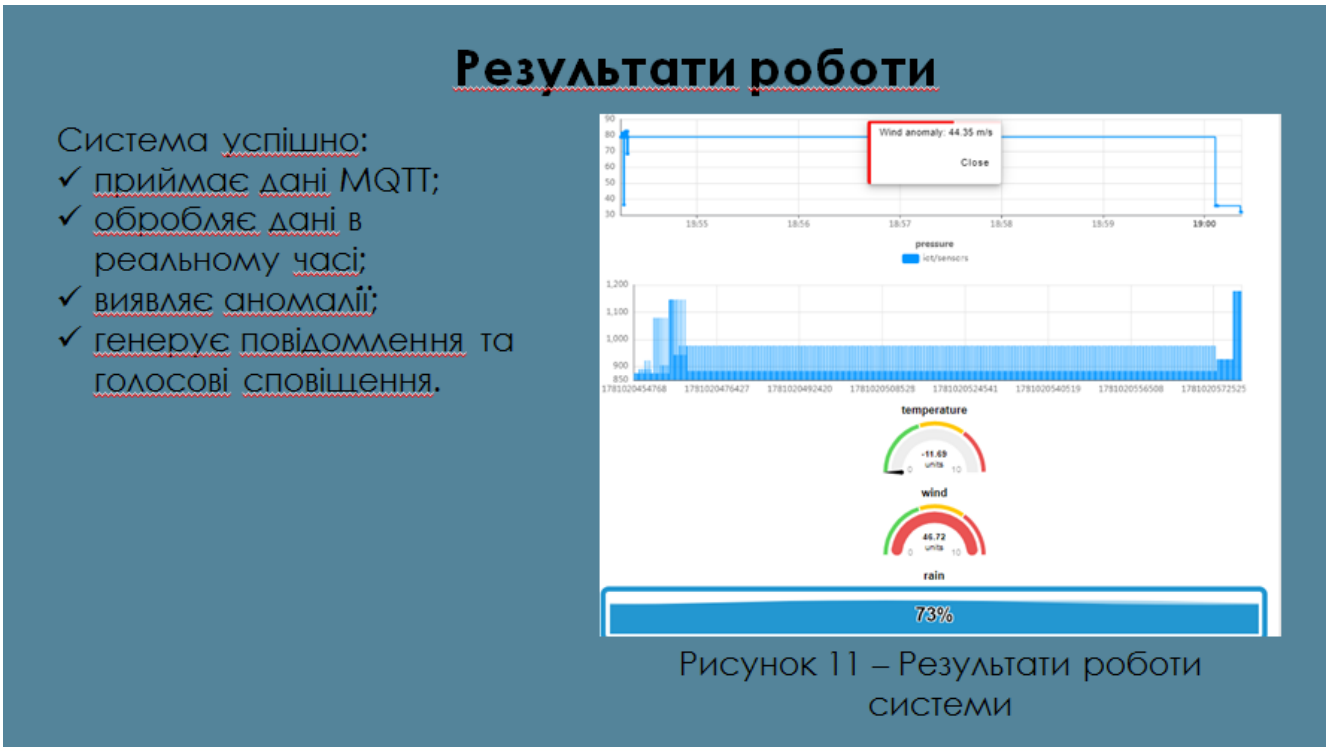

Рисунок Б.12 – Слайд 12

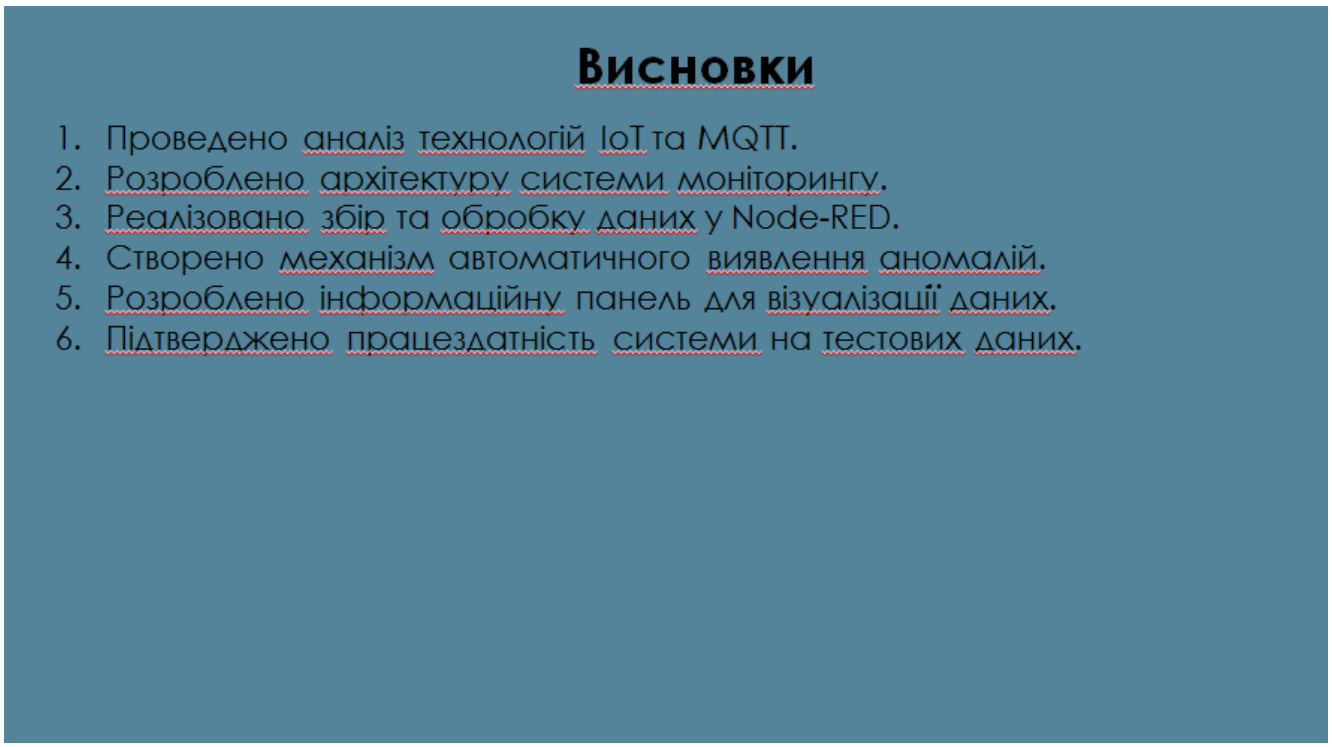

Рисунок Б.13 – Слайд 13

Рисунок Б.14 – Слайд 14