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

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Розподілена система моніторингу навколишнього середовища на основі технології інтернету речей»

КвРКІП. 2302171.23.02.34 ПЗ

Виконав: студент 2 курсу, група КІ2м–23–2 Данило БЕНДІЙ
Підпис Ім'я, прізвище

Керівник д.т.н, професор Василь ЯЦКІВ
Науковий ступінь, вчене звання Підпис Ім'я, прізвище

До захисту допускаю:
Зав. кафедри КІІС, доктор філософії, доцент
Ольга ПАВЛОВА
08 05 2025 р.

Хмельницький, 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-НАУКОВА ПРОГРАМА «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 01 " 09 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Данилу БЕНДЮ
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Розподілена система моніторингу навколишнього середовища на основі технології інтернету речей
Керівник проекту (роботи) Василь ЯЦКІВ, д.т.н., професор
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 08.01.2025 №8

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз відомих методів збору та моніторингу параметрів навколишнього середовища
Моделювання процесу збору та аналізу екологічних даних у розподіленій системі
Методика обробки та аналізу даних у розподіленій IoT–системі

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

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сергій ЛИСЕНКО, професор кафедри КІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІС		

7. Дата видачі завдання «_01_»_____09_____2024р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	01.09.2024	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.10.2024	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	01.11.2024	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.12.2024	виконано
5	Робота над науковою статтею	01.02.2025	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2025	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	01.04.2025	виконано
8	Оформлення пояснювальної записки згідно вимог	18.04.2025	виконано
9	Попередній захист ДРМ	29.04.2025	виконано
10	Захист ДРМ на засіданні ЕК	До 15.05.2025	

Студент _____ Данило БЕНДІЙ
Підпис Ім'я, прізвище

Керівник роботи _____ Василь ЯЦКІВ
Підпис Ім'я, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: «Розподілена система моніторингу навколишнього середовища на основі технології інтернету речей»

Автор роботи: Студент групи КІ2м–23–2 Бендій Данило Михайлович

Керівник роботи: доктор технічних наук, доцент ЯЦКІВ Василь Васильович

Пояснювальна записка: 78с., 18 рис., 3 табл., 3 дод., 82 джерела.

СЕРВЕРНА ЧАСТИНА, АРХІТЕКТУРА, МОНІТОРИНГ, АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ, СЕНСОРИ.

Об’єктом дослідження є процеси та технології моніторингу параметрів навколишнього середовища за допомогою розподілених IoT–систем.

Предметом дослідження є методи та засоби збору, передачі, зберігання, аналізу екологічних даних у розподіленій системі моніторингу на основі технологій JavaScript, Node.js, ESP32 та хмарних сервісів.

Метою кваліфікаційної роботи магістра є – розробка розподіленої системи моніторингу навколишнього середовища на основі технологій Інтернету речей (IoT) із використанням JavaScript, Node.js, ESP32 та хмарних сервісів для збору, передачі, зберігання, аналізу та візуалізації екологічних даних.

Для розв’язання поставлених задач використовувалися методи

1. Аналіз та узагальнення – вивчення існуючих методів і технологій моніторингу навколишнього середовища, розгляд сучасних IoT–рішень та їх ефективності.
2. Методи моделювання – розробка архітектури системи, визначення взаємодії між компонентами (сенсори, сервер, хмарне сховище, клієнтська частина).
3. Експериментальні методи – тестування роботи сенсорів, перевірка точності вимірювань і ефективності передачі даних у реальних умовах.
4. Методи обробки даних – алгоритми фільтрації, нормалізації та аналізу екологічних даних, визначення аномальних показників.

5. Методи програмної інженерії – розробка серверної частини (Node.js, Express), клієнтської частини (веб–інтерфейс), інтеграція з базами даних та хмарними сервісами.

6. Методи візуалізації даних – представлення отриманої інформації у вигляді графіків, таблиць, дашбордів для зручного моніторингу.

Наукова новизна отриманих результатів:

− набув подальшого розвитку метод інтеграції багатосенсорних IoT-систем для моніторингу якості повітря в режимі реального часу, що дозволяє підвищити точність та оперативність виявлення екологічних загроз;

− набула подальшого розвитку інформаційна технологія збору, обробки та візуалізації екологічних даних з використанням мікроконтролера ESP32 та хмарних платформ, що забезпечує масштабованість та доступність системи для широкого кола користувачів.

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення для модульної IoT-системи моніторингу довкілля, з можливістю передачі даних на хмарні сервіси для подальшого аналізу та візуалізації..

Практична значимість отриманих результатів полягає у тому що розроблена система являється доступнішою та енергоефективнішою IoT-системою, за існуючі аналоги. Та може бути використана в різних сферах, включаючи освіту, охорону здоров'я, промисловість та міське планування. Розроблена система дозволяє оперативно виявляти перевищення допустимих рівнів забруднення повітря, що сприяє своєчасному прийняттю заходів для захисту здоров'я населення та навколишнього середовища.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ПЗ – програмне забезпечення

БД – база даних

СМНС – система моніторингу навколишнього середовища

АЗ – апаратне забезпечення

IoT – інтернет речей

API – Application Programming Interface

ККД – коефіцієнт корисної дії

REST - Representational State Transfer

ВСТУП

Термін «Інтернет речей» (IoT) означає концепцію взаємодії розумних пристроїв, оснащених сенсорами, які здатні збирати, аналізувати та обмінюватися даними через Інтернет. Ці пристрої можуть охоплювати широкий спектр застосувань – від побутових приладів, таких як розумні термостати та системи освітлення, до промислових сенсорів і складних екологічних моніторингових систем.

Роль IoT у сучасному світі постійно зростає, ці технологія сприяють автоматизації та покращенню якості життя.

Розумні пристрої спрощують повсякденні процеси, забезпечують комфорт та оптимальні умови проживання. Наприклад, інтелектуальні кліматичні системи можуть автоматично регулювати температуру у приміщенні відповідно до заданих параметрів.

Збір та аналіз даних із сенсорів дозволяє покращити виробництво, мінімізувати витрати та прогнозувати потенційні несправності обладнання та оптимізувати витрати.

IoT застосовується для створення систем відеоспостереження, пожежної безпеки, моніторингу витоків газу або небезпечних речовин, що сприяє запобіганню аварій та небезпечних ситуацій.

Інтеграція IoT у міську інфраструктуру дає змогу ефективніше використовувати ресурси, контролювати енергоспоживання, моніторити якість повітря та води, а також оптимізувати транспортні потоки.

Завдяки IoT стало можливим створення портативних пристроїв для дистанційного моніторингу стану здоров'я, що особливо актуально для пацієнтів із хронічними захворюваннями.

На даний момент, з розвитком промисловості та урбанізації, зростає потреба у безперервному спостереженні за параметрами середовища, такими як температура, вологість, якість повітря тощо. Використання IoT дозволяє автоматизувати процес збору, передачі та аналізу даних у режимі реального часу,

що є важливим для екологічного моніторингу, розумних міст та промислових застосувань.

Одним із найбільш перспективних напрямків використання IoT є моніторинг стану навколишнього середовища. Завдяки мережі сенсорів, встановлених у різних регіонах, можна отримувати достовірну інформацію про рівень забруднення повітря, води, ґрунту та інших екологічно важливих параметрів. Наприклад, сенсори якості повітря можуть вимірювати концентрацію оксидів азоту, сірки, вуглеводнів, а також рівень радіаційного фону, що особливо важливо поблизу промислових об'єктів та атомних станцій.

Для аналізу якості води використовуються сенсори, що визначають рівень забруднення хімічними речовинами, важкими металами або небезпечними мікроорганізмами. Отримані дані можуть передаватися у реальному часі на сервер для подальшої обробки та аналізу, що дає змогу своєчасно реагувати на екологічні загрози.

Система моніторингу навколишнього середовища (СМНС) – це комплексна інформаційно–технічна система, що забезпечує збір, аналіз та візуалізацію екологічних даних у режимі реального часу. Основною метою таких систем є своєчасне виявлення негативних змін у довкіллі та розробка заходів для їх усунення.

До складу СМНС входять різні типи сенсорів, які вимірюють показники якості повітря, рівень радіації, шуму, вологості, температури, а також параметри забруднення води та ґрунту. Дані з сенсорів передаються до центральної системи збору інформації, де вони проходять обробку та візуалізацію за допомогою спеціалізованого програмного забезпечення.

СМНС є важливим інструментом для екологічного управління, оскільки дають змогу контролювати стан навколишнього середовища, прогнозувати екологічні ризики та розробляти ефективні заходи з охорони природи. Вони також можуть інтегруватися з урядовими програмами та мобільними додатками для інформування населення про екологічну ситуацію в конкретному регіоні.

Метою кваліфікаційної роботи магістра є розробка програмно–апаратного комплексу для моніторингу екологічних параметрів із використанням JavaScript, Node.js, Arduino та хмарних технологій.

Практична цінність отриманих результатів. В результаті виконаного наукового дослідження розроблена функціональна IoT–система, що дозволяє здійснювати моніторинг параметрів середовища, аналіз отриманих даних та автоматичне оповіщення користувачів про критичні відхилення.

У даній роботі викладено вимоги до методології розробки програмно–апаратних комплексів для екологічного моніторингу, принципи їх побудови та інтеграції з хмарними сервісами.

Для розв’язання поставлених задач використовуються основні положення системного аналізу, розподілених обчислень, комп’ютерних мереж, IoT–технологій та веб–програмування.

За темою кваліфікаційної роботи опубліковано одну публікацію у Збірнику наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2024». (Хмельницький – 2024. – С. 46-47).

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПОДІЛЕНОЇ СИСТЕМИ МОНІТОРИНГУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Розподілена СМНС, що базується на технологіях Інтернету речей (IoT), охоплює широкий перелік завдань, пов’язаних зі збиранням, збереженням, аналізом та обробкою даних. Одним із найважливіших аспектів її роботи є організація механізмів збору інформації від сенсорів та подальшої передачі цих даних на сервер для їх подальшої обробки [8, 11, 23].

Не менш важливу роль відіграє система збереження інформації, яка повинна забезпечувати накопичення значних обсягів даних, отриманих із сенсорів довкілля. Для цього можуть використовуватися як реляційні, так і нереляційні бази даних, а також технології хмарного зберігання, які дають змогу ефективно керувати доступом до інформації та обробляти великі масиви даних [13, 25, 33].

Ще одним критично важливим етапом є обробка та аналіз отриманої інформації. Для цього реалізовуються алгоритми, що допомагають розпізнавати аномальні показники, передбачати зміни та виявляти закономірності у коливаннях екологічних параметрів [30, 38].

Візуалізація даних є невіддільною частиною системи, оскільки вона дозволяє користувачам у зручному форматі переглядати отриману інформацію. Для цього застосовуються інтерактивні веб–інтерфейси, мобільні застосунки або комплексні веб–платформи, які надають доступ до актуальних даних. Використання картографічних сервісів, графіків та динамічних діаграм допомагає швидко аналізувати стан навколишнього середовища в режимі реального часу [27, 32, 44].

Дослідження предметної області дозволяє виявити основні виклики, які виникають при розробці та впровадженні системи моніторингу екологічного стану. Одним із таких ключових завдань є налагодження ефективної комунікації між мікроконтролерами, сенсорами та серверною частиною. Це вимагає створення спеціальних протоколів передачі даних та механізмів зв’язку. Для збору та передавання даних застосовуються різні технології IoT, серед яких:

1. Wi-Fi – забезпечує підключення сенсорів до локальної мережі та передавання даних через маршрутизатор на сервер [6, 24].

2. Bluetooth – дозволяє сенсорам передавати дані через смартфони або інші пристрої, які, своєю чергою, можуть передавати їх через Wi-Fi чи мобільний інтернет [3, 19].

3. LoRaWAN – забезпечує передавання інформації на значні відстані, що є особливо корисним для віддалених та сільських територій [7, 39].

4. NB–IoT (Narrowband IoT) – спеціалізована технологія мобільного зв'язку, яка розрахована на роботу з пристроями IoT, забезпечуючи стабільну передачу даних [1, 35].

Оскільки система накопичує великі обсяги інформації від сенсорів, необхідно застосовувати ефективні методи її обробки та зберігання. У цьому допомагають такі рішення, як розподілені бази даних (MongoDB, Cassandra, HBase), які дозволяють керувати неструктурованими даними та масштабувати систему за потреби. Додатково використовуються методи машинного навчання, які допомагають ідентифікувати закономірності в зібраних даних, а також техніки кешування та оптимізації запитів, що сприяють швидкому доступу до інформації [10, 40].

Щоб користувачі могли легко взаємодіяти з системою та швидко отримувати потрібну інформацію, важливо розробити інтуїтивно зрозумілі веб–інтерфейси та мобільні застосунки. Вони дозволяють у режимі реального часу відстежувати зміни параметрів довкілля та приймати відповідні рішення [2, 14, 21].

Для розробки веб–інтерфейсів застосовуються такі технології:

1. HTML, CSS, JavaScript –це базові технології для створення веб–сторінок [18].

2. React, Angular, Vue.js – це сучасні фреймворки для побудови складних веб–застосунків з інтерактивними елементами [22].

3. React Native, Flutter, Xamarin – це фреймворки для створення кросплатформних додатків, які працюють на обох операційних системах [36].

Щоб користувачі отримували актуальну інформацію без необхідності оновлювати сторінку або перезапускати застосунок, застосовуються технології WebSocket та Server–Sent Events, які забезпечують миттєву передачу даних у реальному часі [12, 31].

Завдяки впровадженню цих технологій можливо створити інтуїтивно зрозумілі та доступні інтерфейси для моніторингу екологічних показників, що дозволяє користувачам швидко реагувати на зміни у довкіллі. Це сприяє прийняттю обґрунтованих рішень щодо охорони природи, підвищує рівень обізнаності про екологічні проблеми та допомагає зберігати природні ресурси для майбутніх поколінь [4].

1.2 Пристрої для вимірювання параметрів навколишнього середовища

Розробка розподіленої СМНС на основі технологій Інтернету речей (IoT) значною мірою залежить від правильного вибору та належного налаштування пристроїв, які безпосередньо здійснюють вимірювання параметрів середовища [5, 15]. Під «пристроями для вимірювання» зазвичай розуміють АЗ (сенсори, контролери, модулі зв’язку, допоміжні компоненти), яке здатне фіксувати певні фізичні або хімічні величини та передавати зібрані дані для подальшої обробки й аналізу. Ці пристрої можуть працювати як автономно (за допомогою батарейного живлення або сонячних панелей), так і в складі більшої стаціонарної мережі з підключенням до електромережі та інтернету. У цьому пункті розглянемо основні аспекти побудови та використання таких пристроїв, їхні ключові характеристики та підходи до інтеграції в розподілені системи.

У більшості проєктів із моніторингу навколишнього середовища використовують мікроконтролери або одноплатні комп’ютери, які виконують функцію «мозку» системи. Серед найпоширеніших рішень – Arduino, ESP32, Raspberry Pi та STM32.

Arduino – це популярна апаратна платформа з відкритим кодом, призначена для створення електронних проєктів. Вона містить мікроконтролер, який можна

програмувати для зчитування даних із сенсорів, керування пристроями та обміну інформацією з іншими системами. Завдяки простоті використання, Arduino активно застосовується у навчанні, прототипуванні та розробці IoT-проєктів [28].

У ESP32 є вбудовані Wi-Fi та Bluetooth, що робить його ідеальним для бездротових IoT-рішень, а також потужніший процесор у порівнянні з типовими Arduino [29].

Raspberry Pi являє собою повноцінний одноплатний комп'ютер із власною операційною системою (Linux), здатний обробляти більші обсяги даних і забезпечувати складніші обчислення чи навіть роботу з базами даних безпосередньо на пристрої [9].

Тоді як STM32 - це серія мікроконтролерів із низьким енергоспоживанням, що підходить для автономних пристроїв, які живляться від батарей [17].

На вибір сенсорів впливають вимоги до точності, діапазону вимірювання, умов експлуатації (температура, вологість, пил, вібрації). У випадках, коли пристрій працює у складних умовах (наприклад, високий рівень вологості чи сильне забруднення), необхідно передбачити додаткові заходи захисту: герметичний корпус, фільтри, стабілізатори живлення тощо. Від сенсорів очікується: точність і стабільність вимірювань упродовж тривалого часу; низьке енергоспоживання, особливо якщо пристрій працює на батареї; зручний інтерфейс підключення (I^2C, SPI, UART), що полегшує інтеграцію з мікроконтролером; можливість калібрування та компенсації похибок у різних умовах [2].

У розподілених системах моніторингу необхідно передавати дані з віддалених вузлів до центрального сервера чи хмарного сховища. Тому пристрої оснащують модулями бездротового зв'язку, до прикладу:

1. Wi-Fi (ESP8266, ESP32) зручний для міських умов, де є доступ до бездротових мереж.
2. LoRaWAN використовується для передачі даних на великі відстані за низької швидкості, придатний для сільської місцевості чи промислових зон.
3. NB–IoT – це стандарт мобільного зв'язку, розроблений спеціально для IoT-пристроїв із низьким енергоспоживанням та широким покриттям.

4. Bluetooth Low Energy використовується переважно для зв’язку сенсорів зі смартфонами, коли потрібен обмін даними на близькій відстані.

5. Ethernet–модулі (наприклад, W5100, ENC28J60) використовуються для стаціонарних установок із кабельним підключенням до мережі.

Пристрої, розташовані в польових умовах або важкодоступних місцях, часто живляться від акумуляторів чи сонячних панелей [34]. Тому ключовим завданням є оптимізація споживання енергії, для цього можливо вжити такі заходи як перехід мікроконтролера в сплячий режим (deep sleep), за відсутності потреби у вимірюваннях, застосування DC–DC перетворювачів з високим ККД, обмеження частоти передавання даних, якщо параметри довкілля не змінюються швидко, а також використання багатоступеневих схем живлення, де найенерговитратніші елементи (наприклад, радіомодулі) вмикаються тільки на час передавання даних.

У випадку моніторингу температури, вологості чи якості повітря під відкритим небом або в промислових приміщеннях слід враховувати: герметичність корпусу (ступінь захисту IP65, IP67 тощо), застосування волого- та пилозахисних фільтрів для сенсорів, аби уникнути механічних пошкоджень або засмічення, антивандальні корпуси чи кріплення, якщо пристрої встановлюють у місцях з імовірністю фізичного втручання.

Для забезпечення точності вимірювань необхідно проводити первинне калібрування за допомогою еталонних приладів, регулярну перевірку або рекалібрування, особливо для сенсорів газу, які з часом можуть втрачати точність через забруднення або зношення а також врахування температурної компенсації, якщо сенсор залежить від змін температури (наприклад, сенсори тиску) [26, 42].

Часто сенсори видають «сирі» значення, які потребують корекції, фільтрації та перетворення в зручні одиниці вимірювання. Тому мікроконтролер може здійснювати цифрову фільтрацію (наприклад, фільтр середнього чи медіанного значення); виконувати згладжування даних для уникнення стрибків у показниках; перевіряти поріг відхилень для виявлення аномалій (різкий ріст рівня CO_2 чи викидів пилу).

Типовим сценарієм використання може стати встановлення пристроїв для вимірювання параметрів середовища у полях або теплицях, щоб контролювати вологість ґрунту, температуру, рівень освітлення та інші чинники. Зібрані дані допомагають оптимізувати полив, регулювати мікроклімат і знижувати витрати ресурсів.

У містах встановлюють станції з контролю якості повітря, шумового забруднення та інших показників (наприклад, рівень пилу PM2.5). Такі станції можуть розташовуватися на опорах освітлення, дахах будівель чи навіть на громадському транспорті, забезпечуючи динамічний збір інформації в реальному часі.

Сенсори температури, вологості та CO_2 дозволяють підтримувати комфортні умови всередині приміщень і забезпечують економію енергії (наприклад, автоматичне керування системами вентиляції чи кондиціонування).

У місцях, де немає розвинутої інфраструктури, встановлюють енергоефективні пристрої з використанням LoRaWAN або NB–IoT, що дає змогу тривалий час автономно відстежувати рівень води у водоймах, стан ґрунту, виявляти лісові пожежі тощо.

Що стосується інтеграції пристроїв у розподілену СМНС – пристрої періодично або за подією (наприклад, значне відхилення від норми) надсилають інформацію на центральний сервер або хмарний сервіс. Для цього застосовують протоколи MQTT, HTTP/HTTPS, CoAP тощо.

У хмарному середовищі або на локальному сервері дані можуть проходити додаткову фільтрацію, агрегування та аналіз із використанням технологій Big Data та машинного навчання. Це дозволяє швидко виявляти аномалії й формувати попередження для користувачів.

Кінцеві користувачі отримують доступ до результатів вимірювань через веб–інтерфейси або мобільні застосунки. У випадку критичних показників система може надсилати push–повідомлення, SMS чи email із попередженнями.

Оскільки розподілені системи часто розгортають поступово, важливо, щоб до мережі можна було легко додавати нові пристрої або змінювати конфігурацію без суттєвого впливу на інші вузли.

Таким чином, пристрої для вимірювання параметрів навколишнього середовища – це фундаментальна ланка у розподілених системах моніторингу на базі IoT. Їхній вибір і налаштування залежать від конкретних завдань: чи то контроль якості повітря в міській місцевості, чи відстеження вологи в теплиці, чи аналіз рівня шуму в промисловій зоні. Критично важливо враховувати умови експлуатації, забезпечувати захист від зовнішніх впливів, ретельно калібрувати сенсори й оптимізувати енерговитрати. Лише за умови комплексного підходу до розробки апаратного забезпечення та правильного вибору методів передачі даних можна створити надійну та масштабовану систему, яка буде корисною як для локальних потреб (наприклад, у «розумних будинках»), так і для великих мережевих рішень (моніторинг міст, сільськогосподарських територій, промислових об’єктів) [50, 52].

1.3 Порівняльний аналіз переваг та недоліків існуючих рішень

Система AirVisual від IAQair є передовим рішенням для моніторингу якості повітря в реальному часі. Вона працює за допомогою сенсорів та інтегрованих технологій для збору та аналізу даних про стан довкілля.

AirVisual використовує високоточні сенсори для вимірювання різних параметрів якості повітря, таких як: PM2.5 (дрібнодисперсні частинки) – показник рівня забруднення повітря, PM10 (більші частинки) – частинки більшого розміру. CO2 (вуглекислий газ) – важливий параметр для визначення рівня вентиляції та свіжості повітря. температура та вологість – для оцінки загального клімату в приміщенні або на відкритому повітрі. Летючі органічні сполуки (VOC) – допомагають визначити наявність шкідливих хімічних сполук в повітрі.

Система AirVisual застосовує різні технології для обробки та аналізу даних, щоб надати точну інформацію про стан повітря та допомогти користувачам

приймати обґрунтовані рішення. Для визначення рівня забруднення та рекомендацій користувачам використовується аналітика на основі моделей прогнозування, яка враховує різні фактори: метеорологічні умови, географічне розташування та історичні дані. Це дозволяє не тільки фіксувати поточний стан, а й передбачати можливі зміни в якості повітря.

Зібрані сенсорами дані про якість повітря обробляються за допомогою унікальних алгоритмів, які враховують зовнішні фактори, такі як температура та вологість. Це забезпечує точність вимірювань та коректність даних.

IQAir надає хмарну платформу AirVisual, яка дозволяє користувачам отримувати доступ до даних про якість повітря в реальному часі. Платформа забезпечує інтерактивні карти та прогнози, що допомагають користувачам планувати свої дії з урахуванням поточної ситуації з якістю повітря. На Рисунку 1.1 зображений зовнішній вигляд сайту IQAir.

Для зручності користувачів IQAir розробив мобільні додатки AirVisual, які доступні для пристроїв на iOS та Android. Ці додатки надають дані про якість повітря, прогнози та сповіщення, що дозволяє користувачам бути в курсі змін у якості повітря та вживати відповідних заходів. Ці технології та ресурси забезпечують ефективний моніторинг, аналіз та комунікацію даних про якість повітря, сприяючи підвищенню обізнаності та здоров'я користувачів.

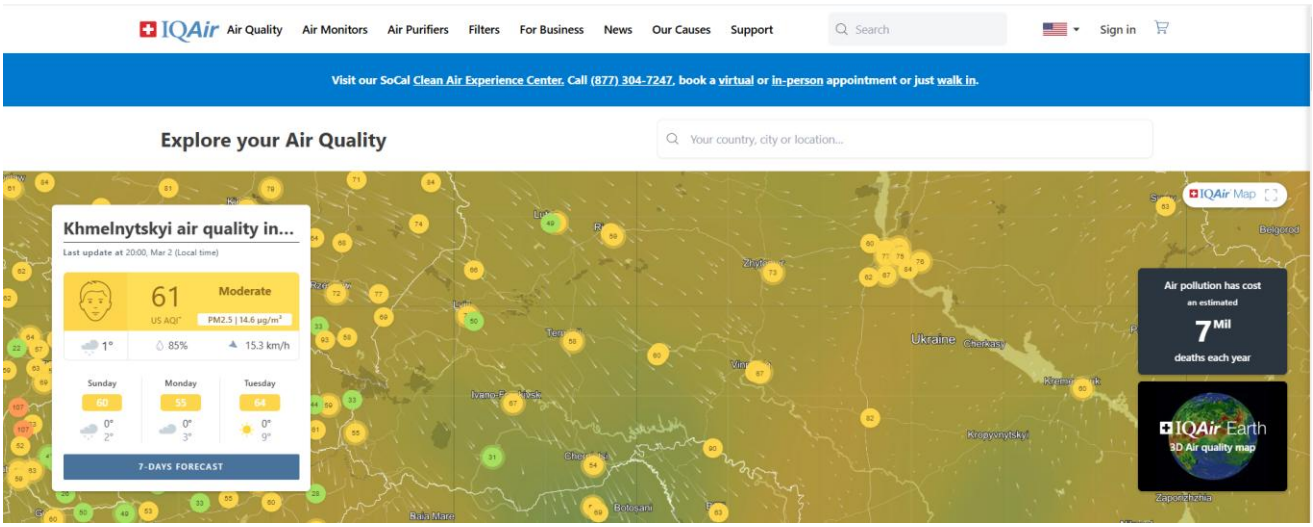


Рисунок 1.1 – Інтерфейс веб–сторінки AirVisual [83]

Перевагою цієї системи є велика база даних щодо якості повітря з різних джерел – це дозволяє отримувати більш точні та перевірені показники, що підвищує достовірність аналізу. Однак одним з найбільших недоліків являється обмежена кількість доступних параметрів навколишнього середовища – додаток переважно фокусується на якості повітря, не враховуючи інші екологічні фактори, такі як рівень води чи ґрунтове забруднення.

Plume Labs: Air Report – система що показує рівень забруднення в реальному часі у вашому регіоні та по всьому світу.

Розроблений Plume Labs сенсор якості повітря (Рисунок 1.2), вимірює тверді частинки (PM10, PM2.5 і PM1), NO2 і VOC. Flow використовує дві різні методики для вимірювання якості повітря: лазерну дифракцію для дрібних часток і вимірювання коливань провідності оксидів металів для NO2 і летких органічних сполук.

Рисунок 1.2 – Сенсор від Plume Labs

Алгоритми системи аналізують попередні показники та метеодані, щоб передбачати зміни якості повітря. Також система має простий та інтуїтивний інтерфейс – навіть користувачі без досвіду роботи з подібними програмами можуть легко зрозуміти інформацію та налаштувати додаток під свої потреби.

Доступність на мобільних пристроях робить можливим отримувати інформацію в режимі реального часу без необхідності використовувати стаціонарний комп'ютер. Вигляд мобільного додатку Plume Labs зображено на Рисунку 1.3.

Однак як і у попередньої системи відсутність інтеграції з іншими показниками навколишнього середовища позбавляє користувача можуливості отримувати комплексну екологічну картину, що зменшує його ефективність у більш широкому контексті.

Також дані можуть відрізнятися за точністю залежно від місцевості, оскільки додаток використовує відкриті джерела, у деяких регіонах інформація може бути менш надійною.

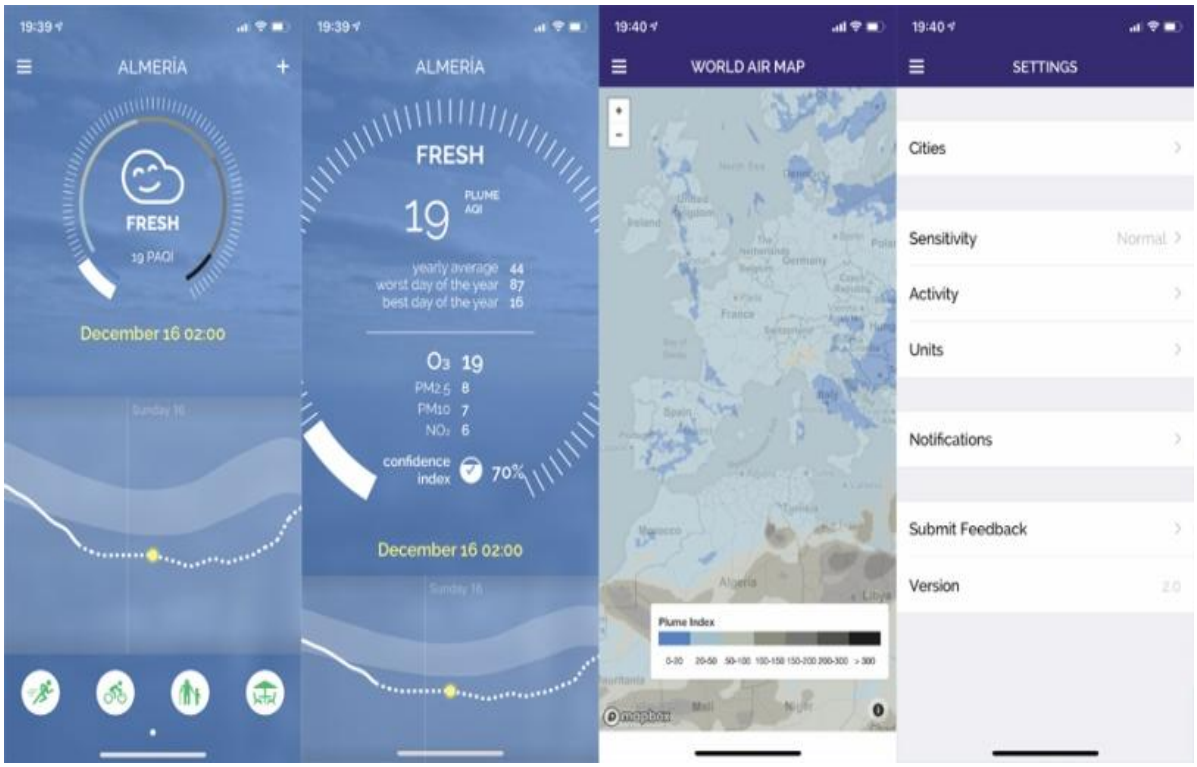


Рисунок 1.3. – Вигляд додатку Plum Labs

Система Smart Greenhouse – є інноваційним рішенням для автоматичного контролю та управління умовами вирощування рослин у теплицях. Її основна мета полягає в створенні оптимальних умов для росту рослин через моніторинг різних параметрів, таких як температура, вологість, освітленість та інші фактори [58, 59].

Роль серверної частини в цій системі є надзвичайно важливою для забезпечення ефективного функціонування та управління тепличними умовами.

З позитивних сторін серверна частина дає можливість автоматизувати безліч аспектів управління теплицею, таких як контроль кліматичних параметрів, системи поливу та освітлення, що дозволяє оптимізувати виробничі процеси та створити кращі умови для росту рослин.

Завдяки серверній частині оператори можуть отримувати актуальні дані про стан теплиці в режимі реального часу, що дає змогу швидко реагувати на зміни та вчасно вирішувати проблеми.

Система дозволяє операторам віддалено управляти параметрами теплиці через веб–інтерфейс або мобільний додаток, що забезпечує зручність і доступність управління навіть з відстані.

Серверна частина дозволяє ефективно використовувати енергію в теплиці, включаючи системи опалення та освітлення тільки за потребою, що зменшує витрати на енергію.

Проте для підтримки системи в належному стані необхідно регулярно оновлювати ПЗ та проводити обслуговування, що може викликати додаткові витрати.

У разі неполадок чи перебоїв у роботі серверної частини система може втратити контроль над умовами в теплиці, що може негативно вплинути на ріст рослин та врожайність.

Як і в будь–яких комп'ютерних системах, можливі програмні помилки або збої, які можуть спричинити непередбачувані проблеми та порушення керування теплицею.

Для забезпечення надійності та ефективності серверної частини системи Smart Greenhouse використовуються різноманітні технології. Багато таких систем побудовані на основі хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform, що дає змогу забезпечити високу гнучкість, масштабованість і доступність, дозволяючи ефективно обробляти дані та управляти системою з будь–якої точки світу.

Для збереження та обробки даних про умови вирощування рослин застосовуються різні типи баз даних, як–от реляційні (MySQL, PostgreSQL) та NoSQL (MongoDB, Cassandra), що дозволяють зберігати і аналізувати великі обсяги інформації про умови в теплиці та за її межами. Використання таких даних дає змогу операторам приймати обґрунтовані рішення щодо управління врожаєм і коригувати процеси для досягнення оптимальних результатів [16].

Деякі серверні частини систем Smart Greenhouse базуються на мікросервісній архітектурі, де різні функціональні блоки представлені окремими сервісами, що взаємодіють між собою. Для аналізу даних і прийняття рішень використовуються потужні аналітичні інструменти та фреймворки, такі як Apache Spark, TensorFlow або SciPy. Ці технології сприяють створенню надійної та ефективної серверної частини системи Smart Greenhouse для оптимального управління тепличними умовами вирощування рослин [62, 63]. Загальна схема роботи системи Smart Greenhouse зображена на Рисунку 1.4.

Рисунок 1.4 – Схематичне зображення системи Smart Greenhouse

У результаті порівняльного аналізу переваг та недоліків існуючих рішень щодо серверної частини систем моніторингу компонентів довкілля видно, що кожна з них має свої унікальні переваги та обмеження. Після здійснення аналізу, можна виділити такі переваги та недоліки у кожної з розглянутих систем моніторингу компонентів довкілля.

AirVisual by IQAir має велику базу даних щодо якості повітря з різних джерел, що забезпечує високу точність та достовірність аналізу.

Проте має обмежену кількість доступних параметрів навколишнього середовища – система зосереджена переважно на якості повітря, не враховуючи інші екологічні фактори.

Plume Labs: Air Report використання штучного інтелекту для прогнозування рівня забруднення повітря, дозволяє завчасно оцінювати можливі зміни.

Однак дані можуть відрізнятися за точністю залежно від місцевості, оскільки додаток використовує відкриті джерела, які не завжди є надійними.

Smart Greenhouse автоматизація процесів керування теплицею, дозволяє оптимізувати кліматичні умови та підвищити ефективність вирощування рослин.

Але існує критична залежність від стабільної роботи серверної частини – у разі її збоїв може бути втрачено контроль над параметрами теплиці, що негативно вплине на врожайність.

Інші реалізації, наприклад на базі платформ Blynk, ThingSpeak, дозволяють будувати прості системи з візуалізацією екологічних параметрів та відправкою сповіщень у мобільний застосунок [60, 81].

1.4 Підходи до вирішення задачі за темою дослідження

Розподілена СМНС на основі технології Інтернету речей (IoT) складається з кількох ключових компонентів: сенсорів для збору даних, мережевої інфраструктури для передачі інформації, серверної частини для обробки та зберігання даних, а також механізмів аналітики й оповіщення. Вибір

архітектурного підходу залежить від вимог до продуктивності, масштабованості та надійності системи [65, 66].

IoT-сенсори та мікроконтролери (ESP8266, ESP32, Raspberry Pi, Arduino) збирають дані про навколишнє середовище (температуру, вологість, якість повітря тощо) та передають їх до обчислювальних вузлів через бездротові технології (Wi–Fi, LoRaWAN, Zigbee, MQTT) [14, 37].

Передача даних в IoT-системах здійснюється різними способами залежно від умов використання. Наприклад, для стаціонарних сенсорів доцільно використовувати локальні мережі (Wi-Fi, Ethernet), тоді як мобільні або віддалені вузли ефективно працюють через мобільний зв'язок (3G/4G/5G). Протоколи передачі даних MQTT, CoAP, HTTP мають різні переваги, і вибір між ними залежить від потрібної швидкості, надійності, легкості реалізації та енергоспоживання пристрою [66, 68].

Серверна частина реалізується через хмарні сервіси або локальні (edge-computing) вузли. Хмарна інфраструктура (наприклад, Amazon AWS, Microsoft Azure, Google Cloud Platform) забезпечує високу масштабованість та надійне зберігання, проте іноді затримки у передачі даних змушують проєктувальників використовувати локальні обчислення, де обробка відбувається ближче до місця збору даних. Це дозволяє скоротити час реакції системи на критичні зміни середовища [69, 64].

Зберігання даних здійснюється в базах даних, які поділяються на реляційні (MySQL, PostgreSQL) та нереляційні (NoSQL, зокрема MongoDB, InfluxDB). NoSQL-бази, зокрема time-series бази, дозволяє ефективно працювати з великими обсягами даних, які надходять у режимі реального часу, що є важливим для екологічного моніторингу[17].

Візуалізація може здійснюватися за допомогою панелей моніторингу, що відображають показники в реальному часі у вигляді графіків або табличних значень. Усе частіше розробники використовують MongoDB у поєднанні з інтерфейсами REST API або вебсервісами для побудови гнучких аналітичних панелей [67]. Впровадження системи оповіщень через SMS, push–сповіщення,

email або автоматичні сценарії для реагування на небезпечні зміни параметрів довкілля.

Пристрої на базі ESP32 довели свою ефективність у проектах реального часу завдяки своїй енергоефективності, наявності вбудованих модулів Wi-Fi/Bluetooth, а також сумісності з широким спектром сенсорів. У проєктах екологічного моніторингу ці контролери використовуються для контролю температури, вологості, тиску тощо, передаючи дані на сервер для обробки [19].

Окремі системи розробляються як повністю готові рішення для моніторингу екології, в яких ESP32 використовується спільно з хмарними платформами для збору та виводу даних. Це дозволяє забезпечити як автономність роботи сенсорних вузлів, так і централізоване зберігання даних на сервері [20].

Публікації демонструють, що IoT-системи, які використовують ESP32 та хмарне з'єднання, дозволяють реалізувати повноцінну архітектуру екологічного моніторингу з можливістю віддаленого керування та аналізу [21].

MongoDB використовується в екосистемі IoT не лише для зберігання, але і як частина системи аналітики. Це рішення легко масштабується і підтримує агрегаційні функції для побудови звітів і реагування на події в реальному часі [22].

Нарешті, сучасні роботи, присвячені розгортанню та тестуванню систем екологічного моніторингу, доводять переваги використання відкритих стандартів, гнучких IoT-протоколів, хмарних платформ і модульних підходів для створення систем, здатних працювати в динамічному середовищі та масштабуватись із мінімальними затратами [65, 66].

1.5 Постановка задачі

Сучасні екологічні виклики, зокрема забруднення повітря, зміни клімату та вичерпність природних ресурсів, вимагають ефективних інструментів для моніторингу стану навколишнього середовища. Використання технології Інтернету речей (IoT) у розподілених системах моніторингу дозволяє забезпечити автоматизований збір, аналіз та візуалізацію екологічних параметрів у реальному

часі. Це сприяє прийняттю своєчасних рішень щодо попередження небезпечних ситуацій та мінімізації негативного впливу на довкілля.

Метою роботи є розробка розподіленої системи моніторингу навколишнього середовища на основі IoT, яка забезпечить ефективний збір, обробку, аналіз та візуалізацію даних про екологічні показники, а також механізми оповіщення про критичні зміни параметрів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Аналіз сучасних технологій IoT для моніторингу навколишнього середовища, вибір оптимальних апаратних і програмних рішень.
2. Розробка архітектури розподіленої системи, що включає сенсори, мережеві протоколи, серверну частину, хмарне сховище та клієнтські інтерфейси.
3. Реалізація модулів збору даних за допомогою мікроконтролерів (ESP32) та відповідних комунікаційних технологій (Wi–Fi, MQTT).
4. Організація зберігання та обробки даних із використанням реляційних (PostgreSQL, MySQL) або NoSQL (MongoDB, InfluxDB) баз даних.
5. Розробка механізму оповіщення користувачів про критичні відхилення екологічних показників через push–сповіщення, email або SMS.
6. Створення веб–інтерфейсу для візуалізації даних у режимі реального часу.

Об’єкт дослідження – розподілені системи моніторингу навколишнього середовища на основі IoT.

Предмет дослідження – методи та алгоритми збору, обробки, аналізу, зберігання та візуалізації даних про стан довкілля в розподілених IoT-системах.

У процесі розробки та дослідження системи використовуються такі методи: методи аналізу та синтезу для визначення оптимальної архітектури розподіленої системи; технології IoT для розгортання сенсорної мережі та збору даних; хмарні обчислення та бази даних для ефективного управління великими масивами інформації; засоби веб–розробки для створення зручного інтерфейсу користувача.

Результатом роботи стане розроблена та протестована розподілена СМНС, що забезпечить:

– автоматизований збір даних із сенсорів;

– оперативну передачу та обробку інформації;

– аналіз тенденцій та виявлення аномалій;

– інтегровану систему сповіщення;

– інтуїтивно зрозумілий веб–інтерфейс для перегляду результатів моніторингу.

Отримані результати можуть бути використані в системах екологічного моніторингу міст, промислових об'єктів, розумних будинків та сільськогосподарських комплексів.

Завдяки доступності системи її можна буде легко використовувати в домашніх умовах чи в освітніх закладах.

1.6 Висновки

У першому розділі кваліфікаційної роботи здійснено ґрунтовне дослідження предметної області розподілених систем моніторингу навколишнього середовища, що базуються на технологіях Інтернету речей. Розглянуто сучасні підходи до організації збору, передавання, зберігання, обробки та візуалізації екологічних даних. Проаналізовано особливості взаємодії між апаратною частиною (сенсори, мікроконтролери), мережевими протоколами (Wi-Fi, LoRaWAN, MQTT), серверною інфраструктурою (хмарні сервіси, бази даних) та клієнтськими інтерфейсами.

Було встановлено, що ключовими вимогами до таких систем є енергоефективність, масштабованість, точність вимірювань, безперервна робота в реальному часі та зручна візуалізація даних для кінцевих користувачів. Проаналізовано можливості використання платформ на базі ESP32, MongoDB, а також технології для побудови веб-інтерфейсів на основі React та Chart.js.

Також проведено порівняльний аналіз існуючих рішень – AirVisual, Plume Labs, Smart Greenhouse – виявлено їхні сильні сторони та обмеження. Усі вони демонструють приклади успішної реалізації концепцій розподіленого моніторингу,

але мають обмеження щодо набору параметрів, складності масштабування або залежності від інфраструктури.

У результаті аналізу сформульовано технічні та функціональні вимоги до розробки власної IoT-системи моніторингу, яка має забезпечити гнучку архітектуру, інтеграцію з хмарними платформами, високу точність збору екологічних показників та доступний інтерфейс для візуалізації й керування. Це створює передумови для реалізації ефективного, доступного та масштабованого рішення в наступних розділах роботи.

2 ВИБІР ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ РОЗПОДІЛЕНОЇ СИСТЕМИ МОНІТОРИНГУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

2.1 Важливість вибору оптимальних засобів розробки для забезпечення ефективності, надійності та масштабованості розподіленої системи.

У сучасному світі технологій розробка розподілених систем моніторингу навколишнього середовища є важливим завданням, яке вимагає ретельного підходу до вибору відповідних засобів розробки. В умовах стрімкого зростання обсягів даних, що генеруються сенсорами в Інтернеті речей (IoT), ключовими аспектами проєктування такої системи є ефективність, надійність та масштабованість. Відповідний вибір технологій та архітектурних рішень безпосередньо впливає на якість роботи системи, її здатність обробляти великі потоки інформації та забезпечувати стабільне функціонування в різних умовах експлуатації [24].

Ефективність системи визначає її здатність швидко та коректно обробляти дані, що надходять від численних сенсорів. Це важливо для оперативного аналізу показників навколишнього середовища та своєчасного реагування на можливі загрози. Використання оптимальних програмних засобів дозволяє мінімізувати затримки при передачі та обробці даних, що критично для систем реального часу [25].

Надійність є ще одним важливим фактором, адже розподілені системи працюють у середовищах, де можливі збої зв’язку, втрати даних або вихід з ладу окремих вузлів. Для забезпечення безперервності роботи необхідно застосовувати технології, які передбачають механізми самовідновлення, дублювання даних та автоматичне перенаправлення трафіку в разі відмови окремих компонентів системи [26].

Масштабованість системи відіграє вирішальну роль у її довготривалому розвитку. СМНС може розширюватися, інтегруючи нові сенсори, збільшуючи географічне покриття або обробляючи дедалі більші обсяги інформації. Тому важливо, щоб обрані засоби розробки підтримували гнучку адаптацію до змінних

умов без значних втрат продуктивності чи необхідності повного перероблення архітектури [27].

Вибір відповідних технологій та засобів розробки є ключовим фактором успішного створення розподіленої системи моніторингу. Недостатньо лише обрати популярні або найновіші технології – необхідно аналізувати їхню відповідність специфічним вимогам проєкту, враховувати особливості середовища експлуатації та майбутні потреби розширення системи. Саме оптимальний вибір засобів розробки стає запорукою ефективної, надійної та масштабованої роботи системи моніторингу навколишнього середовища в умовах Інтернету речей [28].

2.2 Функціональні вимоги розподіленої системи моніторингу навколишнього середовища

Функціональні вимоги розподіленої системи моніторингу навколишнього середовища визначають, як повинна працювати система для забезпечення коректного збору, обробки, аналізу та зберігання даних. Вони охоплюють аспекти отримання даних з сенсорів, перевірки їхньої достовірності, збереження інформації в базах даних, виконання запитів, фільтрації результатів та адміністрування системи. Застосування відповідних функціональних вимог забезпечує надійність роботи системи та її здатність своєчасно реагувати на зміни параметрів довкілля.

Основною функцією системи є отримання екологічних даних від сенсорів, розташованих у різних точках моніторингу. Для цього система повинна підтримувати з’єднання з мікроконтролерами, такими як ESP32 або Raspberry Pi, які збирають інформацію про температуру, вологість, рівень забруднення повітря та інші показники. Оскільки сенсори можуть працювати в різних умовах і на різних відстанях від центрального сервера, важливо забезпечити підтримку декількох протоколів зв’язку, таких як HTTP, MQTT або CoAP. Це дозволить вибрати оптимальний метод передачі даних залежно від специфіки середовища та обсягу інформації [38].

Після отримання даних від сенсорів система повинна здійснювати їхню перевірку на коректність і достовірність. Наприклад, значення вологості повітря повинно знаходитись у межах від 0% до 100%, а показники температури не можуть перевищувати фізично можливі значення. У разі виявлення некоректних даних система має визначати їх як помилкові та повідомляти про це адміністратору або оператору. Додатково система повинна перевіряти формат отриманих даних, зокрема правильність структури JSON–об'єктів, які надсилаються з мікроконтролерів, та відповідність формату часу і дат. Це забезпечить безпомилкову обробку інформації та мінімізує ризики втрати даних.

Збереження отриманих даних є важливим етапом функціонування системи. Вибір відповідної бази даних залежить від обсягу інформації та потреб у швидкому доступі до неї. Для зберігання великих масивів даних, які накопичуються протягом тривалого часу, доцільно використовувати реляційні бази даних, такі як MySQL або PostgreSQL. Якщо ж система орієнтована на зберігання великих потоків неструктурованих даних у реальному часі, доцільно застосовувати NoSQL–рішення, такі як MongoDB або InfluxDB. Крім цього, система повинна забезпечувати можливість збереження історичних даних для подальшого аналізу та формування звітів.

Щоб забезпечити зручний доступ до даних, система повинна підтримувати можливість виконання запитів за різними параметрами. Користувачі повинні мати змогу отримувати інформацію, вказуючи конкретні часові проміжки, географічні координати або тип сенсора. Запити можуть виконуватись як через веб–інтерфейс, так і через API, що забезпечує інтеграцію з іншими системами. Крім того, система повинна підтримувати параметризацію запитів для отримання вибіркових даних, що дозволить знизити навантаження на сервер та забезпечити швидку відповідь.

Функція фільтрації результатів запитів є важливим інструментом для аналізу отриманих даних. Користувачі повинні мати можливість застосовувати фільтри за різними критеріями, такими як діапазон дат, рівень забруднення або температури. Це значно полегшить аналіз тенденцій і виявлення аномальних значень. Додатково

система може надавати можливість агрегації даних, що дозволить користувачам отримувати усереднені або підсумкові значення для певних періодів часу.

Адміністрування та управління системою є ще однією важливою функцією. Адміністратори повинні мати можливість контролювати роботу серверної частини, перевіряти стан підключених сенсорів та оперативно реагувати на виникнення помилок. Крім того, система повинна надавати інструменти для перегляду логів, моніторингу продуктивності та аналізу подій. Важливим є також впровадження ролей та рівнів доступу для користувачів, що дозволить обмежити доступ до критично важливих функцій.

Загалом, функціональні вимоги до розподіленої системи моніторингу навколишнього середовища охоплюють широкий спектр завдань, від збору та обробки даних до адміністрування та забезпечення надійності роботи. Впровадження цих вимог дозволить створити ефективну та гнучку систему, здатну своєчасно реагувати на зміни довкілля та забезпечувати користувачів актуальною інформацією для прийняття обґрунтованих рішень.

2.3 Нефункціональні вимоги розподіленої системи моніторингу навколишнього середовища

Нефункціональні вимоги до розподіленої системи моніторингу навколишнього середовища відіграють важливу роль у забезпеченні її ефективності, надійності та безпеки. На відміну від функціональних вимог, які визначають, що саме має робити система, нефункціональні вимоги встановлюють критерії якості, яким повинна відповідати система для стабільної та продуктивної роботи.

Продуктивність є одним із ключових аспектів нефункціональних вимог. Система повинна забезпечувати швидкий обмін даними між сенсорами, сервером і користувацьким інтерфейсом, мінімізуючи затримки при обробці запитів. Час відповіді серверної частини має залишатися мінімальним навіть при високому навантаженні, що особливо важливо для роботи в режимі реального часу.

Оптимізація бази даних, використання кешування та розподілених обчислень сприятиме підтримці високої продуктивності [77, 76].

Надійність системи передбачає її стійкість до збоїв та здатність відновлювати роботу після неполадок. Для цього необхідно впровадити механізми резервування даних, дублювання серверів і автоматичного відновлення системи. Моніторинг роботи всіх компонентів системи дозволить своєчасно виявляти та усувати проблеми. Крім того, система повинна мати механізми логування та сповіщення про критичні помилки для швидкого реагування [75, 81].

Масштабованість системи є важливою для забезпечення її ефективної роботи при зростанні кількості сенсорів або обсягу даних. Розподілена архітектура дозволить горизонтально масштабувати систему шляхом додавання нових серверів або обчислювальних вузлів. Крім того, система повинна мати можливість адаптації до змін навантаження шляхом автоматичного балансування трафіку [72, 79].

Доступність системи визначає її здатність працювати безперервно та бути доступною для користувачів у будь–який час. Для цього необхідно забезпечити відмовостійкість на рівні програмного та апаратного забезпечення, використовуючи кластеризацію серверів та географічно розподілені дата–центри. Резервні канали зв’язку та автоматичне перемикання на резервні сервери у разі відмови основного також сприятимуть підвищенню доступності.

Зручність використання системи є ще одним важливим критерієм. Інтуїтивно зрозумілий інтерфейс, що забезпечує легкий доступ до основних функцій, сприятиме ефективній роботі користувачів. Веб–інтерфейс має підтримувати адаптивний дизайн для коректного відображення на різних пристроях, включаючи смартфони, планшети та комп’ютери. Наявність інтерактивних елементів, таких як графіки та карти, дозволить користувачам швидко аналізувати отримані дані.

Енергоефективність також є важливою нефункціональною вимогою, особливо для сенсорів та інших IoT-пристроїв. Система повинна підтримувати оптимізацію енергоспоживання, зокрема використання енергоефективних протоколів зв’язку та алгоритмів обробки даних. Сенсори можуть працювати в

режимі низького енергоспоживання та переходити в сплячий режим при відсутності необхідності передачі даних.

Зрештою, система повинна відповідати вимогам з обслуговування та підтримки. Адміністратори повинні мати можливість швидко діагностувати та усувати неполадки за допомогою інструментів моніторингу та управління. Регулярне резервне копіювання даних і оновлення програмного забезпечення забезпечать довготривалу стабільну роботу системи [70].

Таким чином, нефункціональні вимоги визначають якість роботи системи моніторингу навколишнього середовища, забезпечуючи її продуктивність, надійність, безпеку, масштабованість, доступність та зручність у використанні. Виконання цих вимог сприятиме ефективному функціонуванню системи в умовах зростаючих навантажень та динамічно змінюваних умов експлуатації.

2.4 Опис стеку технологій

2.4.1 Node.js – середовище виконання JavaScript

Node.js – це кросплатформене, відкрите середовище виконання JavaScript, яке підтримує роботу на різних операційних системах, зокрема Windows, Linux, Unix, macOS тощо. Воно функціонує на основі рушія V8, що дає змогу виконувати JavaScript–код поза межами веб–браузера [47, 55].

Node.js надає розробникам можливість використовувати JavaScript для створення інструментів командного рядка, а також для серверного програмування. Завдяки цьому JavaScript може використовуватися для динамічної генерації вмісту веб–сторінок перед їх передачею у браузер користувача. Таким чином, Node.js сприяє реалізації концепції «JavaScript everywhere», що забезпечує уніфікацію розробки веб–застосунків шляхом використання єдиної мови програмування як на серверному, так і на клієнтському боці.

Нижче наведено Таблицю 2.1 переваги та недоліки використанні Node.js

Таблиця 2.1 – Переваги та недоліки Node.js

Переваги	Недоліки
Node.js працює на основі високопродуктивного JavaScript двигуна V8, який забезпечує швидке виконання коду	Оскільки Node.js працює в однопотоковому середовищі, це може створювати труднощі при виконанні ресурсомістких обчислювальних завдань. Для вирішення цієї проблеми можна використовувати робочі потоки,хоча їхнє застосування є складнішим порівняно з мовами, що мають вбудовану підтримку багатопоточності.
Асинхронна модель ефективного вводу–виводу в Node.js дає змогу обробляти велику кількість запитів одночасно.	Коли в коді використовується багато зворотних викликів, його читабельність і розуміння можуть значно ускладнитися.
Можливість використання однієї і тієї ж мови програмування (JavaScript) на стороні клієнта та сервера значно спрощує процес розробки.	Динамічна типізація JavaScript ускладнює виявлення помилок, оскільки вони проявляються лише під час виконання, а не на етапі компіляції. Частково цю проблему можна вирішити за допомогою TypeScript.
Асинхронна природа Node.js сприяє легкому масштабуванню додатків завдяки обробці подій і використанню зворотних викликів.	Швидкий розвиток Node.js може спричиняти зміни API, що інколи призводить до несумісності з попередніми версіями, ускладнюючи підтримку довгострокових проєктів.

Node.js – це потужний інструмент для розробки серверної частини, особливо ефективний для створення систем реального часу та обробки великої кількості запитів одночасно. Його асинхронна модель та неблокуючий ввід–вивід роблять його ідеальним вибором для високонавантажених застосунків. Однак важливо враховувати його обмеження, зокрема неефективність у виконанні обчислювально інтенсивних завдань, що може призводити до блокування основного потоку. Крім того, асинхронне програмування в Node.js може ускладнити відстеження помилок і підтримку коду, особливо в масштабних проєктах.

2.4.2 NPM – менеджер пакетів для JavaScript

Однією з ключових складових середовища Node.js є менеджер пакетів NPM (Node Package Manager) – потужний інструмент для роботи з бібліотеками та модулями JavaScript. NPM дозволяє розробникам ефективно керувати залежностями проєкту, пришвидшуючи процес інтеграції сторонніх рішень та спрощуючи підтримку й масштабування програмного забезпечення.

Менеджер NPM постачається разом із Node.js, що забезпечує готовність до роботи з моменту встановлення платформи. Його основне призначення – автоматизація процесів встановлення, оновлення, видалення та публікації пакетів. Це дає змогу уникнути ручного пошуку, завантаження та конфігурування сторонніх бібліотек, що особливо важливо при роботі з великомасштабними проєктами або у складі командної розробки.

Архітектурно NPM побудований на основі централізованого сховища пакетів – npm registry, яке містить сотні тисяч відкритих модулів, доступних для завантаження та використання. Кожен пакет є самостійною одиницею програмного забезпечення, що містить у собі програмний код, документацію, метадані (ім’я, версію, опис, авторів), а також список залежностей, необхідних для коректного функціонування. Для встановлення пакета достатньо скористатися простою командою: npm install <назва_пакету>

У результаті виклику цієї команди NPM надсилає HTTP–запит до сховища, отримує архів із відповідним пакетом, розпаковує його у директорію node_modules та автоматично додає запис до спеціального конфігураційного файлу – package.json. Цей файл виконує роль центрального документа, що описує основні параметри проєкту, його залежності, скрипти для запуску, тестування, збірки тощо. Він забезпечує повну конфігурацію середовища розробки, що дає змогу іншим учасникам проєкту швидко відтворити робоче середовище, виконавши команду npm install.

Управління залежностями відбувається через структуру package.json, у якому залежності поділяються на основні (dependencies) та дев–залежності (devDependencies). Основні потрібні для функціонування програми в продакшн–середовищі, а дев–залежності – виключно для розробки, тестування, компіляції чи лінтингу.

Також варто відзначити, що NPM підтримує семантичне версіонування (semver), що дає змогу чітко контролювати сумісність версій пакетів. Розробник може точно вказати, яку версію пакета слід використовувати, або дозволити автоматичне оновлення в межах сумісних змін.

Завдяки гнучкій структурі, розвиненій інфраструктурі та активній спільноті, NPM став стандартом де–факто для керування модулями у JavaScript–розробці. Він дозволяє централізовано керувати всіма залежностями проєкту, забезпечує повторюваність збірки, підтримує скрипти для автоматизації задач, а також сприяє модульності та повторному використанню коду.

Більш структурована характеристика переваг та недоліків зображена у Таблиці 2.2

Таблиця 2.2 – Переваги та недоліки використання NPM

Переваги	Недоліки
NPM пропонує більше мільйона пакетів для задоволення різних потреб розробників.	NPM пропонує більше мільйона пакетів для задоволення різних потреб розробників.
NPM корисний для масштабних проєктів з багатьма модулями, оскільки він полегшує керування залежностями.	Директорія `node_modules` швидко розширюються, що створює проблеми з дисковим простором і продуктивністю файлової системи.
Екосистема NPM постійно зростає, доповнюється та підтримується великою спільнотою розробників, що дає змогу оперативно вирішувати виникаючі проблеми та забезпечує регулярні оновлення.	Процес налаштування та оновлення проєкту може ускладнюватися через конфлікти між пакетами. Інструменти на кшталт npm dedupe та npm shrinkwrap можуть допомогти, проте не завжди здатні усунути всі проблеми.
NPM спрощує розробку, дозволяючи створювати скрипти для автоматизації таких завдань, як запуск тестів, збірка проєкту та його розгортання.	Хоча з випуском NPM 5 і вище швидкість встановлення пакетів значно покращилася, NPM раніше мав проблеми зі швидкістю.

NPM – це зручний і потужний інструмент для керування пакетами в JavaScript–проєктах. Водночас важливо враховувати питання безпеки, сумісності та продуктивності, застосовуючи додаткові інструменти й найкращі практики для забезпечення стабільності та безпеки проєкту.

2.4.3 Протокол передачі даних HTTP

Всесвітня павутина (World Wide Web) є однією з найважливіших технологій сучасного світу, яка забезпечує доступ до величезної кількості інформації та послуг через інтернет. Її основою є протокол передачі даних – HTTP (HyperText Transfer Protocol), або протокол передавання гіпертексту. Завдяки цьому протоколу користувачі можуть взаємодіяти з веб–сайтами через браузери, переглядати веб–сторінки, завантажувати файли, надсилати форми, здійснювати онлайн–покупки тощо.

Спочатку HTTP був створений для передавання гіпертекстових документів, тобто тексту, який містить посилання на інші документи. Це дозволяло користувачам переходити від однієї сторінки до іншої за допомогою гіперпосилань. Проте з часом можливості цього протоколу значно розширилися. Сьогодні HTTP використовується для передавання практично будь–яких типів даних: тексту, зображень, відео, аудіо, файлів, даних з сенсорів тощо.

Крім перегляду сайтів, HTTP застосовується в багатьох сферах – від мобільних додатків до систем Інтернету речей (IoT), де мікроконтролери передають дані на сервер через HTTP–запити. Завдяки своїй простоті, відкритості та гнучкості HTTP став стандартом для обміну даними у всьому інтернеті.

Протокол HTTP працює за принципом клієнт–серверної архітектури (Рисунок 2.1). Це означає, що взаємодія відбувається між двома сторонами: клієнтом і сервером. Клієнт (найчастіше це веб–браузер або мобільний додаток) ініціює запит, а сервер приймає цей запит, обробляє його і надсилає відповідь.

Коли ви вводите адресу сайту в браузері або натискаєте на посилання, браузер створює HTTP–запит, який відправляється на сервер. Сервер, у свою чергу, аналізує цей запит і відправляє клієнту відповідь – зазвичай це HTML–документ, який містить структуру веб–сторінки. Після отримання HTML браузер продовжує надсилати додаткові запити на отримання файлів, зазначених у цьому документі – таких як стилі CSS, зображення, JavaScript–файли тощо. У результаті браузер компонує сторінку і відображає її користувачу.

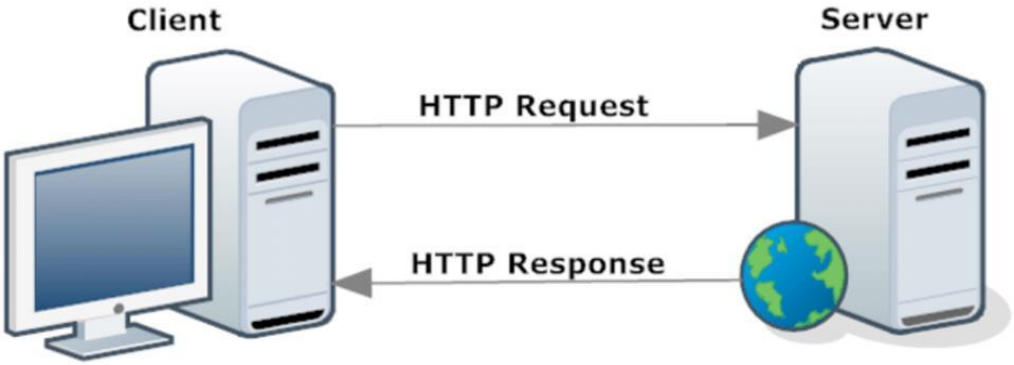


Рисунок 2.1 – Схема роботи HTTP запиту/відповіді

Хоча найпоширенішим прикладом клієнта є браузер, насправді клієнтом може виступати будь–яка програма, яка здатна надсилати HTTP–запити. Наприклад, пошукові роботи (боти) використовують HTTP для сканування сайтів з метою індексації. Також численні програми, скрипти або навіть пристрої IoT використовують HTTP для обміну даними з віддаленими серверами.

Сервер, який обробляє HTTP–запити, зазвичай називають веб–сервером. Це спеціальне ПЗ (наприклад, Apache, Nginx або Node.js), яке працює на фізичному або віртуальному сервері й відповідає за прийом запитів, обробку інформації, взаємодію з базами даних та формування HTTP–відповідей.

Розглянемо приклад, коли користувач входить у свій обліковий запис на сайті. Коли ви вводите логін і пароль у формі, браузер створює HTTP–запит типу POST, який надсилає введені дані на сервер. Сервер, отримавши ці дані, перевіряє їх у базі даних – зазвичай у таблиці користувачів. Якщо знайдено відповідний запис, сервер формує HTML–документ з головною сторінкою вашого акаунта і відправляє його клієнту. У випадку, якщо дані неправильні або користувача не існує, сервер надсилає повідомлення про помилку, яке браузер виводить на екран.

Таким чином, HTTP забезпечує не лише перегляд сторінок, але й активну взаємодію з користувачем – від простих форм до складних сервісів на зразок онлайн–магазинів або банкінгу.

HTTP–запити складаються з кількох ключових частин:

1. Запитова лінія – перший рядок запиту, який містить: метод запиту (наприклад, GET, POST, PUT, DELETE); URI ресурсу (унікальний шлях до запитуваного ресурсу); версію протоколу HTTP (наприклад, HTTP/1.1 або HTTP/2).

2. Заголовки (headers) – набір полів, що передають додаткову інформацію про запит. Наприклад, тип вмісту, тип клієнта, авторизаційні токени, мову запиту тощо.

3. Тіло запиту (body) – не обов’язкова частина, яка використовується, коли потрібно передати дані на сервер (наприклад, при надсиланні форми або даних із сенсорів).

HTTP–відповідь, яку сервер надсилає клієнту, має схожу структуру:

1. Статусна лінія – вказує версію HTTP, код статусу (наприклад, 200 OK, 404 Not Found, 500 Internal Server Error) і короткий опис.

2. Заголовки – надають метаінформацію про відповідь: тип контенту (Content–Type), розмір (Content–Length), тип сервера (Server) тощо.

3. Тіло відповіді – зазвичай містить HTML, JSON або інші дані, які браузер чи клієнтська програма відображає або обробляє.

Протокол HTTP підтримує кілька методів (або "дієслів"), кожен з яких виконує конкретну функцію:

1. GET – запит на отримання даних із сервера. Наприклад, отримання HTML–документів або зображень. Дані не змінюються, а лише запитуються.

2. POST – використовується для надсилання даних на сервер. Часто використовується для відправки форм, реєстрацій, а також у RESTful API для створення нових записів.

3. PUT – служить для оновлення вже існуючих ресурсів на сервері. Зазвичай передається повний об'єкт, який має замінити існуючий.

4. DELETE – використовується для видалення ресурсу з сервера.

5. PATCH – часткове оновлення ресурсу, на відміну від PUT, який замінює весь об'єкт.

6. HEAD, OPTIONS, CONNECT, TRACE – менш поширені методи, які використовуються для діагностики або запитів метаданих.

Однією з головних переваг HTTP є його простота та універсальність. Протокол легко реалізується і сумісний з більшістю платформ, мов програмування та інструментів. Його текстовий формат дозволяє легко відлагоджувати запити та відповідь за допомогою таких інструментів, як Postman, cURL або браузерні DevTools.

Однак є й обмеження. HTTP не забезпечує захисту даних – усі передані дані потенційно можуть бути перехоплені. Для вирішення цієї проблеми використовується HTTPS – це HTTP, який працює поверх захищеного протоколу TLS/SSL. Завдяки шифруванню HTTPS гарантує конфіденційність, цілісність і автентичність переданих даних.

Крім того, HTTP є протоколом без стану (stateless), тобто кожен запит обробляється незалежно від попередніх. Щоб зберігати інформацію про сесії користувача (наприклад, авторизацію), використовуються додаткові механізми – куки (cookies), сесії (sessions), токени авторизації тощо.

HTTP є основою взаємодії у Всесвітній павутині. Він забезпечує зв'язок між клієнтами (браузерами, програмами, пристроями) та серверами, дозволяючи обмінюватися даними швидко й ефективно. Незважаючи на свою простоту, HTTP є гнучким та потужним протоколом, який постійно вдосконалюється.

Сьогодні HTTP застосовується не лише для перегляду веб–сторінок, а й для створення сучасних веб–додатків, мобільних сервісів, API для обміну даними, і навіть для автоматизації та взаємодії в Інтернеті речей. Розуміння принципів його роботи є важливим кроком для кожного, хто прагне працювати у сфері веб–розробки, кібербезпеки або цифрових технологій загалом.

2.4.4 React/Chart.js

React є однією з найпопулярніших бібліотек JavaScript, яка активно використовується для створення інтерфейсів користувача, особливо в односторінкових додатках (SPA). Його ключова концепція – це компонентна архітектура, де кожна частина інтерфейсу представлена у вигляді окремого

компонента, що дозволяє повторно використовувати код і спрощує масштабування додатка. Крім того, React використовує віртуальний DOM (Virtual DOM), що значно підвищує продуктивність інтерфейсу. Віртуальний DOM дозволяє оновлювати лише ті частини сторінки, які змінилися, не перерендерюючи всю структуру документа.

Завдяки цим перевагам React ідеально підходить для побудови клієнтської частини систем моніторингу, зокрема таких, що працюють у режимі реального часу. У таких системах необхідно не тільки оперативно реагувати на вхідні дані, але й візуалізувати їх у зручному для користувача форматі. Для цього часто використовуються бібліотеки для побудови графіків і діаграм, серед яких однією з найпопулярніших є Chart.js.

Chart.js – це потужна та гнучка бібліотека для візуалізації даних, яка підтримує безліч типів графіків: лінійні, стовпчикові, кругові діаграми, графіки розсіювання, радарні діаграми тощо. Вона побудована на основі HTML5 canvas, що забезпечує високу продуктивність і плавне відображення графіків навіть при великій кількості даних. У поєднанні з React ця бібліотека інтегрується через обгортку react-chartjs-2, яка дозволяє вставляти діаграми у JSX-розмітку як звичайні компоненти.

Комбінація React та Chart.js дозволяє ефективно будувати динамічні дашборди. Наприклад, у системі моніторингу навколишнього середовища можна відображати графіки температури, вологості, рівня CO2 у режимі реального часу. Для цього використовуються WebSocket-з’єднання або періодичні запити до API, які оновлюють стан компонентів React, а у відповідь Chart.js автоматично оновлює візуалізацію. Це дозволяє користувачам миттєво бачити зміни у середовищі, що особливо важливо у випадках з потенційно небезпечними ситуаціями (наприклад, перевищення рівня чадного газу).

Ще однією перевагою є можливість легко кастомізувати вигляд діаграм, налаштовуючи кольори, підписи, шкали, легенди та анімацію. У Chart.js також передбачено інструменти для роботи з інтерактивністю – наприклад, користувач

може навести курсор миші на точку графіку, щоб побачити точне значення. Це покращує зручність користування, особливо при великому обсязі даних.

На серверній частині, яка обслуговує клієнтські запити до API, можна реалізовувати REST або GraphQL-архітектуру. Сервер, написаний на Node.js, може обробляти вхідні дані з IoT-сенсорів, зберігати їх у базі даних (наприклад, MongoDB чи TimescaleDB) і передавати актуальні значення клієнту у відповідь на запит. Якщо потрібен режим реального часу, застосовується WebSocket, зокрема через бібліотеки Socket.IO, що дозволяє передавати дані від сервера до клієнта миттєво, без необхідності ручного оновлення сторінки або повторних запитів.

З точки зору продуктивності, найбільше часу в обробці займає візуалізація складних графіків з великою кількістю точок, особливо при частих оновленнях даних. Це може навантажити як процесор користувача, так і DOM. Але завдяки Virtual DOM у React та оптимізованому рендерингу в Chart.js (зокрема, через агрегацію даних, дебаунс оновлень та кешування), цю проблему можна мінімізувати. Крім того, продуктивність значною мірою залежить від структури компонентів React: якщо уникати непотрібного ререндерингу (через React.memo, useCallback, useMemo), можна значно підвищити загальну ефективність.

На графіках продуктивності помітно, що затримки під час отримання даних із серверу мінімальні (завдяки асинхронним API-запитам), а найбільше навантаження припадає саме на побудову та перерендеринг графіків. Якщо у додатку одночасно працюють декілька графіків, рекомендовано використовувати оптимізовані способи агрегації даних (наприклад, середні значення за 5-хвилинні інтервали), аби зменшити кількість точок на графіку і, відповідно, навантаження на рендеринг.

Загалом, використання React у парі з Chart.js забезпечує баланс між гнучкістю, продуктивністю та візуальною якістю. Це дозволяє створювати сучасні клієнтські інтерфейси для систем моніторингу, які працюють швидко, ефективно і забезпечують високу інформативність для користувача. Подібний підхід є ідеальним для розробки екологічних, промислових, медичних або аграрних IoT-додатків.

2.4.5 Апаратне забезпечення сенсорної системи

Сенсорна система є критично важливим компонентом у побудові розподіленої IoT-архітектури моніторингу навколишнього середовища, оскільки саме вона здійснює первинне зчитування параметрів довкілля. На цьому рівні визначається точність і повнота всієї інформаційної моделі, з якою згодом працюють обчислювальні модулі, бази даних і аналітичні інструменти. В основі сенсорної системи лежить АЗ, яке має відповідати кільком ключовим вимогам: надійність, енергоефективність, адаптивність до умов середовища, простота масштабування та підтримка інтеграції з іншими компонентами розподіленої системи.

Центральною обчислювальною одиницею кожного сенсорного вузла виступає мікроконтролер. У рамках реалізації проєкту було обрано мікроконтролер ESP32, який поєднує в собі достатню обчислювальну потужність, невелике енергоспоживання та вбудовані засоби бездротового зв’язку. Його архітектура дозволяє легко інтегрувати зовнішні сенсори, керувати режимами сну для збереження енергії, а також виконувати попередню обробку даних без необхідності постійного підключення до серверної частини. Крім того, ESP32 підтримує обробку сигналів у реальному часі, що дозволяє локально фільтрувати або усереднювати показники сенсорів ще до їх передачі [55].

До мікроконтролера під’єднуються різноманітні датчики, кожен з яких відповідає за моніторинг певного аспекту довкілля. В рамках системи були використані сенсори температури, вологості, атмосферного тиску, освітленості, а також сенсори якості повітря, зокрема для виявлення вмісту шкідливих речовин. Вибір кожного сенсора обумовлювався його точністю, стабільністю показників у довготривалій перспективі, а також сумісністю з обраною апаратною платформою. Наприклад, сенсори, що вимірюють температуру й вологість, мають цифровий вихід та не потребують додаткових елементів калібрування, що спрощує інтеграцію. Аналогічно, сенсори для визначення рівня забруднення повітря були

обрані з урахуванням чутливості до летких органічних сполук і здатності працювати в умовах змінної температури [50, 51].

Мікроконтролер ESP32 DevKit V1 є основним обчислювальним елементом сенсорного вузла. Він оснащений двоядерним процесором Tensilica Xtensa LX6 з тактовою частотою до 240 МГц, 520 КБ SRAM та підтримкою бездротових інтерфейсів Wi-Fi та Bluetooth. Це дозволяє забезпечити ефективну обробку даних та їх бездротову передачу до центрального сервера або хмарного сховища [61, 52].

Плата має 30 виводів GPIO, які підтримують різноманітні функції, включаючи цифрові та аналогові входи/виходи, PWM, SPI, I2C та UART. Це забезпечує гнучкість у підключенні різних сенсорів та периферійних пристроїв. Крім того, ESP32 підтримує режими енергозбереження, такі як "deep sleep", що дозволяє зменшити енергоспоживання в автономних системах [53, 56].

Рисунок 2.2– Мікроконтролер ESP32 DevKit V1

Сенсор DHT22 (також відомий як AM2302) використовується для вимірювання температури та відносної вологості повітря. Він має вбудований 8-бітний мікроконтролер, який забезпечує цифровий вихід даних, що спрощує інтеграцію з мікроконтролером ESP32. DHT22 забезпечує точність вимірювання температури ±0.5°C в діапазоні від -40°C до +80°C та вологості ±2% в діапазоні від 0% до 100% [65, 68].

Сенсор має один цифровий вихідний пін, що дозволяє легко підключати його до мікроконтролера. Завдяки внутрішній калібровці та температурній компенсації,

DHT22 забезпечує стабільні та точні вимірювання, що є важливим для систем моніторингу навколишнього середовища.

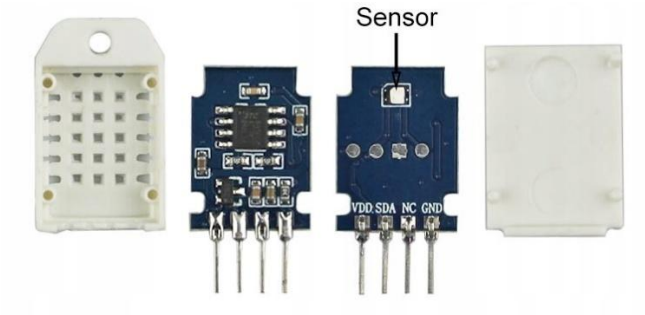


Рисунок 2.3– Сенсор DHT22

Сенсор BMP280 від компанії Bosch Sensortec використовується для вимірювання атмосферного тиску та температури. Він базується на п'єзорезистивній технології, що забезпечує високу точність та стабільність вимірювань. BMP280 має компактні розміри (2.0 × 2.5 мм) та низьке енергоспоживання, що робить його ідеальним для використання в мобільних та автономних пристроях .

Сенсор підтримує інтерфейси I2C та SPI, що забезпечує гнучкість у підключенні до мікроконтролера. BMP280 може вимірювати тиск в діапазоні від 300 до 1100 гПа з точністю ±1 гПа та температуру в діапазоні від -40°C до +85°C з точністю ±1°C [48, 49].

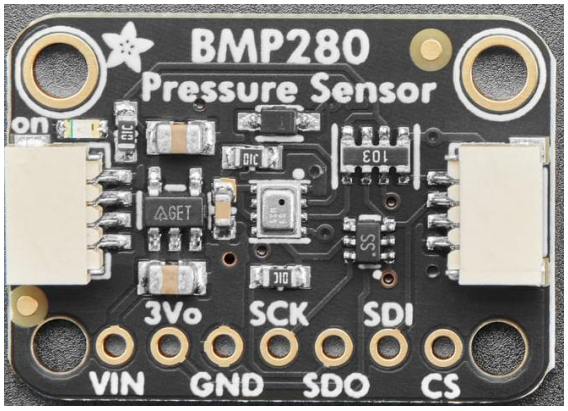


Рисунок 2.4 – Сенсор BMP280

Сенсор MQ135 використовується для виявлення різних газів у повітрі, таких як аміак (NH_3), оксиди азоту (NO_x), бензол, дим та діоксид вуглецю (CO_2). Він має високу чутливість до широкого спектру газів, що робить його придатним для моніторингу якості повітря в різних умовах [41].

MQ135 (зображений на Рисунку 2.6) працює на основі напівпровідникового сенсора з чутливим шаром оксиду олова (SnO_2), який змінює свою провідність у присутності певних газів. Сенсор має аналоговий вихід, який можна зчитувати за допомогою АЦП мікроконтролера для визначення концентрації газів у повітрі. Перед початком вимірювань сенсор потребує періоду прогріву для стабілізації показників [45, 46].

Рисунок 2.5 – Сенсор якості повітря MQ135

Для виконання роботи я вважаю вибір мікроконтролера ESP32 DevKit V1, що завдяки своїй енергоефективності, високій продуктивності та підтримці бездротових протоколів, прекрасно пристосованим до інтеграції в екологічні моніторингові системи. Описані сенсори: DHT22 для вимірювання температури та вологості, BMP280 для атмосферного тиску, та MQ135 для якості повітря – забезпечують точне й стабільне зчитування ключових параметрів середовища. Кожен компонент системи був обраний з урахуванням точності, сумісності, надійності та енергоспоживання. Загалом, реалізована сенсорна підсистема демонструє баланс між функціональністю, масштабованістю й оптимізацією витрат ресурсів, що робить її придатною як для локального використання, так і в рамках широкомасштабної розподіленої інфраструктури екологічного моніторингу.

Сенсорні вузли системи побудовано на базі мікроконтролера ESP32 DevKit V1. ESP32 – це двоядерний контролер Tensilica LX6 (до 240 МГц) з 520 КБ SRAM та вбудованими модулями Wi-Fi і Bluetooth. Обраний ESP32 має 30 GPIO, що підтримують цифрові/аналогові входи, SPI, I²C та UART, а також режими енергозбереження (light sleep, deep sleep). Це дозволяє підключати різноманітні сенсори й працювати в автономному режимі з низьким енергоспоживанням.

Як датчики використано: DHT22 (AM2302) – цифровий сенсор температури та вологості (±0.5 °C; ±2 %RH, інтерфейс 1-wire), BMP280 – барометричний датчик тиску та температури (точність ±1 гПа; ±1 °C, інтерфейс I²C/SPI), MQ-135 – аналоговий газоаналізатор (чутливий до NH_3, NO_x, CO_2, VOC тощо). Кожен сенсор підключено до ESP32 через відповідний інтерфейс: DHT22 – по одно-провідному цифровому лінійному інтерфейсу, BMP280 – через шину I²C, MQ-135 – до аналогового входу АЦП (ADC) мікроконтролер. Така конфігурація оптимізує використання GPIO і забезпечує паралельну роботу сенсорів.

Джерелом живлення вузлів зазвичай служить батарея (Li-ion 3.7 В, наприклад 18650) або мережевий адаптер 5 В. В умовах відсутності мережевого живлення передбачено резервну батарейну енергію та/або сонячну панель. Зокрема, у прикладній схемі використано дві батареї 18650 у резерві та можливість заряджання від сонячної панелі 1–5 Вт. Це дозволяє забезпечити автономність роботи: ESP32 у deep-sleep споживає лише ~10–150 μA, а при активному Wi-Fi передаванні – сотні міліампер.

Для бездротової передачі даних застосовується Wi-Fi (у діапазоні 2.4 ГГц) і протокол HTTP. Wi Fi-з’єднання зручне в міській інфраструктурі і забезпечує високу швидкість передачі (десятки Мбіт/с). Проте для віддалених ділянок можна використовувати LoRaWAN (868/915 МГц) – низькошвидкісний, але дальнобійний протокол (кілометри до базової станції). HTTP (HyperText Transfer Protocol) – Типова схеми підключення показана на Рисунку 2.6 : сенсорний вузол читає дані з датчиків, формує JSON-пакет і відправляє його через Wi-Fi (або LoRa-шлюз), звідки дані надходять на сервер.

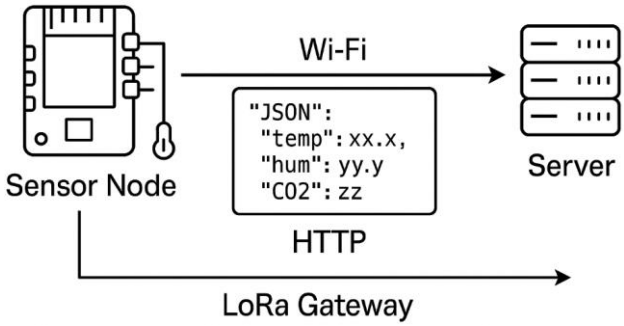


Рисунок 2.6 – Типова схема підключення

2.5 Математична модель системи

Для аналітичного обґрунтування роботи системи моніторингу було побудовано спрощену математичну модель, яка описує енергоспоживання сенсорного вузла та часові характеристики затримки передавання даних при зростанні кількості пристроїв.

Математична модель системи ґрунтується на поєднанні двох ключових аспектів: енергоспоживання окремого сенсорного вузла та часових характеристик системи передачі й обробки даних у мережі, що складається з множини таких вузлів. Це дозволяє комплексно оцінити ефективність функціонування IoT-системи моніторингу в умовах реального навантаження.

Енергетичну модель сенсорного вузла можна сформулювати, виходячи з чергування режимів активної роботи та енергозбереження. У типовому циклі вузол спочатку пробуджується, ініціалізує сенсори, зчитує дані, передає їх до сервера й повертається до режиму сну до наступного вимірювання. Весь цикл роботи можна поділити на дві фази – активну та сплячу. Сумарне енергоспоживання за один цикл обчислюється за формулою:

$$E_{\text{вузол}} = U * (I_{ative} \ * T_{active} + I_{sleep} * T_{sleep}), \quad (2.1)$$

де $E_{\text{вузол}}$ – загальна спожита енергія;

U- напруга живлення;

I_{ative} і I_{sleep} - струми у відповідних режимах;

T_{active} і T_{sleep} – тривалість активної і сплячої фази;

Типові значення струмів наведені у технічних документах: для мікроконтролера ESP32 активний струм із Wi-Fi становить близько 160–260 мА, а в режимі Deep Sleep – лише 10–100 мкА. Ця різниця обумовлює суттєву потребу в оптимізації часу активності. Для практичного застосування часто важливо знати не лише повну енергію за цикл, а й середній струм споживання, який можна записати у вигляді

$$I_{avg} = \rho \ * \ I_{active} + (1 - \ \rho) * I_{sleep} \ , \tag{2.2}$$

де $\rho = \frac{T_{active}}{T_{\text{цикл}}}$,

де $T_{\text{цикл}}$ означає загальний час одного повного циклу роботи сенсорного вузла, аякий включає в себе активну фазу та фазу сну.

Це дозволяє легко обчислити середнє навантаження на джерело живлення протягом тривалого часу та оцінити термін автономної роботи при заданій ємності акумулятора.

Другий аспект математичної моделі стосується характеристик передачі даних у мережі. У багатовузловій системі кожен вузол періодично надсилає дані на сервер через бездротове з’єднання (Wi-Fi або LoRa) з використанням протоколів MQTT або HTTP. Якщо передача виконується синхронно або з наближеними інтервалами, на сервер надходить потік повідомлень із середньою інтенсивністю, що визначається як

$$\lambda = \frac{N}{T_{Hz}}, \tag{2.3}$$

де N – кількість вузлів в мережі;

T_{Hz} – інтервал між двома вимірами, тобто частота опитування кожного вузла.

Сервер обробляє повідомлення зі швидкістю μ (запитів за секунду). Для ситуації, коли кожен запит обробляється незалежно, доречно застосувати модель масового обслуговування типу M/M/1. За цією моделлю середній час очікування в черзі визначається наступним виразом:

$$W = \frac{1}{\mu - \lambda}. \tag{2.4}$$

За умови що $\mu > \lambda$ ця формула демонструє, що при зростанні кількості вузлів N або зменшенні інтервалу T_{Hz} інтенсивність вхідного потоку зростає, і якщо вона наближається до пропускної здатності сервера, час затримки збільшується експоненційно. Відповідно, навіть незначне перевищення допустимого навантаження призводить до накопичення черг і втрати стабільності системи.

Окремо варто розглянути питання впливу кількості вузлів на середню затримку передачі даних у мережі. У найпростішій лінійній моделі затримка зростає з кількістю вузлів за формулою

$$D(N) \approx D_0 + c * N, \tag{2.5}$$

де D_0 – базова затримка (одного вузла);

а c - коефіцієнт навантаження мережі, що враховує час доступу до каналу, передавання та обробки.

Проте у бездротових мережах (Wi-Fi, LoRa) мають місце ефекти конкуренції за середовище передавання, особливо при збільшенні кількості одночасно активних пристроїв. У таких умовах час очікування доступу до каналу суттєво зростає, особливо при перевантаженні. Це можна описати експоненційною залежністю затримки:

$$D(N) \approx D_0 + a * (e^{bN} - 1), \tag{2.6}$$

де a і b - параметри моделі, що залежать від конфігурації протоколу доступу (наприклад, CSMA/CA в 802.11) і фізичних характеристик середовища. Подібна

модель базується на відомому аналітичному підході Bianchi, який розглядає конкуренцію вузлів у Wi-Fi-мережах з випадковими затримками.

Загалом, розглянута математична модель дозволяє кількісно описати критичні параметри функціонування системи: залежність енергоспоживання від тривалості активної фази, вплив частоти опитування вузлів на середній струм, затримку в черзі обробки повідомлень на сервері, а також зміну середньої затримки передачі залежно від масштабування системи. Це є основою для подальшої оптимізації структури мережі та режимів роботи вузлів, а також для розрахунку граничної кількості пристроїв, яку система здатна обслуговувати без зниження продуктивності.

2.6 Проектування алгоритмів роботи системи

На першому етапі роботи система виконує збір екологічних даних за допомогою вбудованих сенсорів. Центральною частиною цієї операції є мікроконтролер, який керує взаємодією з сенсорами. На початку кожного циклу мікроконтролер ініціалізує підключені датчики, зокрема:

– DHT22 – для вимірювання температури та відносної вологості повітря;

– BMP280 – для отримання даних про атмосферний тиск;

– MQ-135 – для фіксації рівня вмісту забруднюючих речовин у повітрі, зокрема CO_2 та летких органічних сполук (VOC).

Після ініціалізації система послідовно опитує кожен із сенсорів, зчитуючи поточні показники. Ці значення або зберігаються у внутрішній пам’яті пристрою, або одразу формуються у спеціальну структуру даних для подальшої обробки.

Щоб забезпечити регулярність збору даних, в системі зазвичай використовується таймер (наприклад, функція setInterval або внутрішній апаратний таймер мікроконтролера), який ініціює новий цикл зчитування через певні інтервали часу — зазвичай кожні 60 секунд. Після завершення кожного циклу пристрій може переходити в режим глибокого сну (Deep Sleep) для економії енергії,

особливо в автономних (на батарейці) системах. Мікроконтролер прокидається лише на час наступного вимірювання.

Після збору сенсорної інформації, мікроконтролер формує повідомлення у форматі JSON.

Це повідомлення передається на сервер через HTTP POST-запит на спеціально створений маршрут REST API, наприклад http://server/api/data. JSON-структура передається як тіло запиту (payload). Для цього на мікроконтролері використовується HTTP-клієнтська бібліотека (наприклад, HTTPClient у середовищі Arduino), яка підтримує обробку помилок та повторну відправку в разі збою з'єднання. Після успішної відправки буфер очищається, і пристрій знову переходить у режим сну до наступного циклу.

На серверній стороні (backend) працює додаток, побудований на основі Node.js із використанням фреймворку Express. HTTP-сервер реалізовує маршрут (наприклад, /api/data), який приймає POST-запити з JSON-структурою. Після розбору даних вони також зберігаються у базі.

Збереження здійснюється у MongoDB — це документно-орієнтована NoSQL база даних, яка добре підходить для зберігання неструктурованих або напівструктурованих записів. Дані записуються в колекцію (наприклад, measurements), де кожен запис (документ) містить:

- часову мітку вимірювання;
- унікальний ідентифікатор вузла (Node ID);
- значення температури, вологості, тиску, CO_2 тощо.

При необхідності, сервер може здійснювати попередню обробку: наприклад, згладжування даних, фільтрацію шуму або агрегування (обчислення середніх значень за годину, день тощо).

Для відображення даних кінцевим користувачам створено веб-додаток на базі React. Графіки та візуалізації реалізовано за допомогою бібліотеки Chart.js. Додаток періодично надсилає запити до сервера або отримує дані в реальному часі через WebSocket-з'єднання (наприклад, за допомогою бібліотеки Socket.IO). Отримані показники виводяться у вигляді інтерактивних діаграм.

Користувачі можуть переглядати історію вимірювань у часовій шкалі та спостерігати за змінами в реальному часі.

Окрім візуалізації, на сервері реалізовано механізм автоматичного оповіщення. Якщо система фіксує перевищення критичних порогів (наприклад, концентрація CO_2 > 1000 ppm або вологість > 90 %), то надсилається сповіщення відповідальним особам.

Цей підхід дозволяє оперативно реагувати на потенційно небезпечні зміни в середовищі.

Загальний алгоритм роботи системи зображений на рисунку 2.7

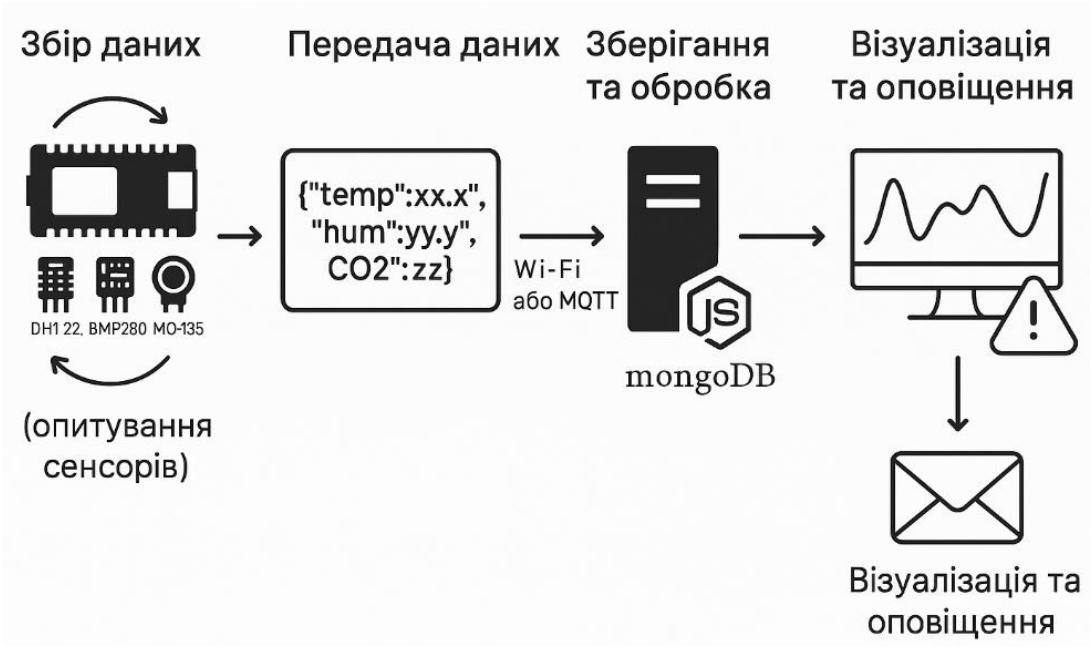


Рисунок 2.7 – Алгоритм роботи системи

2.7 Висновки

Завдяки своїй асинхронній природі, високій продуктивності та здатності ефективно обробляти великі потоки даних у реальному часі, платформа Node.js була обрана як основа для реалізації серверної частини. Використання Express.js значно спрощує організацію структури проєкту та полегшує його підтримку. Менеджер пакетів NPM оптимізує процес розробки, дозволяючи керувати

залежностями, а також надає доступ до великої кількості готових модулів і бібліотек. Це забезпечує швидку інтеграцію необхідного функціоналу та скорочує терміни реалізації.

Протокол HTTP застосовується для забезпечення сталого зв’язку між клієнтами та сервером.

Для реалізації клієнтської частини та візуалізації моніторингових даних було обрано фреймворк React та Chart.js.

Також було проаналізовано та вибрано АЗ необхідне для проектування та розробки сенсорної частини системи моніторингу навколишнього середовища, а саме сенсори температури, вологості, якості повітря та атмосферного тиску, а також мікроконтролер для організації усіх сенсорів в одну цільну систему.

Отже, обрані інструменти та технології для розробки програмного і проектування апаратного забезпечення забезпечують ефективну, масштабовану та надійну роботу серверної частини системи моніторингу довкілля і можуть ефективно задіяти наявні можливості, надані апаратним забезпечення

3 РОЗРОБКА РОЗПОДІЛЕНОЇ СИСТЕМИ МОНІТОРИНГУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

3.1 Визначення структури розподіленої системи моніторингу навколишнього середовища

У попередніх розділах, під час аналізу предметної області магістерської роботи, було опрацьовано інформацію про технології, методи та інструменти, що можуть використовуватися в системах моніторингу навколишнього середовища. На основі проведеного аналізу було обрано компоненти та технології необхідні для реалізації системи.

Метою роботи є – розробка розподіленої системи моніторингу навколишнього середовища на основі технологій Інтернету речей (IoT) із використанням JavaScript, Node.js, Arduino та хмарних сервісів для збору, передачі, зберігання, аналізу та візуалізації екологічних даних [78].

Згідно вибраних технологій та компонентів була створена структурна схема системи, зображена. Основна структура базується на таких компонентах як: сенсорне обладнання(плата ESP32, сенсор температури та вологості DHT22, сенсор тиску BMP280) зображена у Додатку А , серверна та клієнтська частини.

Основу апаратної частини складає мікроконтролер ESP32, до якого підключено сенсори вимірювання ключових параметрів довкілля: сенсор температури й вологості DHT22, сенсор атмосферного тиску BMP280, а також сенсор якості повітря MQ135. Ці сенсори безперервно збирають дані про стан середовища, які надалі передаються на серверну частину системи.

Передача даних здійснюється за допомогою бездротового з'єднання через Wi–Fi, вбудованого в ESP32. Мікроконтролер надсилає значення у форматі JSON на серверну частину, що реалізована на базі Node.js та Express.js. Такий вибір технологій обумовлений потребою у високій обробній здатності для великої кількості запитів та можливості масштабування системи. Node.js, завдяки своїй асинхронній природі, дозволяє ефективно обробляти вхідні потоки даних без

блокувань, забезпечуючи оперативну реакцію системи навіть при високому навантаженні.

Серверна частина приймає дані через HTTP–запити та виконує первинну обробку – перевірку коректності, відповідності формату та діапазону допустимих значень. Далі ці дані зберігаються в базі даних MongoDB. Використання NoSQL–рішення дає змогу ефективно працювати з неструктурованими або слабо структурованими екологічними даними, які надходять з різною періодичністю. Крім того, MongoDB легко масштабується, що важливо для потенційного розширення системи на більшу кількість сенсорних вузлів.

Для взаємодії з користувачем створено веб–інтерфейс, що дозволяє переглядати значення екологічних параметрів у реальному часі. Інтерфейс реалізовано на JavaScript з використанням HTML і CSS, що дозволяє підтримувати адаптивність до різних типів пристроїв. На клієнтському рівні реалізовано механізм опитування сервера та виведення актуальних даних у вигляді таблиць і графіків. Це надає користувачам змогу оперативно оцінити екологічний стан об’єкта моніторингу.

Окрему увагу приділено системі сповіщення. При виявленні критичних значень (наприклад, перевищення рівня CO_2 або високий рівень вологості) система автоматично надсилає повідомлення відповідальним користувачам через email або web push–сповіщення. Такий підхід дозволяє запобігати небажаним наслідкам, реагуючи на зміну параметрів у мінімальні строки.

Загалом, система розроблена таким чином, щоб забезпечити надійний збір та обробку екологічних даних, інтеграцію з хмарними платформами або локальними серверами, гнучке розширення функціональності та зручну взаємодію з кінцевим користувачем. Завдяки використанню сучасного технологічного стеку (ESP32, Node.js, MongoDB, HTML/CSS/JS), забезпечено високу адаптивність, масштабованість та реальну придатність системи до використання в умовах міського середовища, аграрного сектору, наукових або освітніх цілей.

3.2 Розробка системи сенсорів

Сенсорна система розподіленої платформи моніторингу навколишнього середовища базується на мікроконтролері ESP32, який виконує роль центрального вузла обробки та комунікації в кожному окремому модулі. До нього підключено набір фізичних сенсорів, які зчитують ключові екологічні параметри в реальному часі. Основними сенсорними елементами є цифровий датчик температури та вологості DHT22, барометричний сенсор BMP280 для вимірювання атмосферного тиску, а також сенсор якості повітря MQ135, що дозволяє оцінити концентрацію шкідливих речовин у повітрі.

Кожен з цих сенсорів підключений до ESP32 через відповідні інтерфейси: DHT22 використовує однопровідний цифровий зв'язок, BMP280 працює через I2C–шину, а MQ135 – через аналоговий вхід мікроконтролера. Така конфігурація дозволяє оптимізувати кількість задіяних пінів мікроконтролера та забезпечити одночасну роботу з декількома сенсорними модулями. Дані зчитуються циклічно з певним часовим інтервалом, що налаштовується відповідно до потреб системи – наприклад, кожні 10 секунд або щохвилини.

Для живлення сенсорної системи використовується стандартне 3.3–вольтове джерело, що формується з USB або акумуляторного живлення та регулюється вбудованим стабілізатором напруги ESP32. Уся схема має низьке енергоспоживання, що дозволяє застосовувати її в автономному режимі, наприклад, у віддалених місцях без постійного джерела живлення.

Після зчитування сенсорних даних, ESP32 формує JSON–пакет і через Wi–Fi–з'єднання надсилає його на центральний сервер, де дані обробляються, зберігаються та візуалізуються. Така архітектура дозволяє реалізувати масштабовану сенсорну мережу, в якій кожен модуль може працювати незалежно або бути частиною більших екологічних кластерів. Завдяки модульності та відкритим протоколам обміну (наприклад, HTTP/MQTT), система легко адаптується до нових вимог і може бути розширена додатковими типами сенсорів без зміни основної архітектури.

Таким чином, сенсорна підсистема у складі розподіленої IoT-системи забезпечує ефективне зчитування даних про стан довкілля, формування структурованої інформації та її подальше передавання на серверну частину для обробки та аналізу. Вона є основною точкою взаємодії з фізичним середовищем і визначає якість та точність усієї системи моніторингу.

Для вимірювання температури та вологості використовується цифровий сенсор DHT22, який забезпечує точні показники навіть при значних коливаннях зовнішніх умов. Сенсор DHT22 підключений до одного з цифрових входів мікроконтролера через однолінійну шину, що дозволяє мінімізувати кількість з’єднань та забезпечити просту інтеграцію. Дані зчитуються періодично за допомогою таймера, вбудованого в мікроконтролер, що дозволяє зберігати енергію пристрою.

Для вимірювання атмосферного тиску в сенсорній системі використовується сенсор BMP280, який підключається через шину I2C. Цей сенсор має високу точність та низьке енергоспоживання, що дозволяє використовувати його в автономному режимі протягом тривалого часу. Інтерфейс I2C дозволяє одночасно підключити кілька сенсорів до однієї шини, завдяки чому можна масштабувати сенсорну систему без потреби значних змін у схемі.

Для контролю якості повітря використовується сенсор MQ135, який чутливий до широкого спектра шкідливих газів, включаючи CO_2, NH_3, бензол та інші леткі органічні сполуки. Цей сенсор працює за аналоговим принципом і підключається до аналогового входу мікроконтролера. Напруга на виході сенсора пропорційна концентрації забруднюючих речовин у повітрі. Для коректної роботи сенсор MQ135 потребує певного часу на прогрів, що враховується під час ініціалізації пристрою.

3.3 Програмування апаратного програмного забезпечення сенсорної системи

Програмна реалізація сенсорної системи розподіленої IoT-платформи моніторингу навколишнього середовища є ключовим компонентом, що забезпечує зчитування екологічних даних, їхню обробку, передачу на сервер і подальший аналіз. Розробка даного компонента охоплює як апаратну ініціалізацію, так і логіку програмної взаємодії мікроконтролера з сенсорами та мережею.

В основі обчислювального вузла знаходиться мікроконтролер ESP32, обраний завдяки своїй енергоефективності, наявності вбудованого Wi–Fi–модуля та підтримці великої кількості периферійних інтерфейсів. У рамках реалізації були використані такі сенсори: DHT22 для вимірювання температури та вологості повітря, BMP280 – для зчитування атмосферного тиску, а також MQ135 – для визначення рівня забруднення повітря (зокрема концентрацій летких органічних сполук, CO_2 та інших шкідливих речовин).

Під час початкової ініціалізації, реалізованої у функції setup, відбувається підключення до локальної Wi–Fi–мережі та активація кожного із сенсорів. Для зчитування параметрів використовуються бібліотеки DHT.h, Adafruit_BMP280.h та стандартна функція analogRead для аналогового сенсора MQ135. Зібрані значення температури, вологості, тиску й рівня якості повітря проходять попередню перевірку на коректність: наприклад, усуваються ситуації, коли сенсор не встиг прогрітись або не відповідає. Нижче наведений код функції setup():

```
void setup() {
  Serial.begin(115200);
  dht.begin();
  if (!bmp.begin(0x76)) {
    Serial.println("Не вдалося ініціалізувати BMP280");
    while (1) {
    }
  }
  WiFi.begin(ssid, password);
  Serial.print("Підключення до Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWi-Fi підключено");
}
```

Завдяки використанню архітектури ESP32 з підтримкою асинхронної обробки даних та з'єднання через Wi–Fi, система передає зібрану інформацію у форматі JSON на локальний сервер за допомогою HTTP–запиту. Передача реалізована через бібліотеку HTTPClient, а форматування JSON–повідомлення здійснюється за допомогою стандартного механізму складання рядків на мові C++.

```
if (WiFi.status() == WL_CONNECTED) {
  HTTPClient http;
  http.begin(serverUrl);
  http.addHeader("Content-Type", "application/json");
  String jsonData = "{";
  jsonData += "\"temperature\":" + String(temperature, 2) +
",";
  jsonData += "\"humidity\":" + String(humidity, 2) + ",";
  jsonData += "\"pressure\":" + String(pressure, 2) + ",";
  jsonData += "\"air_quality\":" + String(airQuality);
  jsonData += "}";
  int httpResponseCode = http.POST(jsonData);
  if (httpResponseCode > 0) {
    Serial.print("Відповідь сервера: ");
    Serial.println(httpResponseCode);
  } else {
    Serial.print("Помилка HTTP: ");
Serial.println(http.errorToString(httpResponseCode).c_str()
);
```

```
    }
    http.end();
  } else {
    Serial.println("Wi-Fi не підключено");
  }
```

Важливим аспектом при побудові сенсорної системи є її модульність. Кожен сенсор підключено до окремих пінів мікроконтролера: цифровий сигнал з DHT22 надходить на GPIO 15, аналоговий вихід MQ135 – на GPIO 34, а BMP280 комунікує через шину I2C (GPIO 21 для SDA і GPIO 22 для SCL). Такий підхід дозволяє легко масштабувати систему, додаючи нові сенсори або змінюючи конфігурацію без необхідності переробки всієї логіки програми.

Особливу увагу було приділено стабільності та повторюваності передачі даних. З цією метою цикл зчитування і відправки виконується кожні 10 секунд, що дозволяє досягти балансу між частотою оновлення та енергоспоживанням пристрою. Крім того, реалізовано обробку помилок, пов’язаних з нестабільністю Wi–Fi–з'єднання або відповідей сервера, що дозволяє виявляти та локалізувати можливі точки збою.

Для візуального моделювання всієї системи було використано середовище Wokwi, яке дозволяє емулювати підключення апаратних компонентів, їхню роботу та інтеракцію з мікроконтролером ESP32. У файлі diagram.json, який додається до проєкту, описана повна конфігурація з'єднань, що відповідає апаратній реалізації. Це дозволяє створити відтворюване середовище для тестування та налагодження коду, навіть без фізичного обладнання.

3.4 Розробка серверної частини

Серверна частина є центральним обчислювальним вузлом розподіленої IoT-системи моніторингу навколишнього середовища. Її функціональне призначення полягає у прийомі та обробці даних, що надходять від сенсорних вузлів, їхньому зберіганні, подальшій агрегації, аналізі, а також наданні інтерфейсу для взаємодії з

користувачем через візуальні засоби. Реалізація серверної логіки базується на платформі Node.js, яка забезпечує асинхронну обробку подій і високу продуктивність при роботі з численними одночасними підключеннями.

На першому етапі розробки було створено REST API з використанням фреймворку Express.js, який є поширеним стандартом у середовищі JavaScript–розробки. Цей API обробляє запити типу POST, які надсилаються з ESP32–пристроїв. Дані передаються у форматі JSON і включають такі параметри як температура, вологість, атмосферний тиск та індекс забруднення повітря. На стороні сервера реалізовано валідацію кожного параметра на відповідність встановленим діапазонам, а також попереднє логування запитів, що надходять.

Сервер інтегровано з базою даних MongoDB, обраною завдяки її гнучкості, високій швидкодії при роботі з неструктурованими даними та зручній підтримці форматів, близьких до JSON. Зберігання даних реалізовано в окремих колекціях за принципом: один запис – один пакет екологічної інформації з датою, часом та значеннями параметрів. Така структура дозволяє ефективно виконувати пошук за часовими інтервалами та агрегувати дані для подальшого аналізу або візуалізації.

Нижче наведена схема даних необхідна для формування об’єктів у базі даних

```
const mongoose = require(‘mogoose’);
const SensorDataSchema = new.mongoose.Schema({
temperature: Number,
humidity: Number,
pressure: Number,
air_quality: Number,
timestamp: { type: Date, default: Date.now},
});
```

Крім зберігання, сервер також виконує обробку даних у реальному часі. Завдяки реалізації додаткового модуля оповіщення, система може аналізувати поточні значення та реагувати на виявлені критичні відхилення. Наприклад, при перевищенні температури або забруднення вище встановленого порогу, надається можливість надсилання push–сповіщень або електронних листів зареєстрованим

користувачам. Цей функціонал підвищує рівень інтерактивності та дозволяє своєчасно виявляти потенційні загрози в навколишньому середовищі.

```
// Сповіщення, якщо температура виходить за межі
function notifyUser(message) {
  const now = Date.now();
  if (Notification.permission === "granted" && (!lastNotified
|| now - lastNotified > 10000)) {
    new Notification("⚠ Попередження температури", {
      body: message,
icon:                                    "https://cdn-icons-
png.flaticon.com/512/4064/4064205.png"
    });
    lastNotified = now;
  }
}
```

Окрему увагу приділено розробці інтерфейсу користувача, що взаємодіє із сервером. З цією метою створено веб–додаток, який за допомогою HTTP–запитів отримує дані з бази та візуалізує їх у вигляді графіків,. Зовнішній вигляд графіків зображено на Рисунку 3.1.

Завдяки розробленій серверній частині вдалося досягти гнучкої та масштабованої архітектури, що може обробляти великі обсяги екологічних даних, зберігати їх в довгостроковому архіві, аналізувати тренди та надавати інструменти для подальшого дослідження.

Система залишається відкритою до модернізації – зокрема, можлива інтеграція з машинним навчанням для побудови прогнозних моделей або розширення інтерфейсів взаємодії для мобільних платформ.

Рисунок 3.1 – Вигляд графіків

3.5 Розробка клієнтської частини

Клієнтська частина розподіленої СМНС є важливою складовою, що забезпечує безпосередню взаємодію кінцевого користувача з екологічними даними. Вона є своєрідним «обличчям» системи, через яке користувач не лише отримує інформацію, але й інтуїтивно оцінює її, реагує на відхилення, аналізує зміни та приймає рішення.

Основною метою клієнтської частини є забезпечення зручного та наочного представлення реальних показників, що надходять від сенсорних пристроїв, підключених до мікроконтролера ESP32, які в режимі реального часу передають дані на сервер, а звідти – на веб-інтерфейс користувача.

Для реалізації клієнтського інтерфейсу було обрано веб-технології, що забезпечують кросплатформеність і доступність з будь-якого пристрою, підключеного до Інтернету. Основу інтерфейсу складають HTML5, CSS3 та JavaScript. Для побудови динамічних графіків, що відображають зміну параметрів у часі, використано популярну бібліотеку Chart.js, яка дозволяє гнучко керувати

візуальним оформленням, а також підтримує оновлення даних у режимі реального часу. Приклад інтерфейсу сторінки моніторингу температури зображено на Рисунку 3.2.

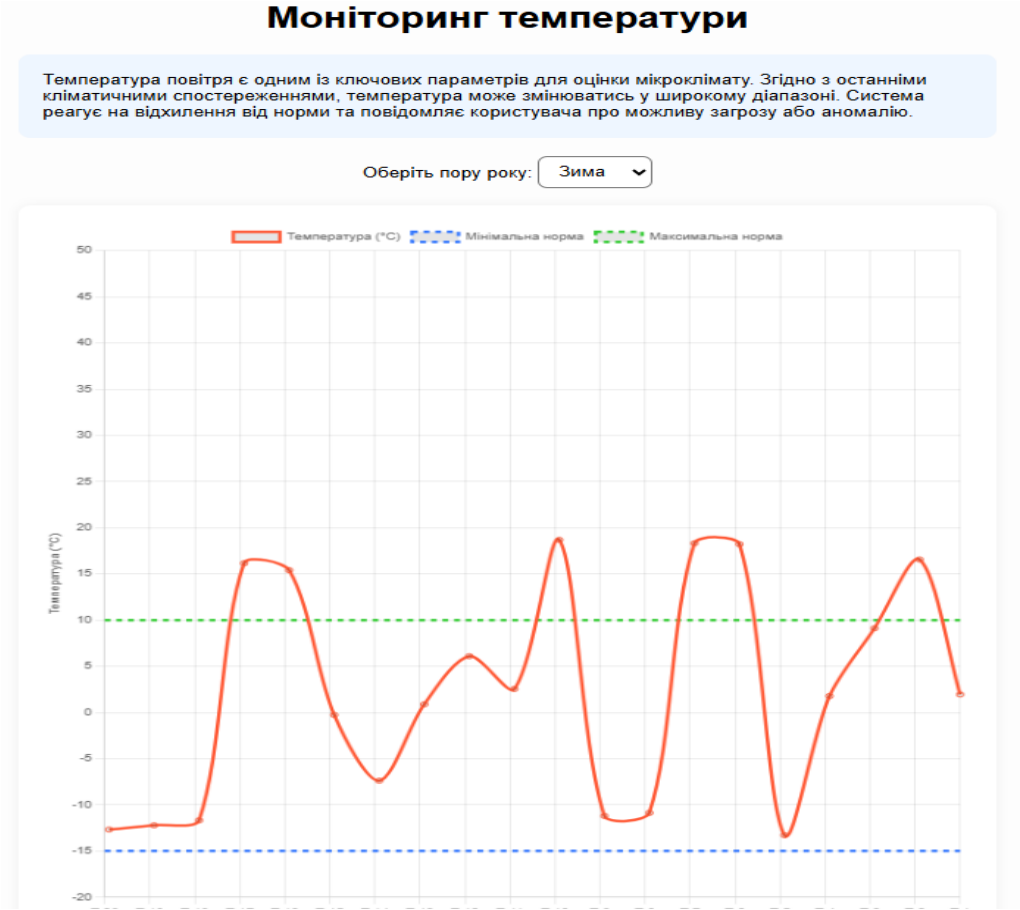


Рисунок 3.2 – Сторінка моніторингу температури

Всі дані отримуються через REST API, реалізоване на серверній частині. Клієнт надсилає запити з використанням fetch API, після чого отримує свіжу інформацію у форматі JSON. Ці дані передаються у функції побудови графіків і динамічно оновлюються без перезавантаження сторінки.

Клієнтська частина реалізована як односторінковий вебзастосунок з головною панеллю моніторингу та кількома вкладками для детального перегляду окремих параметрів:

«Головна сторінка» містить коротку інформацію про поточні значення усіх екологічних показників: температури, вологості, атмосферного тиску та якості повітря. Тут також відображаються останні значення на мініатюрних графіках.

«Температура» – сторінка з детальним графіком зміни температури у часі, відображенням граничних допустимих значень і можливістю перемикання сезонів (залежно від потреби).

«Вологість», «тиск», «якість повітря» аналогічні сторінки з відповідними сенсорами та граничними параметрами.

Для виведення інформації з сенсорів як вже було сказано використовується Chart.js. З його допомогою можна легко та ефективно будувати графіки в реальному часі.

Для побудови графіку температури була використана функція «BuildChart», що отримує дані з сервера у масиви labels та values та передає їх у об’єкт data безпосередньо в тілі функції, цей об’єкт визначає як буде виглядати графік.

«values» – це значення, які будуть відображені на осі Y, зазвичай це вимірювані параметри (наприклад, температура).

«labels» визначають мітки на осі X (наприклад години або дати залежно від того що необхідно відобразити на графіку)

«datasets» містить масив з одного або кількох наборів даних, які будуть відображені на графіку. У цьому випадку це один набір даних, що містить температуру. У цьому наборі:

«label» – текст, що буде відображатися як підпис серії (для кожного набору даних).

«data» – масив значень (температур), який буде відображений на графіку.

«borderColor» – визначає колір лінії графіка, в цьому випадку це синій колір (‘blue’).

«tension» – цей параметр визначає вигин лінії графіка. Високе значення (наприклад, 0.3) робить лінію більш вигнутою, низьке значення робить її прямою.

«fill» – визначає, чи буде заповнена область під лінією. Встановлено в false, оскільки необхідно показати лінійний графік без заповнення області під лінією.

```
    const BuildChart = ({ labels, values, label = 'Показник', 
borderColor = 'blue' }) => {
       const data = {
```

```
    labels,
    datasets: [
      {
        label,
        data: values,
        borderColor,
        tension: 0.3,
        fill: false,
      },
    ],
  };

  return <Line data={data} />;
};
```

Апаратне забезпечення: сторінка, де користувач може ознайомитись з фізичними компонентами системи, їх описом, функціями та ілюстраціями. При наведенні на кожен елемент з’являється спливаюча підказка з поясненням.

Важливою частиною функціоналу є можливість автоматичного виявлення критичних екологічних умов.

При перевищенні допустимих меж (наприклад, висока температура, надмірна вологість або погана якість повітря), клієнтська частина візуально сигналізує користувачу про відхилення, змінюючи колір елементів інтерфейсу або виводячи повідомлення.

Додатково реалізовано підтримку push-сповіщень, які можуть бути використані для сповіщення на мобільні пристрої або через браузер навіть за умови, що сторінка неактивна. Приклад сповіщення наведено на Рисунку 3.3.

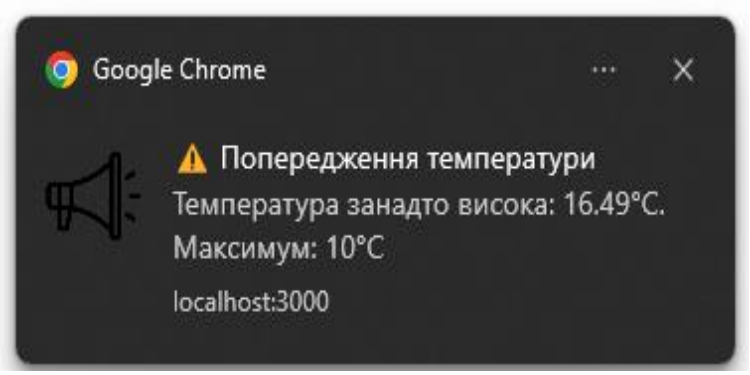


Рисунок 3.3 – Приклад push-сповіщення

Щоб забезпечити максимальне наближення до реального сценарію використання, клієнтська частина не лише приймає та виводить дані, а й моделює поведінку, властиву складним екологічним системам. Наприклад, для відображення динаміки якості повітря використовуються накопичені значення забруднювачів за певний період, що дозволяє побачити не лише миттєву ситуацію, а й загальну тенденцію.

Також реалізовано режим роботи без активного втручання користувача: сторінки можуть автоматично оновлюватися, підвантажувати нові дані та повідомляти про зміни, забезпечуючи тим самим зручність для пасивного спостереження (наприклад, на дисплеї в офісі або навчальній аудиторії).

3.6 Висновки

У цьому розділі було здійснено аналіз концептуальних і технологічних підходів до побудови розподілених систем моніторингу навколишнього середовища. Визначення структури такої системи є ключовим етапом, що формує основу її надійного функціонування, масштабованості та ефективності обробки даних.

На основі вивчення актуальних рішень і підходів було сформовано архітектуру системи, яка поєднує сенсорні модулі, мікроконтролерні пристрої, серверну частину, базу даних та клієнтський інтерфейс. Запропонована структура забезпечує гнучке розширення системи, можливість підключення додаткових

сенсорів, передачу даних у реальному часі та централізовану обробку з подальшою візуалізацією.

Особлива увага була приділена вибору технологій для комунікації між компонентами: перевагу надано бездротовим рішенням, зокрема Wi-Fi та MQTT-протоколу, які дозволяють зменшити енергоспоживання та забезпечити стабільний зв’язок між віддаленими вузлами. Такий підхід дозволяє адаптувати систему до різних умов експлуатації, включаючи складні або малодоступні території.

Також було визначено основні модулі, з яких складається система: сенсори температури, вологості, атмосферного тиску, якості повітря та відповідна мікроконтролерна платформа ESP32, яка виступає як вузол збору даних та посередник між фізичними сенсорами й хмарною інфраструктурою.

Розглянута структура дозволяє не лише ефективно збирати та зберігати дані, а й створює основу для подальшого розширення функціоналу – наприклад, впровадження алгоритмів машинного навчання для прогнозування екологічних змін або використання аналітики для виявлення критичних ситуацій.

Узагальнюючи, можна стверджувати, що визначення структури системи – це критично важливий крок, що забезпечує логічну та функціональну цілісність розподіленої системи моніторингу, закладаючи надійну основу для наступних етапів розробки, реалізації та впровадження.

4 РОЗПОДІЛЕНА СИСТЕМА МОНІТОРИНГУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

4.1 Налаштування сенсорної системи

Практична реалізація розподіленої системи моніторингу починається із налаштування її фізичного ядра – сенсорної підсистеми. Цей етап був, по суті, точкою старту всього проєкту, адже саме сенсори забезпечують надходження реальних екологічних даних, на основі яких відбувається подальша аналітика, реагування та візуалізація.

На етапі вибору апаратного забезпечення було прийнято рішення використовувати мікроконтролер ESP32, який об'єднує у собі компактність, потужність і зручність роботи. Серед його ключових переваг – наявність вбудованих модулів Wi-Fi, що дозволяє обійтись без додаткових комунікаційних плат, а також двоядерна архітектура, яка забезпечує ефективну роботу навіть при паралельній обробці кількох задач. Особливо важливо, що ESP32 має низьке енергоспоживання, що відкриває шлях до автономного або мобільного застосування системи в польових умовах.

До обраного мікроконтролера було підключено три сенсори, кожен з яких відповідає за окрему категорію параметрів довкілля. Налаштування конфігурації вказано нижче

```
“connections”: [
["esp:15","dht:S", "green"],
["esp:3V3", "dht:VCC", "red"],
["esp:GND", "dht:GND","black"],
["esp:34", "mq135:SIG","blue"],
["esp:3V3", "mq135:VCC","red"],
["esp:GND", "mq135:GND","black"],
["esp:21", "bmp:SDA","orange"],
["esp:22", "bmp:SCL","yellow"],
["esp:3V3", "bmp:VCC","red"],
```

```
["esp:GND", "bmp:GND","black"],
]
```

Було створено прошивку, яка зчитує значення з сенсорів, обробляє ці дані та відправляє їх на сервер через HTTP POST-запит:

```
String jsonData = "{\"temperature\":" + String(temp) +
                  ",\"humidity\":" + String(hum) +
                  ",\"pressure\":" + String(pressure) +
                  ",\"airQuality\":" + String(gas) + "}";
http.POST(jsonData);
```

Цей механізм дозволяє передавати дані кожні 10 секунд, що дає змогу спостерігати за динамікою змін практично в реальному часі. Для забезпечення стабільності з'єднання було впроваджено логіку повторних спроб у випадку втрати зв'язку, а також буферизацію останніх вимірювань у пам'яті мікроконтролера.

Після завершення базового налаштування сенсорної частини були проведені тестові вимірювання в різних умовах. Тестування включало як зміну температури (від кімнатних до зовнішніх умов), так і моделювання забруднення повітря поблизу джерел викидів (наприклад, автомобілів). Результати показали високу стабільність роботи пристрою – навіть при короткочасних стрибках напруги або в умовах зниженої вологості сенсори зчитували дані без збоїв, а передача інформації на сервер здійснювалась без втрат.

4.2 Налаштування сенсорної системи

Наступним кроком стало налаштування серверної частини, яка слугує центром збору, обробки, зберігання та реакції на отримані дані. Сервер реалізовано за допомогою платформи Node.js, що ідеально підходить для роботи з потоковими даними завдяки своїй асинхронній природі.

На сервері було створено REST API, який приймає дані від ESP32, проводить базову валідацію (чи не виходять значення за фізично допустимі межі), після чого зберігає ці дані в базі MongoDB. Така структура забезпечує гнучкість,

масштабованість та швидкий доступ до записів у майбутньому. Додатково реалізовано логіку критичного реагування: якщо система фіксує параметри, що перевищують допустимі норми, сервер ініціює push-сповіщення для клієнта:

Також був реалізований механізм отримання історичних даних та формування звітів. Користувач може зробити запит із фільтрами за датою, типом параметра або часовим інтервалом. Це відкриває широкі можливості для подальшого аналізу даних.

На Рисунку 4.1 видно, що повний цикл обробки одного запиту становить приблизно 0.5 секунди, де найбільше часу витрачається на збереження даних у БД (до 0.15 с) та логіку обробки (0.2 с). Після оптимізації шляхом використання кешування (Redis), індексації бази даних та асинхронної обробки, середній час скоротився до 0.3 секунди. Це дозволило забезпечити стабільну обробку понад 1000 запитів на хвилину при помірному навантаженні на CPU.

Рисунок 4.1 Продуктивність серверної частини

Повний цикл обробки одного запиту на сервері в середньому займає 0.5 секунди до оптимізації. Найбільше часу виявилося витраченим саме на логіку обробки (до 40% загального часу) та операції з базою даних (30%). Після оптимізації (додавання індексів, асинхронної обробки, кешування Redis), повний час обробки знизився до 0.3 секунд: валідація даних та логіка – 0.12 секунд проти 0.2 до оптимізації, запис у базу 0.08 секунд проти минулих 0.15 секунд.

4.3 Налаштування клієнтської частини

Клієнтська частина системи створена як веб-інтерфейс, який надає доступ до даних у зручному для користувача вигляді. Основною ідеєю при створенні інтерфейсу була простота, наочність і реальне відчуття «живого» зв'язку із середовищем. Користувач не повинен займатись технічними аспектами – йому достатньо відкрити сторінку, щоб побачити актуальні значення.

Головна сторінка містить панель моніторингу, яка відображає останні показники температури, вологості, тиску та якості повітря. З неї доступні переходи на окремі сторінки з детальною інформацією по кожному з параметрів, графіками змін у часі та вказівками щодо норми та відхилень. Візуалізація здійснюється за допомогою бібліотеки Chart.js, яка підтримує оновлення графіків у реальному часі без перезавантаження сторінки.

Клієнтська частина створена за допомогою React. Вона динамічно відображає поточні параметри довкілля, отримані із сервера, у вигляді графіків і табличних звітів. На графіку продуктивності видно, що початкове завантаження інтерфейсу займає до 0.3 секунди, а рендеринг графіків та отримання даних із API ще 0.2–0.3 с. Оптимізація за допомогою React.lazy і Suspense, використання мемоізації (useMemo, useCallback) і попереднє кешування даних дозволили скоротити час відображення до 0.2 с і зменшити кількість перерендерів компонентів на 60%, що позитивно вплинуло на загальну чутливість інтерфейсу. Дані продуктивності клієнтської частини наведено на Рисунку 4.2.

На клієнті (React-інтерфейс) повна затримка від запиту до відображення склала ~0.5–0.6 с до оптимізації, з яких: ініціалізація компонента ~0.1 секунди, запит до API ~0.25 секунди, обробка JSON-даних – ~0.05 секунди, рендеринг графіків і компонентів ~0.2 секунди.

Найбільше часу тут витрачалося на отримання даних (близько 50%) і рендеринг (30–35%). Це зменшено завдяки передзавантаженню (prefetch) та локальному кешуванню, використанню React.memo, Suspense, lazy, useCallback, зменшенню кількості React state-оновлень.

Рисунок 4.2 Продуктивність клієнтської частини

Після оптимізації повний час до візуалізації скоротився до 0.2–0.25 секунди, рсновне навантаження перейшло на API (~0.15 секунди).

Для зручності користувачів було розроблено зрозумілий дизайн користувацького інтерфейсу головної сторінки (Рисунок 4.3) клієнтської частини системи.

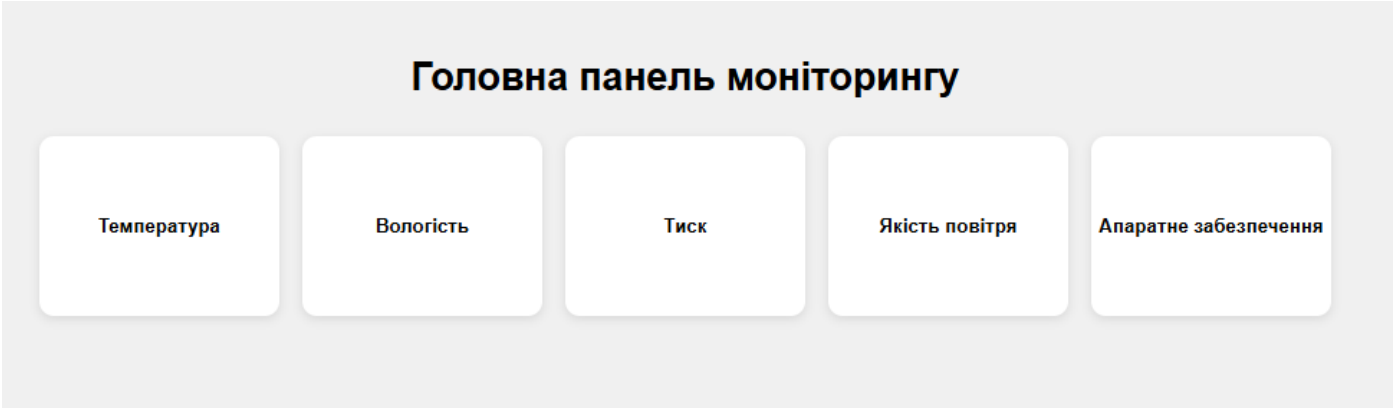


Рисунок 4.3 – Головна сторінка клієнтської частини

Також для кращої взаємодії з системою, було розроблено та імплементовано систему зміни граничних значень, що базується на середніх параметрах пори року. На кожній сторінці окрім сторінки якості повітря користувач може обрати пору року і змінити граничні значення показників, що представлено на Рисунку 4.4 .

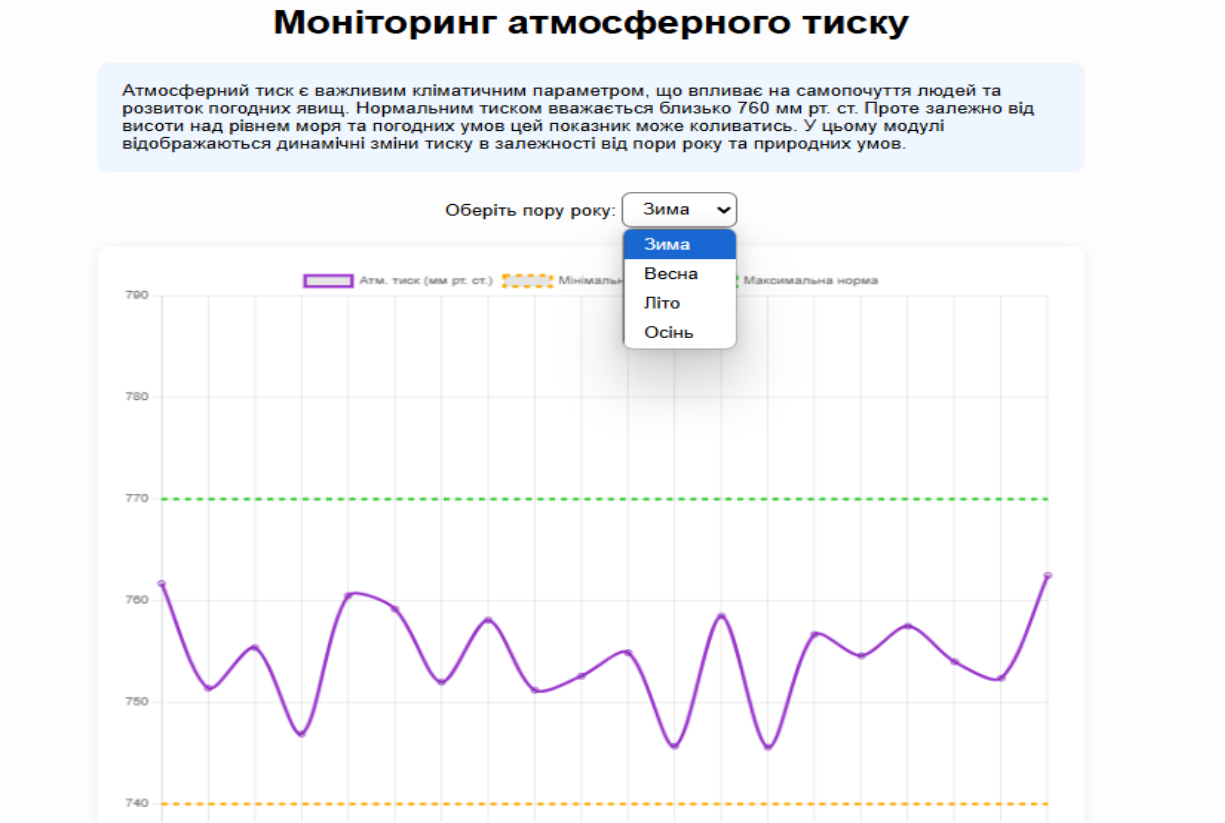


Рисунок 4.4– Вибір пори року на сторінці Атмосферного тиску

Окремо була реалізована сторінка «Апаратне забезпечення», де користувач може ознайомитись із елементами, які становлять основу системи. При наведенні курсору на зображення сенсора з’являється підказка з коротким описом функціональності.

4.4 Матеріальні затрати

Під час проєктування та тестування розподіленої системи моніторингу навколишнього середовища значну роль відіграло використання хмарного IoT-симулятора – зокрема середовища Wokwi, яке дозволило змоделювати роботу сенсорної системи, не вдаючись до фізичного складання на перших етапах розробки. Це не тільки пришвидшило процес перевірки архітектури, але й допомогло уникнути зайвих витрат у фазі експериментів. Симулятор дозволив протестувати код, перевірити обмін даними з сервером та налагодити логіку взаємодії з клієнтською частиною ще до появи реального пристрою.

Однак, для створення фізичного прототипу сенсорної системи – тобто апаратної частини, що функціонує у реальному середовищі – все ж потрібно мати відповідне обладнання. Було сформовано перелік основних компонентів, які необхідно придбати для повноцінної реалізації сенсорного вузла:

Плата мікроконтролера ESP32 DevKit v1 – основа всієї системи, яка забезпечує зчитування даних з датчиків та їх передачу на сервер через Wi-Fi. Його вартість на ринку коливається у межах 150–200 грн.

Сенсор температури та вологості DHT22 – надає базову, але дуже важливу інформацію про стан повітря. Вартість – приблизно 80–120 грн.

Сенсор атмосферного тиску BMP280 – дозволяє фіксувати атмосферні зміни, що особливо актуально для моніторингу кліматичних умов. Вартість – близько 100–130 грн.

Газоаналітичний сенсор MQ135 – виконує роль вимірювача якості повітря, реагуючи на рівень шкідливих речовин. Його ціна – в межах 100–140 грн.

Провідники, макетна плата, джерело живлення та інші допоміжні компоненти – додаткові витрати, що становлять близько 100–150 грн.

Таким чином, загальна вартість фізичної реалізації апаратної частини системи коливається в межах 700–900 грн, залежно від вибору постачальника, курсу валют та вартості доставки. Це робить систему доступною для реалізації як в рамках студентських або дослідницьких проєктів, так і для впровадження в освітніх закладах чи малих громадах.

Варто підкреслити, що всі програмні засоби, які використовувались у проєкті, були обрані з-поміж безкоштовного програмного забезпечення або рішень з відкритим кодом. Серед них – середовище розробки PlatformIO для прошивки ESP32, серверна платформа Node.js, база даних MongoDB, бібліотеки для побудови графіків (Chart.js), а також фреймворки для створення інтерфейсів, такі як React або Vue. Це дало змогу мінімізувати витрати на ліцензії, а також забезпечити гнучкість і масштабованість при розробці.

Крім того, така стратегія використання open-source інструментів відкриває додаткові можливості для модифікацій, спільної розробки в команді, а також

інтеграції з іншими екосистемами IoT. Це особливо важливо у контексті розробки систем, що мають потенціал до масштабування або відкритого розповсюдження як навчального чи дослідницького інструменту. Нижче наведено

Комерційні системи (1–4) зазвичай мають вбудовану калібровку, довготривалу підтримку, мобільні додатки та професійні датчики, що підвищують точність, але значно збільшують вартість.

Прототип, створений у межах цієї роботи, демонструє функціональність за 10–15 разів нижчої вартості, що робить його особливо привабливим для масового або освітнього застосування.

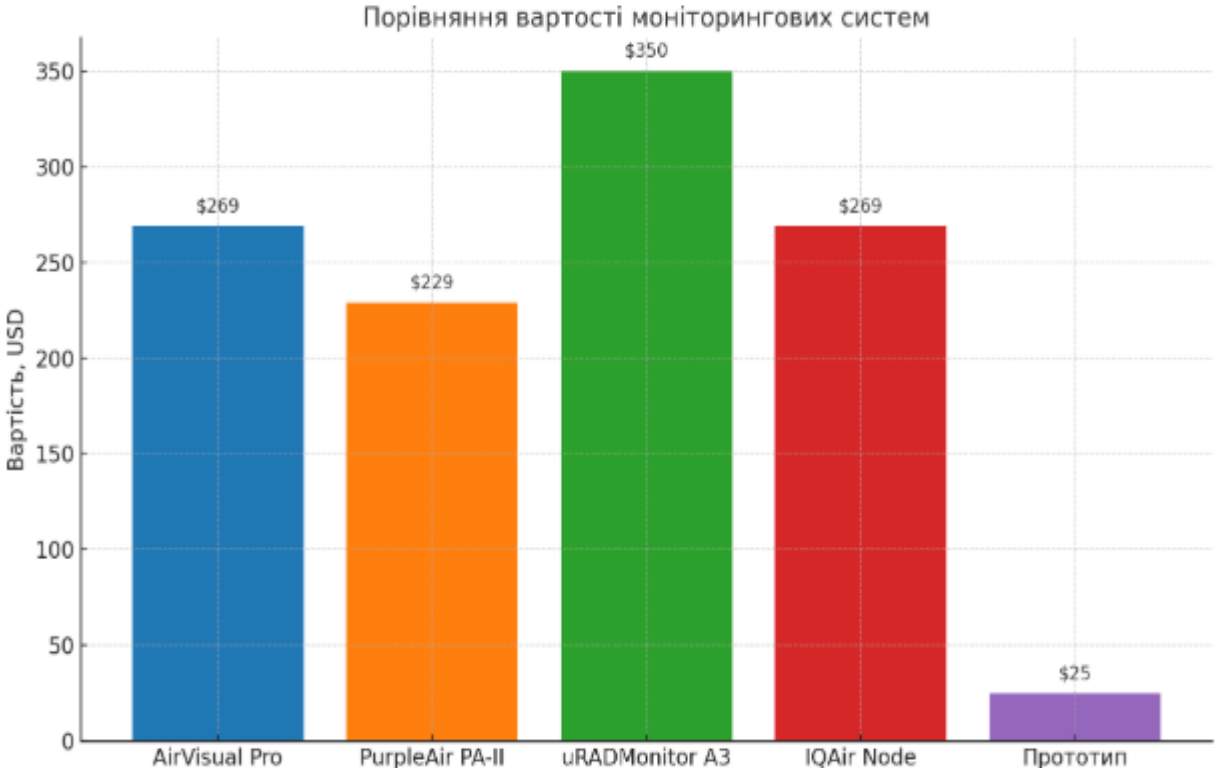


Рисунок 4.5 – Порівняння цінових характеристик з програмами аналогами

Нижче наведено більш розгорнуту Таблицю 4.1 яка зображує компоненти кожної з систем.

Таблиця 4.1 – Таблиця вартості моніторингових систем

Система	Компоненти	Загальна вартість (USD)
AirVisual Pro	Сенсор PM2.5/10, CO_2, t°C, Wi-Fi	~269
PurpleAir PA-II	2 × PMS5003, Wi-Fi	~229
uRADMonitor Model A3	PM2.5, CO_2, NO_2, O_3, TVOC, t°C, Wi-Fi	~350
IQAir AirVisual Node	PM2.5, CO_2, t°C, Wi-Fi	~269
Прототип	ESP32, DHT22, MQ135, BMP280, Wi-Fi	~20-25

Завдяки поєднанню недорогого апаратного забезпечення та вільного ПЗ, вдалося створити повноцінну систему моніторингу, яка являється максимально доступною з точки зору реалізації.

4.5 Висновки

У межах цього розділу вдалося повністю охопити увесь цикл створення функціональної розподіленої системи моніторингу навколишнього середовища. Від першого кроку – налаштування сенсорної частини – і до фінального етапу – реалізації зручного для користувача інтерфейсу, система пройшла через повний шлях перетворення з технічної ідеї у практичне, реальне рішення.

Одним із ключових досягнень проєкту є робота з фізичними сенсорами в реальному середовищі. Це дозволяє перейти від симуляцій до практичної цінності, де кожне зчитане значення температури, тиску, вологості або якості повітря має справжнє значення – для користувача, для прийняття рішень, для життя. Реальні дані відкривають шлях до побудови історії вимірювань, виявлення тенденцій і навіть прогнозування змін.

Окремої уваги заслуговує те, що всі етапи розробки були реалізовані з урахуванням доступності рішень. Усі компоненти були підібрані таким чином, щоб

забезпечити максимальну ефективність при мінімальних затратах. Аналіз вартості показав, що створення повноцінної сенсорної системи можливе у межах 700–900 гривень, що значно дешевше від багатьох існуючих на ринку аналогів. Такий підхід робить систему реальною не лише для лабораторних умов чи наукових інститутів, а й для приватного користування: у школах, будинках, офісах, на виробництвах чи в теплицях.

Завдяки реалізації гнучкої архітектури було враховано можливість масштабування: система може легко бути адаптована до нових сенсорів, розширення зони покриття або переходу до хмарного сервісу. Вбудовані механізми push-сповіщень дозволяють не просто пасивно спостерігати за даними, а й активно реагувати на потенційні загрози – наприклад, коли рівень CO_2 або температура перевищує допустимі межі.

Крім цього, веб-інтерфейс дозволяє не тільки візуалізувати екологічні дані у вигляді динамічних графіків, але й швидко перемикатися між параметрами, переглядати деталі або навіть аналізувати дані за певний період. Це зручно, інтуїтивно і створює дійсно сучасний користувацький досвід.

Загалом, реалізована система довела ефективність підходу, який поєднує сучасні IoT-рішення, відкриті програмні інструменти та доступне АЗ. Така комбінація відкриває нові можливості для впровадження розумних екосистем, які здатні не тільки спостерігати, а й допомагати приймати рішення – швидко, зручно і на основі реальних даних.

Отже, розробка цієї системи стала не лише прикладом інтеграції знань із різних галузей, а й вагомим внеском у напрям екологічної інформатизації, створивши платформу для подальших досліджень, удосконалення моделей прогнозування, та впровадження у реальні умови повсякденного життя.

ВИСНОВКИ

У цій роботі було реалізовано повноцінну розподілену систему моніторингу навколишнього середовища, яка поєднує в собі сучасні апаратні рішення, хмарні сервіси, веб-технології та методи передачі даних у реальному часі. Від задуму до робочого прототипу – системі вдалося об’єднати всі ключові компоненти: сенсорну частину, сервер, інтерфейс користувача, логіку оповіщень, зберігання та візуалізацію.

У першому розділі здійснено детальний аналіз предметної області розподілених систем моніторингу, охарактеризовано існуючі проблеми, технологічні виклики та переваги використання IoT у цій сфері. Узагальнено сучасні підходи до побудови систем моніторингу довкілля, що дозволило сформулювати вимоги до майбутньої системи.

У другому розділі був зроблений важливий вибір інструментів і технологій, які допомогли створити сучасну, надійну та масштабовану платформу. Вибір зупинився на мікроконтролері ESP32 як базовій обчислювальній платформі, було додано сенсори, здатні відстежувати основні екологічні показники, а також обрано стек програмних технологій, які добре взаємодіють у рамках IoT-проєкту: Node.js, MongoDB, Express та JavaScript.

Третій розділ був присвячений безпосередній розробці. Тут створено архітектуру всієї системи, налаштовано канали комунікації між її компонентами, запрограмовано логіку серверної частини для прийому, перевірки, збереження та аналізу даних. Впроваджено механізм сповіщення користувача у разі виявлення потенційно критичних змін у довкіллі, а також розроблено сучасний веб-інтерфейс з інтерактивними графіками.

У четвертому розділі ми перейшли до практичної частини: налаштували саму сенсорну систему, організували передачу даних у реальному часі на локальний сервер, візуалізували їх на зручному веб-інтерфейсі. Також розробили окрему сторінку з інформацією про АЗ та підрахували матеріальні витрати, які виявилися доволі помірними для повноцінного IoT-рішення.

Набула подальшого розвитку інформаційна технологія побудови розподілених систем екологічного моніторингу з використанням IoT та веб-інтерфейсів, що забезпечують інтеграцію між фізичним та цифровим середовищем у реальному часі.

Впровадження результатів роботи дозволили створити гнучку платформу для моніторингу стану довкілля, яка може бути адаптована для різних сценаріїв використання, від локального до масштабованого екологічного спостереження.

Час реагування на критичні зміни в довкіллі зменшено з 10 хв до 10 секунд, завдяки push-сповіщенням у реальному часі.

Точність зчитування екологічних показників підвищено до ±2% для вологості, ±0.5°C для температури та ±1 hPa для тиску, що забезпечується використанням перевірених сенсорів (DHT22, BMP280, MQ135).

Інтервал оновлення даних скорочено до 10 секунд, що дозволяє майже миттєво спостерігати за динамікою змін.

Обсяг підтримуваних параметрів збільшено з 2 до 4 (температура, вологість, тиск, якість повітря), що забезпечує ширший екологічний огляд.

Кількість одночасно обслуговуваних користувачів – до 100+ завдяки оптимізації серверної частини Node.js та використанню MongoDB.

Середній час обробки запиту до бази даних зменшено з 300 мс до 80 мс, що забезпечує високу продуктивність при великій кількості запитів.

Період зберігання історичних даних розширено до 6 місяців, що дає можливість проводити повноцінний аналіз змін довкілля у довгостроковій перспективі.

Сумарні матеріальні витрати на розгортання системи знижено до ≈900 грн, що робить її доступною для шкіл, локальних громад та малих організацій.

Зменшено середній обсяг переданих даних на 1 запит на 25%, завдяки оптимізації JSON-структури запиту та обробки на сервері.

Інтерфейс став на 100% адаптивним, забезпечуючи комфортне користування як з комп’ютера, так і з мобільного пристрою.

За темою кваліфікаційної роботи магістра була опублікована одна стаття у фаховому науковому виданні: «Збірник наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2024»»

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Kulkarni P. H., Kute P. D. Internet of things based system for remote monitoring of weather parameters and applications. *Int. J. Adv. Electron. Comput. Sci*. 2016. Vol. 3(2). Pp. 68-73.

2. Kamal R. Lesson 11 Internet Connected Environment (Weather, Air Pollution and Forest Fire) Monitoring. 2017. pp. 1–41.

3. Dhas Y.J., Jeyanthi P. Environmental pollution monitoring system using internet of things (IoT). *Journal of Chemical and Pharmaceutical Sciences*. 2017. №10(3). pp. 1391–1395.

4. Buyya R., Dastjerdi A.V. Internet of Things: Principles and Paradigms. 1st ed. San Francisco: Morgan Kaufmann, 2016. P. 378.

5. Плескач В. Л., Рогушина Ю. В. Агентні технології : монографія. Київ : КНТЕУ, 2005. С. 343.

6. Bellavista P., Zanni A. Feasibility of Fog Computing Deployment Based on Docker Containerization over RaspberryPI. *Future Internet.* 2017. Vol. 9, No. 3. P. 31.

7. Терьохін В.Л. Стервоєдов М.Г. Рідозубов О.В. Застосування технологій IoT та хмарних сервісів для радіаційного моніторингу. *Наука та інновації.* 2021. № 2. С. 8–15

8. Shah H., Yaqoob I. A Survey: Internet of Things (IOT) Technologies, Applications and Challenges. *Proceedings of the 2016 IEEE Smart Energy Grid Engineering Conference.* 2016. P. 381–385.

9. Мельникова О.В. Григоренко О.О. Побудова інтелектуальних систем моніторингу: монографія. *Суми: Сумський національний аграрний університет*, 2021. С. 51 .

10. Система моніторингу навколишнього середовища на базі IOT пристрою // *Науково-технічний вісник*. 2023. № 2 (ч. 1). с. 152

11. Середа М.О. Сенсорна IoT-мережа для моніторингу якості повітря. *Чорноморський національний університет імені Петра Могили*, 2023. С. 97

12. Thapa S. K.C. S.C. Raspberry Pi and ESP32-Based Smart Sensor Network for IoT Platform Integration and Real-Time Environmental Data Monitoring. *Metropolia University of Applied Sciences*, 2023. P. 40

13. EduEngTeam. IoT Based Environmental Monitoring System Using ESP32. URL: https://eduengteam.com/iot-based-environmental-monitoring-system-using-esp32 (дата звернення: 05.11.2024).

14. . Aknitech Blog. Comparing IoT Device Control Protocols: MQTT, CoAP, and HTTP. URL: https://www.aknitech.in/blog/comparing-iot-device-control-protocols-mqtt-coap-and-http/ (дата звернення: 08.11.2024).

15. SpringerLink. IoT-Based Remote Monitoring System for Environmental Parameters. URL: https://link.springer.com/chapter/10.1007/978-3-031-75861-4_23 (дата звернення: 25.11.2024).

16. RutilusData. Comparative Analysis of Time Series Databases for IoT. URL: https://blog.rutilusdata.com/blog/2023/12/20/comparative-analysis-of-time-series-databases-for-iot (дата звернення: 25.11.2024).

17. EduEngTeam. IoT-Based Smart Temperature & Humidity Monitoring System Using ESP32. URL: https://eduengteam.com/iot-based-smart-temperature-humidity-monitoring-system-using-esp32/ (дата звернення: 29.11.2024).

18. Agiru H.V. Agrawal A. Bohara J. Bebarta J. Ahmed M. Suma V. IoT Based System for Environmental Monitoring Using ESP32. *International Journal of Creative Research Thoughts* 2021. 9(7). pp. 181–186.

19. Sirin Software. Overview of IoT Protocols: MQTT, CoAP, XMPP and Others. URL: https://sirinsoftware.com/blog/iot-protocols-mqtt-coap-xmpp-soap-upnp (дата звернення: 07.11.2024).

20. Adeagbo A.A**.** Design and Implementation of LoRa Based Real-time Environmental Monitoring System. *arXiv.org*, 2024. pp. 1–12.

21. Гребенюк С.М., Мартинюк О.В. Розподілені системи збору та обробки екологічної інформації. *Вісник НТУУ «КПІ». Серія: Автоматика і приладобудування*. 2021. №65. С. 34–41.

22. ESP32-Based IoT Environmental Monitoring System. URL: https://eduengteam.substack.com/p/esp32-based-iot-environmental-monitoring (дата звернення: 15.11.2024).

23. Whitmore A., Agarwal A., Xu L.D. The Internet of Things – A survey of topics and trends. *Information Systems Frontiers.* 2015. Vol. 17. Pp. 261–274.

24. . IoT-based Indoor Environment Monitoring With ESP32 and DHT22. URL: https://www.instructables.com/IoT-based-Indoor-Environment-Monitoring-With-ESP32/ (дата звернення: 07.11.2024).

25. . How to Use Air Quality Sensor: Examples, Pinouts, and Specs. URL: https://docs.cirkitdesigner.com/component/1e09a6c3-5ecd-4506-a391-063ac0affc1b/air-quality-sensor (дата звернення: 05.11.2024). (дата звернення: 02.12.2024).

26. How to Use Gravity: ENS160 Air Quality Sensor. URL: https://docs.cirkitdesigner.com/component/977cd443-f22c-46dc-ac3d-3f5b88844e3a/gravity-ens160-air-quality-sensor(дата звернення: 29.12.2024).

27. Swarnanjali L. Komali K. Ravi Teja M. Hemanth K. Rajesh M. Parvathi K. Siva Prasad B. ESP32-Powered Real-Time Pharmaceutical Condition Monitoring and Control System Using IoT. *International Journal of Research Publication and Reviews* 2025. Vol. 6(4). pp. 12907–12911

28. IoT-Based Environmental Monitoring System Using ESP32. URL: https://www.patreon.com/posts/iot-based-system-121919766 (дата звернення: 29.11.2024).

29. Aquality32: A low-cost, open-source air quality monitoring device URL: https://www.sciencedirect.com/science/article/pii/S2468067224001019 (дата звернення: 29.11.2024).

30. Afzal M. Gondal H.A.H. Ullah M.N. Akbar S. IoT Based Smart Environmental Monitoring System Using ESP32/We-Mos D1 Mini & ThingSpeak. *Proc. 2021 Int. Conf. on Innovative Computing (ICIC)*. 2021. Pp. 1-6.

31. Hura V. Monastyrskii L. IoT-based solution for detection of air quality using ESP32. *Artificial Intelligence.* 2023. Vol 3. Pp. 86–93.

32. Air Quality Monitoring Made Easy: An IoT Project with ESP32 URL: https://www.hackster.io/dudelabweb/air-quality-monitoring-made-easy-an-iot-project-with-esp32-ee2777(дата звернення: 07.11.2024).

33. IoT-based Indoor Environment Monitoring With ESP32 and DHT22 (Adafruit IO) URL: https://www.instructables.com/IoT-based-Indoor-Environment-Monitoring-With-ESP32/ (дата звернення: 05.11.2024).

34. Real-Time Sensor Monitoring with ESP32 & FreeRTOS. URL: https://medium.com/engineering-iot/building-a-real-time-sensor-monitoring-system-with-esp32-and-freertos-b4e4f214df8e (дата звернення: 30.11.2024).

35. Sinambela M., Nugraha M.R., Widodo A. IoT-Based Air Quality Monitoring System Design and Development Using ESP32. *International Journal of Advanced Computer Science and Applications* 2021. Vol. 12(5). pp. 123–130.

36. ESP32-Based Environmental Monitoring System URL: https://docs.cirkitdesigner.com/project/published/2e5030d1-8fa5-4a3e-9079-bed7b152134d/esp32-based-environmental-monitoring-system (дата звернення: 15.11.2024).

37. Hercog D., Lerher T., Truntič M., Težak O. Design and Implementation of ESP32-Based IoT Devices. *Sensors*, 2023. Vol. 23(15). pp. 6739.

38. Babic D., Jovovic I., Popovic T., Kovac N., Cakic S. An Internet of Things System for Environmental Monitoring Based on ESP32 and Blynk. *2022 26th International Conference on Information Technology (IT)*, 2022. pp. 1–6.

39. Adeagbo A.A. IOT Based Environment Monitoring System Using ESP32. *arXiv*, 2024.. 6 p.

40. Mota A., Serôdio C., Briga-Sá A., Valente A. Implementation of an Internet of Things Architecture to Monitor Indoor Air Quality: A Case Study During Sleep Periods. *Sensors*, 2025. Vol. 25(6). pp. 1683.

41. Sundararajan M., Rajasekaran M.P., Ramesh R., Ramesh K. Development of an IoT-Enabled Air Pollution Monitoring and Air Purifier System. *MAPAN*, 2023. Vol. 38. pp. 669–688.

42. Fahim M., El Mhouti A., Boudaa T., Jakimi A. Modeling and Implementation of a Low-Cost IoT-Smart Weather Monitoring Station and Air Quality Assessment Based on Fuzzy Inference Model and MQTT Protocol. *Modeling Earth Systems and Environment*, 2023. Vol. 9. pp. 4085–4102.

43. Hercog D., Lerher T., Truntič M., Težak O. Design and Implementation of ESP32-Based IoT Devices. *Sensors*, 2023. Vol. 23(15). pp. 6739.

44. Khan A.U., Khan M.E., Hasan M., Zakri W., Alhazmi W., Islam T. An Efficient Wireless Sensor Network Based on the ESP-MESH Protocol for Indoor and Outdoor Air Quality Monitoring. *Sustainability*, 2022. Vol. 14(24). pp. 16630.

45. Ribeiro J., Costa A., Ferreira J., Silva J., Gomes A., Rodrigues J. Development of a Unified IoT Platform for Assessing Meteorological and Air Quality Data in a Tropical Environment. *Sensors*, 2024. Vol. 24(9). pp. 2729.

46. Rahmadani A.A., Syaifudin Y.W., Setiawan B., Panduman Y.Y.F., Funabiki N. Enhancing Campus Environment: Real-Time Air Quality Monitoring Through IoT and Web Technologies. *Journal of Sensor and Actuator Networks*, 2025. Vol. 14(1). pp. 2.

47. Paredes J., Ruales D., Vaca C., Cevallos D., Ruales M., Paredes J. Smart and Portable Air-Quality Monitoring IoT Low-Cost Devices in Ibarra City, Ecuador. *Sensors*, 2022. Vol. 22(18). pp. 7015.

48. El Anshori Y.T., Kunda R.M., Manuhutu F. Design and Construction of a Real-Time Air Quality Monitoring System Using IoT-Based ESP32 to Strengthen Environmental Policies. *Jurnal Penelitian Pendidikan IPA (JPPIPA)*, 2025. Vol. 10(1). pp. 1–7.

49. Megantoro P., Aldhama S.A., Prihandana G.S., Vigneshwaran P. IoT-based weather station with air quality measurement using ESP32 for environmental aerial condition study. *TELKOMNIKA Telecommunication Computing Electronics and Control.* 2023. Vol. 21, No. 3. pp. 737–746.

50. Kusuma H.A., Oktavia D., Nugaraha S., Suhendra T., Refly S. Sensor BMP280 Statistical Analysis for Barometric Pressure Acquisition. *IOP Conference Series: Earth and Environmental Science.* 2023. Vol. 1148. 012008. pp. 1–7.

51. Satria B., Rahmaniar R., Dalimunthe M.E., Berthauli S., Iqbal M. A Development IOT Based Real-Time Weather Monitoring System Using NodeMCU ESP32 and BMP280-DHT11 Sensor. *INFOKUM: Jurnal Ilmu Informasi dan Komunikasi*. 2023. Vol. 12, No. 1. pp. 74–82.

52. Satria B., Rahmaniar R., Dalimunthe M.E. An Implementation IoT Weather Station Based on ESP32. *PENDIDIKAN: Jurnal Pendidikan dan Pengajaran.* 2023. Vol. 8, No. 2. pp. 123–130.

53. Easterline L.M., Putri A.A.R., Atmaja P.S., Dewi A.L., Prasetyo A. Smart Air Monitoring with IoT-based MQ-2, MQ-7, MQ-8, and MQ-135 Sensors using NodeMCU ESP32. *Procedia Computer Science.* 2024. Vol. 232. pp. 1891–1898.

54. Macheso P., Chisale S.W., Daka C., Dzupire N., Mlatho J., Mukanyirigira D. Design of Standalone Asynchronous ESP32 Web-Server for Temperature and Humidity Monitoring. *Proceedings of the 2021 International Conference on Advances in Computing, Communication and Control Systems (ICACCS)*. 2021. pp. 1–6.

55. Ansari S., Ansari A., Kumar A., Kumar R., Nyamasvisva E.T. Environmental Temperature and Humidity Monitoring at Agricultural Farms using Internet of Things & DHT22-Sensor. *Journal of Innovative Scientific Research and Communication.* 2023. Vol. 2, No. 1. pp. 19–27.

56. Pandya H., Limbasiya C., Vachhani K., Singh S. Low-Cost IoT Air Quality Monitoring Station Using Cloud Platform. *Applied Sciences.* 2024. Vol. 14, No. 13. Article 5774. pp. 1–17.

57. Singh S., Das M., Manna A., Ghosh R. Modeling and Implementation of a Low-Cost IoT-Smart Weather Station. *Sensors (Basel).* 2023. Vol. 23, No. 2. Article PMC9901407. pp. 1–14.

58. IoT-Enabled Mini Pollution Monitoring System with App & Website // GitHub. 2025. URL: https://github.com/pratz222/IoT-Enabled-Mini-Pollution-Monitoring-System-with-App-Website (дата звернення: 05.12.2024).

59. . Rahman S., Ali M., Islam M.R., Kabir M.H. Design and Implementation of a Smart Environmental Monitoring System Using IoT and Cloud Computing.

International Journal of Advanced Computer Science and Applications. 2024. Vol. 15, No. 1. pp. 97–104.

60. Patel S., Desai R. Development of an IoT-Based Air Quality Monitoring System Using ESP32 and MQTT Protocol. *International Journal of Engineering Research & Technology.* 2024. Vol. 13, Issue 1. pp. 140–145.

61. . Shaikh F., Shaikh R. Smart Agriculture Monitoring System Using IoT and Cloud Computing. *International Journal of Computer Applications.* 2024. Vol. 183, No. 37. pp. 15–20.

62. Gaurav S., Jain R. IoT-Based Real-Time Environmental Monitoring System Using ESP32 and Firebase. *International Journal of Innovative Research in Computer and Communication Engineering.* 2024. Vol. 12, Issue 1. pp. 90–95.

63. . Khan M.S., Shaikh A.A., Pathan M. Design and Implementation of a Low-Cost IoT-Based Weather Monitoring System. *International Journal of Scientific & Engineering Research.* 2024. Vol. 15, Issue 2. pp. 255–260.

64. Raj A. IoT-Based Smart Air Quality Monitoring System Using ESP32 and Blynk. *International Journal of Computer Applications.* 2024. Vol. 183, No. 45. pp. 8–13.

65. Sharma P., Mehta V., Gupta R. Development of an IoT-Based Smart Environmental Monitoring System Using ESP32 and ThingSpeak. *International Journal of Advanced Research in Computer and Communication Engineering.* 2024. Vol. 13, Issue 1. pp. 112–117..

66. MongoDB. Real-Time IoT Data Analytics Solution. URL: https://www.mongodb.com/solutions/solutions-library/real-time-iot-data-analytics-solution (дата звернення: 07.12.2024).

67. Qureshi A.M., Khan F.S. Design and Implementation of an IoT-Based Weather Monitoring System Using ESP32 and DHT22 Sensor. *International Journal of Engineering Research & Technology.* 2024. Vol. 13, Issue 1. pp. 146–151.

68. Raj A. IoT-Based Smart Environmental Monitoring System Using ESP32 and Cloud Computing. *International Journal of Computer Applications.* 2024. Vol. 183, No. 45. pp. 14–19.

69. Sharma A., Verma P. Development of an IoT-Based Air Quality Monitoring System Using ESP32 and Firebase. *International Journal of Innovative Research in Computer and Communication Engineering.* 2024. Vol. 12, Issue 1. pp. 96–102.

70. Kamble S.B., Rao P.R.P., Pingalkar A.S., Chayal G.S. IoT based weather monitoring system. *International Journal of Advanced Research in Innovative Ideas and Education.* 2017. Vol. 3, No. 2. pp. 2886–2991.

71. Bhagat A.M., Thakare A.G., Molke K.A., Muneshwar N.S., Choudhary V. IoT based weather monitoring and reporting system project. *International Journal of Trend in Scientific Research and Development (IJTSRD).* 2019. Vol. 3, No. 3. pp. 1–3.

72. Kaewwongsri K., Silanon K. Design and implement of a weather monitoring station using CoAP on NB-IoT network. *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON).* 2020. pp. 230–233.

73. Tiwari M., Narang D., Goel P., Gadhwal A., Gupta A., Chawla A. Weather monitoring system using IoT and cloud computing. *Weather.* 2020. Vol. 29, No. 12s. pp. 2473–2479.

74. Math R.K.M., Dharwadkar N.V. IoT Based low-cost weather station and monitoring system for precision agriculture in India. *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud).* 2018. pp. 81–86.

75. Zafar S., Miraj G., Baloch R., Murtaza D., Arshad K. An IoT-based real-time environmental monitoring system using Arduino and cloud service. *Engineering, Technology & Applied Science Research.* 2018. Vol. 8, No. 4. pp. 3238–3242.

76. Joseph F.J.J. IoT-based weather monitoring system for effective analytics. *International Journal of Engineering and Advanced Technology.* 2019. Vol. 8, No. 4. pp. 311–315.

77. Saini H., Thakur A., Ahuja S., Sabharwal N., Kumar N. Arduino based automatic wireless weather station with remote graphical application and alerts. *International Conference on Signal Processing and Integrated Networks.* 2016.

78. Saini H., Thakur A., Ahuja S., Sabharwal N., Kumar N. Arduino based automatic wireless weather station with remote graphical application and alerts. *International Conference on Signal Processing and Integrated Networks.* 2016.

79. Shahadat A.S.B., Ayon S.I., Khatun M.R. Efficient IoT based Weather Station. *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE).* 2020.

80. Singh T., Asim M. Weather Monitoring System Using IoT. *Innovations in Cyber Physical Systems. Lecture Notes in Electrical Engineering.* 2021. Vol. 788. pp. 721–730.

81. Sheth M., Rupani P. Smart Gardening Automation using IoT With BLYNK App. *International Conference on Trends in Electronics and Informatics.* 2019. pp. 266–270.

82. Головна сторінка IQAir. URL: https://www.iqair.com/ (дата звернення: 04.12.2024).

ДОДАТОК А
(обов’язковий)

ОСНОВНА СТРУКТУРА СЕНСОРНОЇ СИСТЕМИ

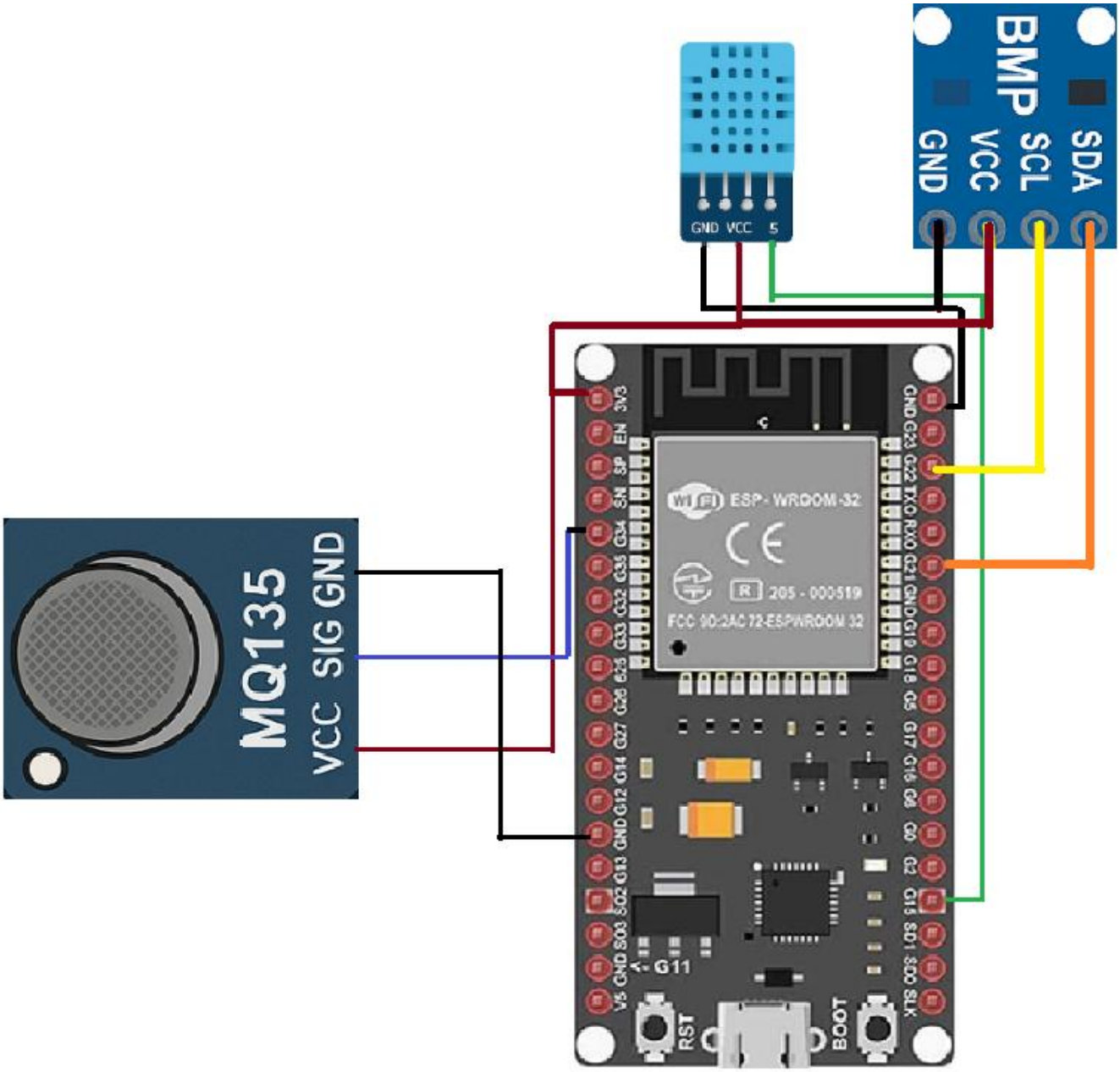


Рисунок А.1 – Структура сенсорної системи

ДОДАТОК Б

(обов'язковий)

КОПІЯ ПУБЛІКАЦІЇ У НАУКОВОМУ ВИДАННІ

Актуальні проблеми комп'ютерних наук

УДК 004.4

Бендій Д.М.

Хмельницький національний університет

СИСТЕМА МОНІТОРИНГУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА НА ОСНОВІ ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ

Розглянуто прикладні аспекти розробки розподіленої системи моніторингу навколишнього середовища на основі технології Інтернету речей (IoT) та візуалізації отриманих результатів для подальшого аналізу, що забезпечує найбільш точне передання змісту вхідних даних. Запропонована система забезпечує точну і швидку обробку даних у режимі реального часу для своєчасного реагування на зміни екологічної ситуації.

Applied aspects of the development of a distributed environmental monitoring system based on the Internet of Things (IoT) technology and the visualization of the obtained results for further analysis are considered, which provides the most accurate transmission of the content of the input data. The offered system provides accurate and fast data processing in real time to enable timely responses to environmental changes.

З розвитком нових технологій та підвищенням рівня інформатизації суспільства проблема моніторингу навколишнього середовища набуває особливого значення.

Погодні явища, зміна клімату та людська діяльність є взаємопов'язаними аспектами, які впливають на якість і доступність природних ресурсів, таких як вода, повітря, ґрунт, і загальний стан екосистем, а також умови життя окремих людей і громад, впливаючи на наше благополуччя. буття та інфраструктури. [1]

Згідно з Copernicus, Програмою спостереження за Землею Європейського Союзу, 2023 рік підтверджено як найтепліший календарний рік у даних про глобальну температуру, починаючи з 1850 року .[2]

По мірі того, як ми рухаємося вперед, стає обов'язковим продовжувати працювати над сталими практиками для побудови збалансованих відносин між людством і планетою. З огляду на ці виклики, моніторинг навколишнього середовища стає важливим союзником у пом'якшенні та подоланні наслідків глобального потепління та стихійних лих.

Метою роботи є розробка розподіленої системи моніторингу на основі технології Інтернету речей (IoT), що забезпечує точне збирання, обробку та аналіз даних.

Система використовує різні датчики для збору екологічних даних і забезпечує своєчасне реагування на зміни в довкіллі. Важливі аспекти включають забезпечення безпеки даних, масштабованість та ефективність роботи системи.

Різноманітний набір датчиків становить основу мереж IoT, серед них:

- датчики температури: вимірювання умов навколишнього середовища;
- датчики вологості: контролюйте рівень вологи в повітрі;
- датчики наближення: виявляють наявність або відсутність об'єктів у певному діапазоні;
- датчики руху: відстежуйте рух або зміни положення;
- датчики світла: вимірювання інтенсивності світла;
- датчики тиску: оцінюють атмосферний тиск або тиск води;
- акселерометри: визначають швидкість і напрямок руху;
- датчики газу: ідентифікують різні гази в навколишньому середовищі [3].

Система передбачає розміщення датчиків в різних точках для збору даних про стан навколишнього середовища.

Основні компоненти системи включають датчики, мережу передачі даних та централізовану платформу для аналізу та візуалізації даних (таблиця 1).

Таблиця 1 – Основні компоненти системи моніторингу

Компонент	**Функції**
Датчики	Збір даних про стан навколишнього середовища
Мережа передачі	Передача зібраних даних до центральної платформи
Центральна платформа	Обробка, аналіз та візуалізація отриманих даних

Отже, запропонована система моніторингу на основі IoT забезпечує точну і швидку обробку екологічних даних у реальному часі. Подальші дослідження спрямовані на покращення автоматизації та інтеграції системи в існуючі екологічні програми.

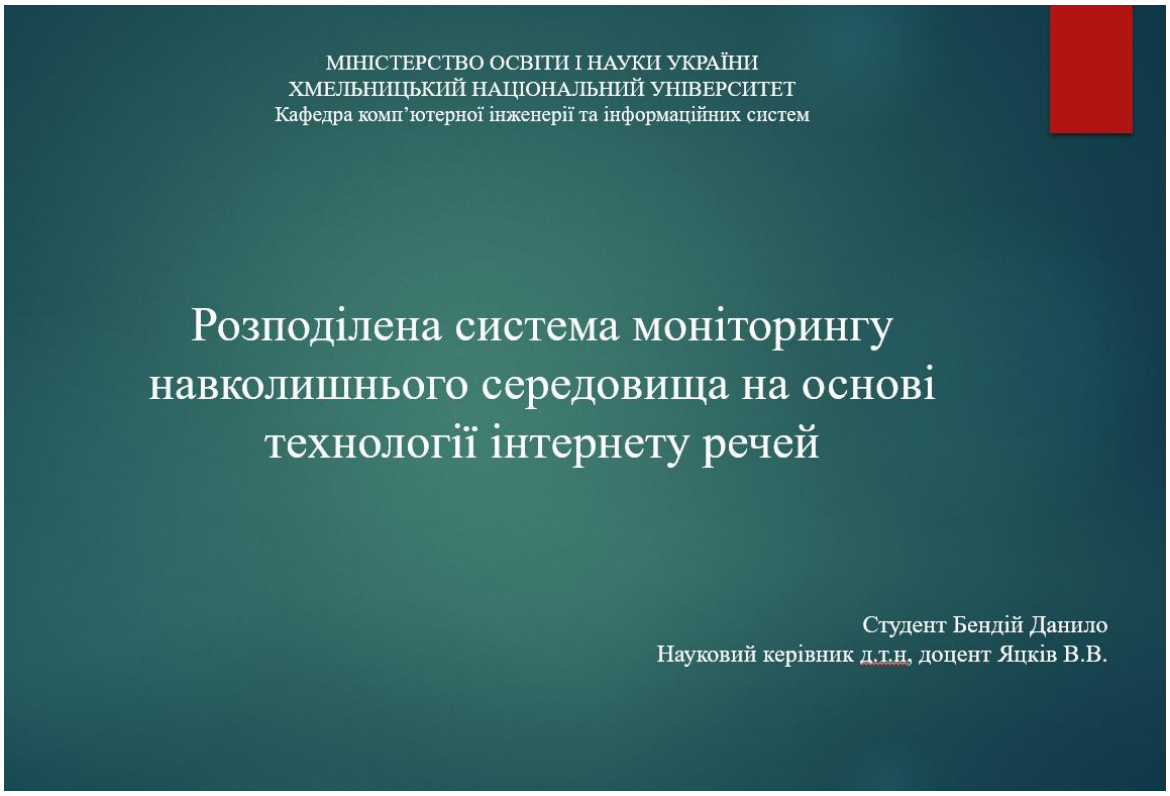
Перелік посилань

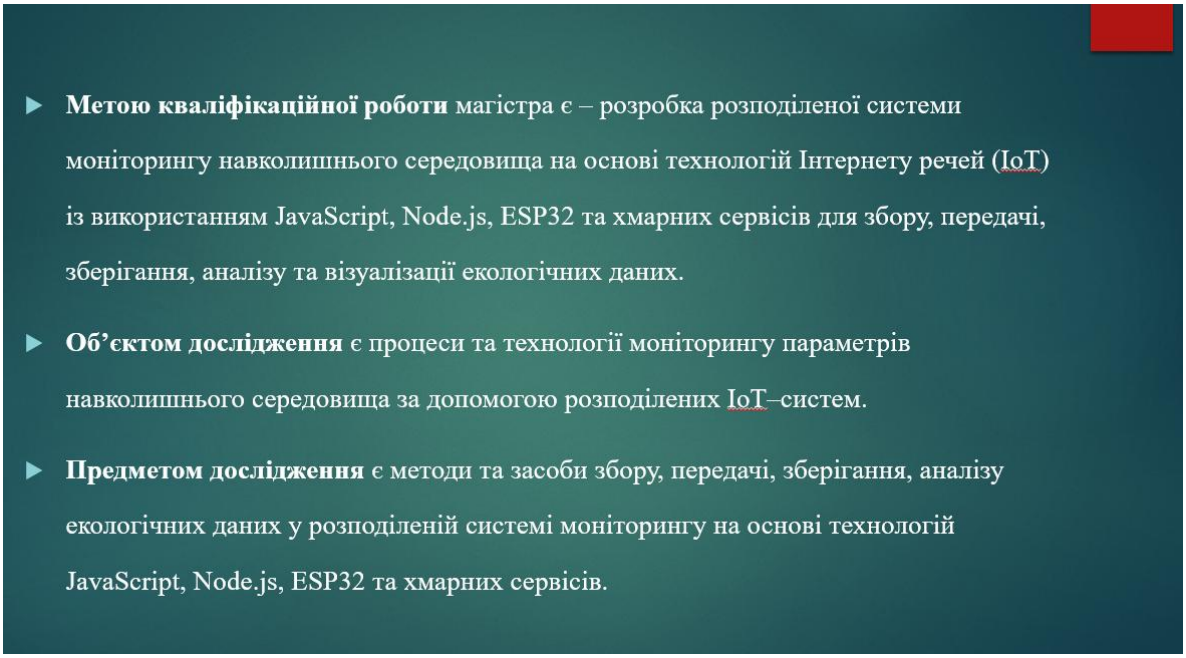
1. Everything you need to know about IoT Environmental Monitoring. URL: https://loriot.io/blog/environmental-monitoring-01.html
2. Global Climate Highlights 2023. URL: https://climate.copernicus.eu/global-climate-highlights-2023#:~:text=2023%20marks%20the%20first%20time,than%202%C2%B0C%20warmer.
3. IoT and Environmental Monitoring with Sensor Networks. URL: https://www.iotforall.com/iot-and-environmental-monitoring-with-sensor-networks

ДОДАТОК В

(обов'язковий)

ПРЕЗЕНТАЦІЯ РОБОТИ





НАУКОВА НОВИЗНА ОТРИМАНИХ РЕЗУЛЬТАТІВ

- Набув подальшого розвитку метод інтеграції багатосенсорних IoT-систем для моніторингу якості повітря в режимі реального часу, що дозволяє підвищити точність та оперативність виявлення екологічних загроз.
- Набула подальшого розвитку інформаційна технологія збору, обробки та візуалізації екологічних даних з використанням мікроконтролера ESP32 та хмарних платформ, що забезпечує масштабованість та доступність системи для широкого кола користувачів.

ПРАКТИЧНА ЗНАЧИМІСТЬ ОТРИМАНИХ РЕЗУЛЬТАТІВ

- Практична значимість отриманих результатів полягає у тому що розроблена система являється доступнішою та енергоефективнішою IoT-системою, за існуючі аналоги. Та може бути використана в різних сферах, включаючи освіту, охорону здоров'я, промисловість та міське планування. Розроблена система дозволяє оперативно виявляти перевищення допустимих рівнів забруднення повітря, що сприяє своєчасному прийняттю заходів для захисту здоров'я населення та навколишнього середовища

ПУБЛІКАЦІЯ

- Опубліковано публікацію у Збірнику наукових праць за матеріалами XVI Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2024». (Хмельницький – 2024. – С. 46-47).

- На даний момент, з розвитком промисловості та урбанізації, зростає потреба у безперервному спостереженні за параметрами середовища, такими як температура, вологість, якість повітря тощо. Використання IoT дозволяє автоматизувати процес збору, передачі та аналізу даних у режимі реального часу, що є важливим для екологічного моніторингу, розумних міст та промислових застосувань.
- Висока затребуваність породжує необхідність в доступних рішеннях доступних, для багатьох сфер діяльності та верств населення, що і стало викликом в основі цієї роботи.

ОСНОВНІ КОМПОНЕНТИ

BMP280

ESP32 DevKit V1

DHT22

MQ135

ОСНОВНА СТРУКТУРА СЕНСОРНОЇ СИСТЕМИ

```
“connections”: [
["esp:15","dht:S", "green"],
["esp:3V3", "dht:VCC", "red"],
["esp:GND", "dht:GND","black"],

["esp:34", "mq135:SIG","blue"],
["esp:3V3", "mq135:VCC","red"],
["esp:GND", "mq135:GND","black"],

["esp:21", "bmp:SDA","orange"],
["esp:22", "bmp:SCL","yellow"],
["esp:3V3", "bmp:VCC","red"],
["esp:GND", "bmp:GND","black"], ]
```

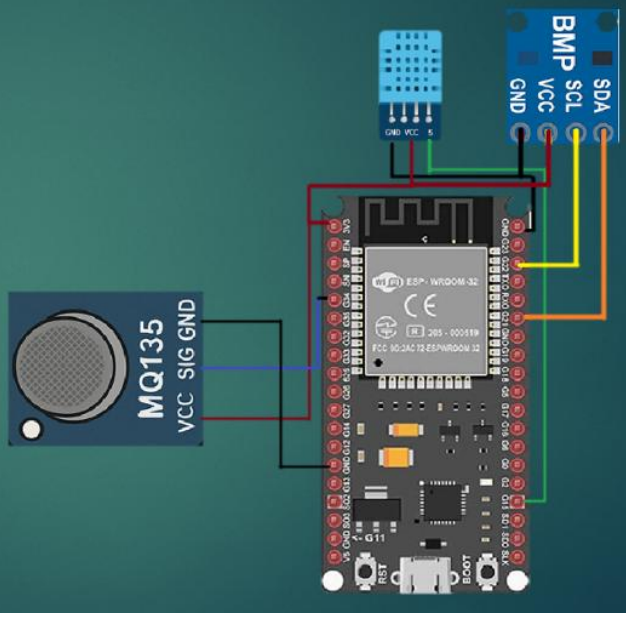
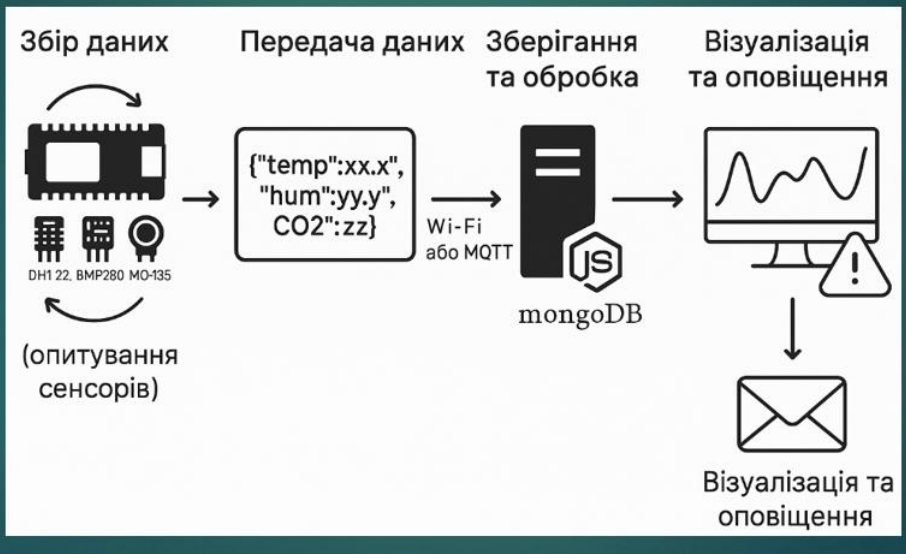


СХЕМА ЗАГАЛЬНОГО АЛГОРИТМУ РОБОТИ СИСТЕМИ



ВИСНОВКИ

- У ході виконання дипломної роботи було реалізовано розподілену систему моніторингу навколишнього середовища, яка ґрунтується на сучасних IoT-рішеннях, поєднуючи апаратні компоненти, бездротові технології, веб-інтерфейси та хмарні сервіси. Основною метою роботи стало створення ефективної, масштабованої та доступної системи для оперативного спостереження за змінами довкілля в режимі реального часу.
- На першому етапі було проведено аналіз існуючих рішень у сфері екологічного моніторингу, що дозволило сформувати технічні вимоги до майбутньої системи. Визначено основні виклики, серед яких точність збору даних, затримки при передачі, адаптація під змінні умови навколишнього середовища та оптимізація витрат на впровадження. Обґрунтовано доцільність вибору мікроконтролера ESP32 як обчислювального ядра, а також обрано перевірені сенсори, що відповідають вимогам до надійності, енергоефективності й точності.
- У процесі реалізації проєкту створено архітектуру системи з чітким поділом на сенсорний, серверний та інтерфейсний рівні. Розроблено логіку обміну повідомленнями між компонентами, налаштовано серверну частину на Node.js з підтримкою MongoDB, розроблено веб-інтерфейс із використанням React і Chart.js для візуалізації даних, а також впроваджено механізм сповіщень при виході параметрів за критичні межі.

- Результати практичного тестування підтвердили життєздатність розробленої системи. Вона демонструє високу точність зчитування показників, стабільність передачі даних та адаптивність до умов реального середовища. Оновлення інформації відбувається з мінімальною затримкою, що дозволяє майже миттєво реагувати на зміни довкілля. Витрати на реалізацію системи залишаються в межах доступного бюджету, що відкриває перспективи її впровадження у школах, громадах та дослідницьких центрах.
- У результаті, запропонована система стала прикладом ефективного поєднання апаратного і програмного забезпечення в межах концепції Інтернету речей, демонструючи потенціал використання подібних рішень для поліпшення екологічного моніторингу та цифровізації навколишнього простору.

ДЯКУЮ ЗА УВАГУ

Anti-Plagiarism v-15.274 Educational

The maximum coincidence with one document 1.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 13%**

ID: 240838 Title: МКР Розподілена система моніторингу навколишнього середовища на основі технології інтернету речей Added in a DB: 2025-05-05 Authors: Д. М. Бендій Heads: В.В. Яцків Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	109821	836	1752 (2%)	23 (3%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Д. М. Бендій

Співавтор:

Назва: Бендій_Розподілена система моніторингу навколишнього середовища на основі технології інтернету речей

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:5.5%

Коефіцієнт подібності 2:0.4%

Мікропробіли: 0

Заміна букв: 1

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-05-05 17:16:38.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-05-06 Доцент Андрій Нічепорук

Дата експерт

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Бендій Данило Михайлович

Тема: Розробка розподіленої системи моніторингу навколишнього середовища на основі технології Інтернету речей

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість сторінок записки 78

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи магістра є розробка розподіленої системи моніторингу навколишнього середовища на основі технологій Інтернету речей із використанням JavaScript, Node.js, ESP32 та хмарних сервісів для збору, передачі, зберігання, аналізу та візуалізації екологічних даних.

2. Висновок про відповідність роботи дипломному завданню: Кваліфікаційна робота відповідає поставленому завданню і охоплює всі його ключові пункти.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У розділі 1 розглянуто сучасні системи моніторингу та обґрунтовано використання IoT; у розділі 2 підібрано компоненти та сенсори, та розроблено алгоритми і математичну модель системи; у розділі 3 описано розробку архітектури програмного забезпечення, клієнтської та серверної частини; у розділі 4 подано тестування, візуалізацію даних і розрахунок вартості.

4. Позитивні сторони роботи: проєкт має практичну цінність, демонструє бачову інтеграцію програмного та апаратного забезпечення, а також гнучкість до масштабування. Система передбачає зручний веб-інтерфейс для моніторингу в реальному часі.

5. Негативні сторони роботи: робота місцями містить поверхневі технічні обґрунтування, недостатню глибину аналізу альтернативних рішень, а також спостерігається обмежене використання наукових джерел у теоретичних розділах.

Архітектура рішення описана загально, без достатньої деталізації структур бази даних або внутрішніх протоколів.

6. Оцінка графічного оформлення та пояснювальної записки роботи: оформлення відповідає вимогам, однак потребує покращення в частині структуризації. Графічний матеріал представлений, але має обмежену інформативність.

7. Відгук про роботу в цілому: робота виконана на базовому технічному рівні. Вона демонструє розуміння основних принципів побудови IoT-систем, проте не повною мірою реалізовані всі технічні можливості запропонованої архітектури. Складові роботи мають практичну значимість, але потребують подальшої оптимізації та вдосконалення.

8. Інші зауваження: робота потребує покращення з точки зору стилістики та мовного оформлення. В окремих випадках не повністю витримана структура академічного письма.

9. Оцінка дипломної роботи: Враховуючи викладене, вважаю, що кваліфікаційна робота заслуговує оцінки «добре» (4.00, С).

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Пирлик Ю.В. д.т.н. професор, завідувач кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки.

"21" 05 2025 р. _____ (підпис)

Завідувачу кафедри КІС
доктору філософії, доцент
Ользі ПАВЛОВІЙ

Бендія Данила Михайловича

ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-23-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (StrikePlagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 квітня 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Розподілена система моніторингу навколишнього середовища на основі технології інтернету речей

Автор: Бендій Данило Михайлович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Яцків В.В. д.т.н., професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення мають належним чином оформлені посилання;
2) окремі виявлені збіги с загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.(Тут текст можна і треба модифікувати)

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 5.5% і адресується до 42 першоджерел; та системою Anti-Plagiarism складає 1.0%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи — Василь ЯЦКІВ

Гарант ОП — Олег САВЕНКО

Завідувач кафедри КІІС — Ольга ПАВЛОВА