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

КВАЛІФІКАЦІЙНА РОБОТА

Кіберфізична система дистанційного контролю рівня шуму в міському середовищі
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 2302128.23.02.34. ПЗ

Виконав здобувач ІІІ курсу, група КІ2с-23-2 _____ Олександра ВДОВИНА
Підпис Ініціали, прізвище

Керівник _____ Ольга АТАМАНЮК
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер _____ Сергій ЛИСЕНКО
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ПЕРШИЙ (БАКАЛАВРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Вдовиній Олександрі Вікторівні
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система дистанційного контролю рівня шуму в міському середовищі
Керівник проекту (роботи) Атаманюк Ольга Вадимівна, асистент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 5

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз проблеми акустичного забруднення міст та постановка технічного завдання
Проєктування інформаційної моделі сховища SQLite, архітектури та алгоритмів кіберфізичної системи
Програмно-апаратна реалізація вузла ESP32, сервера Flask, Ngrok-тунелювання та верифікація комплексу

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Схема структурно-топологічна апаратного забезпечення.
Інформаційна модель та серверна інфраструктура
Схема алгоритмів функціонування та верифікації

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання «_10_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконано
3	Робота над розділом 1 – теоретичні основи досліджуваної проблеми	01.03.2026	виконано
4	Робота над розділом 2 – проектування архітектури, інформаційної моделі та алгоритмів кіберфізичної системи	01.04.2026	виконано
5	Робота над розділом 3 – програмно-апаратна реалізація кіберфізичної системи дистанційного контролю рівня шуму	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач_____ ___Олесандра ВДОВИНА___
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи_____ ___Ольга АТАМАНЮК___
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 2302128.23.02.34 ПЗ	Пояснювальна записка	65		
			Графічні матеріали			
2		КвРКІ 2302128.23.02.34 Е8	Схема структурно	1		
			топологічна апаратного			
			забезпечення			
3		КвРКІ 2302128.23.02.34 Е8	Інформаційна модель та	1		
			серверна інфраструктура			
4		КвРКІ 2302128.23.02.34 Е8	Схема алгоритмів	1		
			функціонування та			
			верифікації			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 2302128.23.02.34 ВП		
Розробив		Вдовина		02.06	Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Атаманюк		02.06		У	1	1
						ХНУ, КІ2с-23-2		
Н. контр.		Лисенко		02.06				
Затв.		Павлова		02.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Кіберфізична система дистанційного контролю рівня шуму в міському середовищі».

Автор роботи: Олександра ВДОВИНА.

Керівник роботи: Ольга АТАМАНЮК.

Пояснювальна записка: 65 с., 16 рис., 4 дод., 50 джерел.

Графічна частина: 3 креслення.

АКУСТИЧНЕ ЗАБРУДНЕННЯ, БАЗА ДАНИХ, ЕКОЛОГІЧНИЙ МОНІТОРИНГ, ІНТЕРНЕТ РЕЧЕЙ, КІБЕРФІЗИЧНА СИСТЕМА.

Кваліфікаційна робота бакалавра присвячена розробці та дослідженню кіберфізичної системи дистанційного контролю рівня шуму в міському середовищі на базі мікроконтролерної платформи та веб-технологій. Актуальність теми зумовлена стрімкими темпами урбанізації та зростанням рівня акустичного забруднення у великих містах, що безпосередньо впливає на здоров'я та комфорт мешканців. Своєчасний безперервний контроль звукового тиску в різних зонах міста дає змогу оперативно виявляти перевищення санітарних норм, накопичувати статистичну інформацію для аналізу екологічної ситуації та приймати обґрунтовані управлінські рішення в межах концепції «розумного міста».

Метою роботи є проєктування, реалізація та тестування апаратно-програмного комплексу для збору, передавання, транзакційного збереження й інтерактивної візуалізації даних про шумове навантаження в режимі реального часу. Для досягнення поставленої мети було виконано аналіз технологій Інтернету речей (IoT) у сфері екологічного моніторингу, обрано апаратну платформу, розроблено алгоритми перетворення аналогових сигналів, спроєктовано структуру реляційної бази даних, а також розроблено серверну архітектуру та користувацький графічний інтерфейс для перегляду аналітики й тривожних сповіщень.

_____ 30.05.2026

Підпис здобувача Дата

ЗМІСТ

Зм.	Арк.	№докум.	Підпис	Дата	КвРКІ. 2302128.23.02.34 ПЗ		
Виконав		Олександра ВДОВИНА		02.06	Кіберфізична система дистанційного контролю рівня шуму в міському середовищі. Пояснювальна записка	Літера: у	Аркуш: 1
Перевір.		Ольга АТАМАНЮК		02.06			Аркушів: 65
Н.контр.		Сергій ЛИСЕНКО		02.06		ХНУ КІ2с-23-2	
Затвер.		Ольга ПАВЛОВА		02.06			

ВСТУП

Сучасні процеси стрімкої урбанізації, постійне ущільнення житлової забудови та активне розширення транспортних і промислових мереж неминуче призводять до суттєвого погіршення акустичної обстановки у великих містах. Сьогодні шумове забруднення визнано однією з найнебезпечніших екологічних загроз прихованого характеру, яка через свій невидимий фізичний вплив тривалий час недооцінювалася суспільством, проте зараз беззаперечно доведено її шкоду для фізіологічного та психоемоційного стану містян. Розробка та впровадження новітніх автоматизованих комплексів дистанційного моніторингу, що базуються на архітектурі Інтернету речей та технологіях граничних обчислень, є критично необхідним інженерним кроком для формування безпечного та комфортного міського простору в межах концепції «Розумного міста».

Об’єктом даного дослідження виступають процеси безперервного автоматизованого збору, обробки та аналізу даних щодо рівня акустичного забруднення в урбанізованому середовищі. Предметом дослідження є програмно-апаратні засоби, алгоритмічне забезпечення та архітектурні рішення, необхідні для створення ефективної кіберфізичної системи дистанційного моніторингу шуму. Головна мета роботи полягає у проєктуванні, розробці та створенні функціонального прототипу комплексу, здатного в режимі реального часу дистанційно вимірювати, надійно зберігати у базі даних та інтерактивно візуалізувати показники шумового навантаження.

Запропоноване технічне рішення, побудоване на базі мікроконтролерів ESP32, веб-фреймворку Flask та системи управління базами даних SQLite, має вагоме практичне значення. Його можна успішно впроваджувати в муніципальні екологічні ініціативи для контролю міських магістралей, використовувати для фонового моніторингу умов праці на промислових виробництвах чи будівельних майданчиках.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз структурних і функціональних особливостей системи

Сучасний етап розвитку людської цивілізації нерозривно пов'язаний із процесами глобальної урбанізації. Стрімке розширення меж великих міст, зростання щільності населення, ущільнення житлової забудови та колосальна інтенсифікація використання міської інфраструктури формують абсолютно нове середовище існування людини. Перетворення традиційних міст на складні мегаполіси приносить беззаперечні економічні, соціальні та культурні переваги, проте одночасно породжує низку критичних екологічних викликів. Серед усього спектра проблем, пов'язаних із деградацією міського середовища, дедалі більшої гостроти та масштабності набуває акустичне, або шумове забруднення. На відміну від забруднення повітря чи води, шумове забруднення має невидиму фізичну природу, через що його руйнівний вплив тривалий час недооцінювався як суспільством, так і органами муніципального управління. Проте сьогодні наукова спільнота визнає надмірний міський шум одним із найнебезпечніших чинників, що систематично знижує якість життя мільйонів людей.

Високий рівень шумового забруднення перестав бути просто фактором дискомфорту; він трансформувався у системний деструктивний чинник, який чинить прямий і багатогранний негативний вплив на здоров'я людини. З медичної та біологічної точок зору, тривалий вплив антропогенного шуму є потужним стресором, що порушує гомеостаз людського організму. Еволюційно слуховий апарат людини сформувався як інструмент виявлення небезпеки, тому будь-який гучний або несподіваний звук автоматично сприймається центральною нервовою системою як сигнал тривоги. В умовах сучасного міста цей механізм працює безперервно, що призводить до хронічної гіперактивності симпатичної нервової системи. Організм, перебуваючи в стані перманентної акустичної атаки, запускає каскад нейроендокринних реакцій, які

супроводжуються надлишковим виділенням гормонів стресу - кортизолу, адреналіну та норадреналіну.

Наслідки такого хронічного гормонального дисбалансу є вкрай руйнівними для серцево-судинної системи. Численні епідеміологічні дослідження доводять пряму кореляцію між проживанням у зонах з підвищеним рівнем шуму (понад 55-60 децибел) та зростанням ризику розвитку артеріальної гіпертензії, ішемічної хвороби серця та інфаркту міокарда. Постійний звуковий тиск викликає спазм кровоносних судин, збільшує частоту серцевих скорочень, підвищує в'язкість крові та сприяє розвитку ендотеліальної дисфункції. Крім того, тривалий акустичний стрес пригнічує імунну систему, роблячи мешканців міст більш вразливими до інфекційних захворювань та уповільнюючи процеси регенерації тканин.

Психоемоційна сфера людини також зазнає сильного удару від акустичного забруднення. Хронічний шум є одним із головних факторів розвитку стійкої дратівливість, агресивності та емоційного виснаження. В умовах постійного фонового гулу значно знижується здатність до концентрації уваги, погіршується оперативна пам'ять та падає загальна працездатність. Працівники офісів, розташованих поблизу жвавих магістралей, демонструють нижчу продуктивність та частіше припускаються помилок. Особливо вразливими до шуму є діти: у школярів, навчальні заклади яких знаходяться в зонах акустичного дискомфорту, фіксується уповільнення когнітивного розвитку, затримка у формуванні навичок читання, проблеми з розумінням мови та підвищений рівень тривожності. Усе це в комплексі призводить до зростання схильності міського населення до депресивних станів, неврозів та соціальної самоізоляції.

Усвідомлення колосальної шкоди, яку завдає шумове забруднення громадському здоров'ю та економіці через втрату робочих днів та витрати на лікування, вимагає від муніципальних органів впровадження жорсткої політики контролю акустичного середовища. Однак ефективне управління неможливе без

об'єктивних, точних та актуальних даних. Традиційні підходи до контролю акустичного стану міст, які застосовувалися протягом останніх десятиліть, виявилися абсолютно неадекватними масштабам сучасної проблеми. Класичний метод полягає у проведенні періодичних вимірювань за допомогою портативних шумомірів, якими оперують співробітники санітарно-епідеміологічних або екологічних служб. Цей підхід має низку фундаментальних недоліків, головним з яких є просторово-часова обмеженість вимірювань. Ручні заміри дають лише точкові, короткочасні зрізи інформації, які зазвичай тривають від кількох хвилин до півгодини. Вони не здатні відобразити складну динаміку коливань шуму протягом доби, тижня чи сезону. Крім того, такі інспекції, як правило, проводяться лише в робочий час і часто ігнорують найважливіший період - нічний час, коли санітарні норми є найсуворішими, а вплив шуму на мешканців - найболючішим.

Фізичний рівень системи формується з територіально розподіленої мережі компактних, автономних вимірювальних модулів, які можуть бути встановлені на опорах освітлення, фасадах будівель, зупинках громадського транспорту чи елементах дорожньої інфраструктури. Кожен такий вимірювальний вузол є повноцінним інтелектуальним пристроєм. Його апаратну основу становить сучасна мікроконтролерна обчислювальна платформа, до якої підключається високочутливий первинний перетворювач - мікрофонний сенсор. Датчик безперервно сприймає коливання повітряного середовища й перетворює їх на аналоговий або цифровий електричний сигнал. Важливою структурною особливістю сучасних кіберфізичних систем є концепція граничних обчислень. Згідно з цією концепцією, мікроконтролер не просто передає "сирі" дані далі по мережі, а здійснює їх складну первинну обробку безпосередньо в точці вимірювання. Мікроконтролер виконує оцифрування сигналу з високою частотою дискретизації, здійснює спектральну фільтрацію для відсікання перешкод, застосовує алгоритми частотного зважування, що імітують чутливість

людського вуха та розраховує математичний еквівалент рівня звукового тиску у децибелах (дБА) за певний проміжок часу.

Транспортний рівень архітектури відповідає за організацію надійного, безперебійного та безпечного інформаційного обміну між десятками чи сотнями периферійних вимірювальних вузлів і центральним координаційним центром. Враховуючи специфіку міської інфраструктури, передача розрахованих акустичних метрик здійснюється виключно за допомогою технологій бездротового зв'язку або мереж мобільної телеметрії. Оскільки обсяг даних, які генерує один датчик після первинної обробки, є відносно невеликим, на транспортному рівні застосовуються спеціалізовані легкотекстові протоколи передачі повідомлень. Вони дозволяють мінімізувати накладні витрати мережевого трафіку та суттєво знизити енергоспоживання комунікаційних модулів, що є критично важливим для вузлів, які працюють від автономних джерел живлення або сонячних панелей. Транспортний рівень також повинен забезпечувати гарантовану доставку даних, передбачаючи механізми тимчасової буферизації вимірювань у локальній пам'яті пристрою у випадках тимчасового зникнення зв'язку із сервером.

Прикладний рівень кіберфізичної системи розгортається на базі високопродуктивних віддалених серверів або хмарних обчислювальних кластерів. Саме тут зосереджується основний інтелектуальний потенціал системи. На цьому рівні відбувається безперервне приймання телеметричних пакетів від усіх активних датчиків міста. Дані дешифруються, валідуються та зберігаються у структурованому вигляді у спеціалізованих часорядних базах даних, які оптимізовані для роботи з безперервними потоками інформації. Серверне програмне забезпечення виконує глибоку аналітичну обробку збережених масивів: обчислює середньогодинні, денні та нічні еквівалентні рівні шуму, фіксує максимальні пікові значення, зіставляє поточні показники із законодавчо затвердженими санітарними нормами для різних функціональних зон міста, наприклад житлової, рекреаційної або промислової.

Функціональний потенціал досліджуваної кіберфізичної системи виходить далеко за межі простого збору цифр, забезпечуючи повний автоматизований цикл менеджменту акустичних даних міського середовища. До ключових функціональних завдань системи належать: повністю автономний збір звукових характеристик у режимі 24/7 без необхідності калібрування чи втручання людини; просторово-часова прив'язка кожного вимірювання геолокація та точна мітка часу, інтелектуальний аналіз трендів акустичного забруднення. Завдяки накопиченню історичних даних, система здатна виявляти добові, тижневі та сезонні закономірності шумового навантаження.

Найважливішим інструментальним функціоналом кіберфізичної системи є візуалізація аналітики та оперативне сповіщення. Прикладне програмне забезпечення надає користувачам - співробітникам міських адміністрацій, екологічних інспекцій, поліції чи навіть пересічним громадянам - доступ до інтуїтивно зрозумілих веб-інтерфейсів. Ці інтерфейси дозволяють відображати актуальну статистику у вигляді графіків, діаграм та динамічних цифрових теплових карт акустичного фону, де зони перевищення норм підсвічуються відповідними кольорами. Процес геоінформаційної інтерполяції дозволяє обчислити рівень шуму навіть у тих точках, де немає фізично встановленого датчика. Окрім візуалізації, система виконує функцію диспетчеризації: у разі виявлення критичного перевищення допустимих рівнів шуму наприклад, через незаконні нічні будівельні роботи або рух великогабаритного транспорту в заборонений час, програмне забезпечення автоматично генерує тривожні сповіщення та розсилає їх відповідальним службам для негайного реагування.

У підсумку, детальний аналіз структурних та функціональних особливостей показує, що розробка та розгортання автоматизованої кіберфізичної системи дистанційного контролю рівня шуму є єдиним адекватним та технологічно обґрунтованим рішенням проблеми акустичного забруднення сучасних мегаполісів. Відмова від неефективних ручних методів на користь інтелектуальних розподілених сенсорних мереж та хмарної аналітики

дозволяє створити безперервну, об'єктивну і високоточну картину акустичного стану міського середовища. Впровадження такої системи не лише підвищить ефективність роботи екологічних контролюючих органів, але й стане надійним фундаментом для прийняття стратегічних управлінських рішень щодо модернізації транспортної інфраструктури, просторового планування міських територій та, найголовніше, забезпечення захисту здоров'я, психоемоційного благополуччя і конституційного права громадян на безпечне для життя і здоров'я довкілля.

1.2 Аналіз програмно-апаратного забезпечення обробки інформації

Ефективність функціонування будь-якої розподіленої кіберфізичної системи дистанційного контролю рівня шуму в міському середовищі безпосередньо залежить від синергії її апаратної та програмної складових. Процес обробки інформації у такому комплексі являє собою складний, багаторівневий контур, який охоплює етапи первинної реєстрації акустичних коливань повітря, перетворення аналогового сигналу в цифровий код, математичну фільтрацію і частотне зважування, пакетування, транспортування через мережу бездротового зв'язку, а також кінцеве збереження та інтелектуальний аналіз на серверній стороні. Враховуючи специфіку експлуатації системи у відкритому урбанізованому просторі, до програмно-апаратного комплексу висуваються суворі вимоги: здатність до безперервної автономної роботи, стійкість до широкого діапазону кліматичних впливів, мінімізація енергоспоживання периферійних вузлів при збереженні високої обчислювальної спроможності, достатня пропускна здатність транспортних каналів та абсолютна надійність збереження великих масивів телеметричних часорядних даних.

Для координації роботи периферійного вузла, зчитування даних та забезпечення бездротової комунікації використовується мікроконтролерна

обчислювальна платформа, архітектура якої повинна підтримувати високу частоту дискретизації для точного охоплення всього значущого спектра міського шуму (від 16 до 44 кілогерц) відповідно до теореми Котельникова. Обчислювальний модуль повинен володіти апаратним блоком обчислень із рухомою комою (FPU) та достатнім об'ємом оперативної пам'яті для організації кільцевих буферів прямого доступу до пам'яті (DMA), що дозволяє зчитувати звуковий потік у фоновому режимі без переривання основних обчислювальних циклів. Окрім обробки звуку, мікроконтролер взаємодіє з підсистемою керування живленням, яка контролює заряд літій-іонного акумулятора від сонячної панелі, та бездротовим трансивером для передачі пакетів. Організація взаємодії всіх внутрішніх апаратних модулів периферійного термінала відображена на структурній блок-схемі (рис. 1.1).

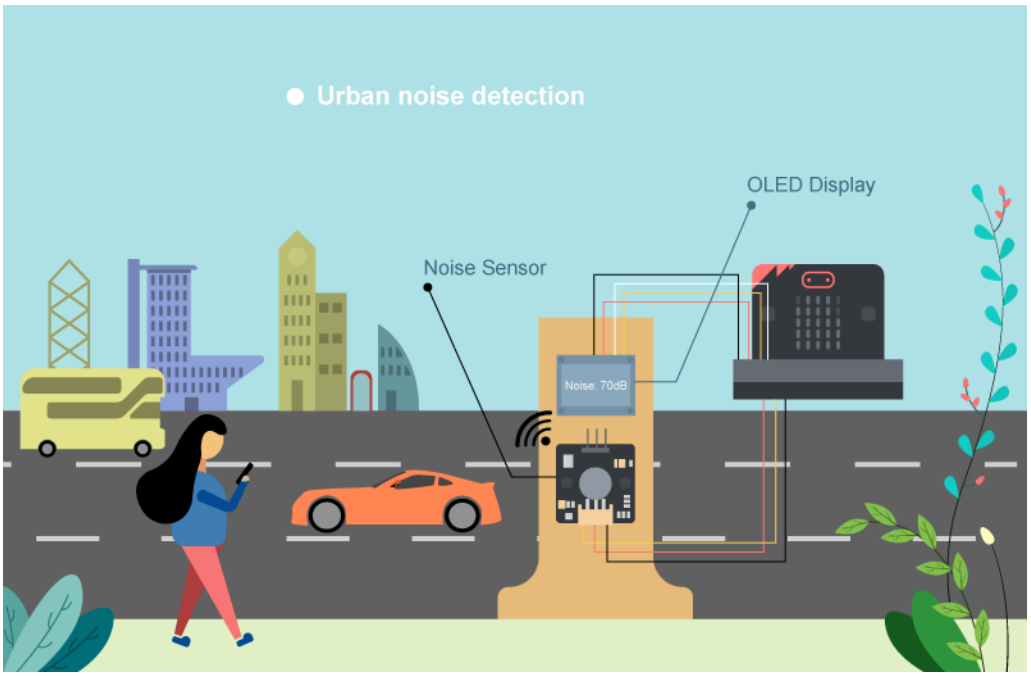


Рисунок 1.1 - Структурна блок-схема периферійного вимірювального вузла кіберфізичної системи[48]

Програмне забезпечення вбудованого рівня відповідає за первинну математичну та алгоритмічну обробку зібраної сигналограми безпосередньо на

місці вимірювання, реалізуючи парадигму граничних обчислень. Передавати нестиснений звуковий потік від сотень міських датчиків на сервер є технічно та економічно недоцільним через колосальні вимоги до пропускної здатності мережі та потенційне порушення приватності громадян. Програмний алгоритм розбиває безперервний цифровий аудіопотік на дискретні часові вікна (фрейми) та застосовує до кожного з них процедуру цифрової фільтрації. Для того щоб результати вимірювань відповідали суб'єктивному сприйняттю гучності людським вухом, програмне забезпечення реалізує частотне зважування за кривою «А». Математично це виконується шляхом застосування каскаду цифрових біквадратних фільтрів, які пригнічують низькочастотні гули (наприклад, від вітру чи вібрацій інфраструктури) і підсилюють частоти в діапазоні 1–4 кГц, до яких людське вухо є найбільш чутливим. Конструктивне та фізичне виконання автономного датчика шуму відіграє вирішальну роль у забезпеченні довговічності та стабільності роботи системи в реальних умовах міського середовища. Оскільки пристрій призначений для цілорічної експлуатації під відкритим небом на стовпах ліній електропередач, щоглах освітлення або фасадах будівель, його апаратна начинка міститься у герметичному всепогодному корпусі із класом захисту не нижче IP65, що надійно захищає мікросхеми від злив, снігу та пилу.

Після частотної фільтрації програма обчислює середньоквадратичне значення амплітуди та конвертує його в логарифмічну шкалу, отримуючи підсумковий результат у децибелах. Важливою особливістю сучасного етапу розвитку таких систем є використання концепції комплексування «злиття» даних. Оскільки швидкість поширення звуку у повітрі та фізичні властивості MEMS-мембрани залежать від параметрів навколишнього середовища, до обчислювального контуру підключаються додаткові цифрові датчики мікроклімату: температури, відносної вологості та атмосферного тиску. Дані з цих сенсорів використовуються в алгоритмі програмної температурної та барометричної компенсації для динамічного коригування розрахованого рівня

шуму. Повна схема алгоритмічної обробки, фільтрації та калібрування акустичного сигналу на граничному рівні представлена на рисунку 1.2.

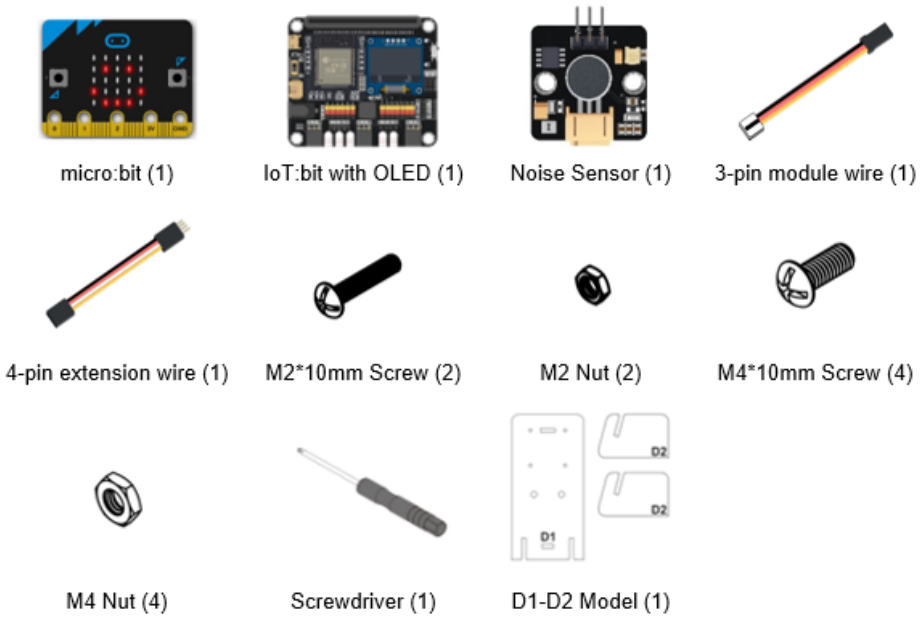


Рисунок 1.2 - Схема алгоритмічної обробки та просторово-часової фільтрації акустичних сигналів на граничному рівні [48]

Для захисту від похибок, що викликаються поривами вітру які створюють турбулентність навколо корпусу і сприймаються мікрофоном як низькочастотний гул, датчик оснащується зовнішнім сферичним вітрозахисним екраном з акустичного поролону або металевої сітки fine-mesh. Повна автономність вимірювального термінала досягається за рахунок інтеграції компактної сонячної панелі, що закріплюється під оптимальним кутом до сонця, та літій-іонного акумулятора розширеного температурного діапазону. Внутрішня архітектура датчика також включає інерційні сенсори (акселерометр та гіроскоп), які виконують антивандальну функцію, реєструючи удари або несанкціоновану зміну положення корпусу. Для надійної фіксації пристрою на елементах міської інфраструктури використовується універсальний монтажний кронштейн із нержавіючої сталі, який забезпечує стійкість до статичних та

динамічних вітрових навантажень. Конструкція кронштейна дозволяє оперативно регулювати кут нахилу сонячної панелі під час встановлення, оптимізуючи генерацію енергії залежно від географічної широти та сезонного положення сонця. Це гарантує безперебійне живлення вимірювального контуру впродовж усього періоду експлуатації комплексу. Зовнішній вигляд, компоновка та конструктивне виконання такого всепогодного автономного датчика шуму наведені на рисунку 1.3.

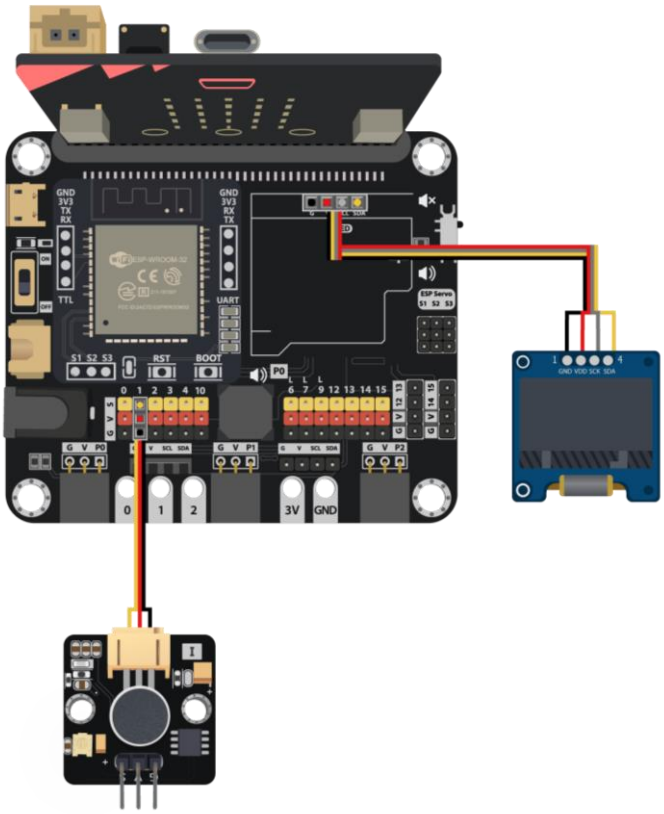


Рисунок 1.3 - Зовнішній вигляд та конструктивне виконання всепогодного автономного датчика шуму[48]

Транспортний підкомплекс кіберфізичної системи забезпечує інформаційну зв'язність між периферійними вимірювальними вузлами та центральною інфраструктурою. Вибір технології передачі даних вимагає пошуку компромісу між енергоспоживанням, дальністю зв'язку та пропускною

здатністю. У сучасних умовах доцільно розглядати переваги високошвидкісних бездротових технологій 5G та стільникових мереж LTE зокрема стандартів NB-IoT. Використання мереж п'ятого покоління відкриває можливості для підключення величезної кількості пристроїв на квадратний кілометр завдяки технології масового міжмашинного зв'язку. Беззаперечними перевагами стільникового зв'язку є низька затримка передачі пакетів та висока пропускна здатність, що дозволяє за необхідності передавати не лише розраховані метрики шуму, але й короткі фрагменти аудіозаписів для поглибленого аналізу на сервері у разі фіксації критичних аномалій. Передача даних здійснюється за допомогою енергоефективного протоколу MQTT або класичного REST API HTTP POST за допомогою JSON-пакетів.

Заключним компонентом програмного забезпечення є система візуалізації та взаємодії з користувачем Front-end, яка реалізується у вигляді веб-застосунку. Інтерфейс користувача містить набір динамічних дашбордів для відображення поточних значень шуму та модулі просторової інтерполяції. Оскільки фізично неможливо встановити датчик на кожному метрі вулиці, програмне забезпечення використовує геоінформаційні моделі для розрахунку розповсюдження звукових хвиль від точок вимірювання до прилеглих територій, генеруючи безперервні теплові карти акустичного забруднення міста. Програмний комплекс також містить потужний модуль диспетчеризації та сповіщень: у разі, якщо еквівалентний рівень шуму у житловій зоні в нічний час перевищує встановлений санітарними нормами ліміт, система автоматично генерує інцидент у базі даних і розсилає сповіщення через Email або Telegram-боти відповідальним інспекторам, мінімізуючи час реакції на екологічні порушення.

Таким чином, розробка програмно-апаратного забезпечення кіберфізичної системи контролю рівня шуму вимагає комплексного, системного підходу. Апаратна складова, деталізована на наведених схемах і рисунках, забезпечує прецизійне вимірювання, захист від зовнішніх факторів та первинну локальну обробку сигналу в умовах жорстких енергетичних обмежень. Водночас

програмна складова, яка охоплює як мікроконтролерний код, так і серверну аналітику на базі потужних систем управління базами даних та алгоритмів штучного інтелекту, визначає загальну функціональність, гнучкість та інтелектуальність системи, створюючи надійний фундамент для автоматизації екологічного моніторингу в масштабах сучасного смарт-міста.

1.3 Архітектура та задачі розробки системи дистанційного моніторингу акустичного стану міста

На основі детального аналізу предметної області, розгляду існуючих методів екологічного моніторингу та оцінки сучасних апаратно-програмних рішень для побудови систем Інтернету речей, виникає обґрунтована необхідність у розробці спеціалізованого комплексу для безперервного вимірювання акустичного забруднення. Існуючі комерційні аналоги здебільшого характеризуються закритістю архітектури, високою вартістю масштабування та надмірною апаратною надлишковістю для задач базового муніципального моніторингу. З іншого боку, використання виключно портативних шумомірів не здатне забезпечити просторово-часову безперервність збору даних, яка є критично важливою для побудови об'єктивної картини акустичного стану великого міста. Таким чином, фундаментальною проблемою, що потребує вирішення в межах даної кваліфікаційної роботи, є подолання розриву між потребою у масовому, дешевому та надійному зборі екологічних метрик і відсутністю відкритої, гнучкої, науково обґрунтованої платформи для реалізації цих функцій.

Головною метою розробки є проєктування, програмно-апаратна реалізація та тестування кіберфізичної системи дистанційного контролю рівня шуму в міському середовищі. Ця система повинна базуватися на сучасних мікроконтролерних архітектурах, використовувати парадигму Інтернету речей для передачі даних та хмарні технології для їхнього накопичення, аналітичної

обробки й просторової візуалізації, забезпечуючи при цьому високу автономність і точність, достатню для прийняття управлінських рішень екологічними службами.

Об'єктом дослідження у роботі виступає процес автоматизованого безперервного моніторингу акустичного стану навколишнього середовища в умовах інтенсивного антропогенного та транспортного навантаження урбанізованих територій. Предметом дослідження є комплекс апаратних та програмних засобів, алгоритми цифрової обробки звукових сигналів на периферійних вузлах, а також методи організації надійної телеметричної взаємодії в розподілених кіберфізичних системах екологічного призначення.

Для досягнення поставленої глобальної мети необхідно виконати глибоку декомпозицію загальної проблеми та формалізувати низку взаємопов'язаних науково-практичних і проєктних задач, які охоплюють усі рівні архітектури кіберфізичної системи - від фізичного збору аналогового сигналу до представлення обробленої інформації кінцевому користувачу.

Першочерговим завданням є проєктування фізичного (апаратного) рівня системи, що полягає у створенні архітектури вимірювального вузла. Необхідно здійснити обґрунтований вибір мікроконтролерної платформи, яка виступатиме ядром обчислень на периферії. Обраний мікроконтролер повинен мати достатню продуктивність для виконання математичних операцій із рухомою комою у реальному часі, а також бути оснащеним необхідними інтерфейсами для підключення датчиків та комунікаційних модулів. Окремою важливою задачею на цьому етапі є вибір акустичного сенсора. Необхідно проаналізувати характеристики електретних та MEMS-мікрофонів, визначити оптимальний діапазон сприйманих частот та чутливість, що дозволить адекватно фіксувати рівень звукового тиску в діапазоні від фонового нічного шуму житлових районів (близько 30-40 дБ) до пікових навантажень поблизу автомагістралей або промислових об'єктів (до 100-110 дБ). Також необхідно спроєктувати підсистему

живлення вузла, яка б забезпечувала його стабільну роботу в умовах можливих перепадів напруги або при використанні автономних джерел енергії.

Також у межах проєктування інтерфейсу користувача та бізнес-логіки системи стоїть задача розробки гнучкої підсистеми сповіщень. Необхідно реалізувати програмний механізм, який дозволятиме адміністраторам системи налаштовувати багаторівневі порогові значення для різних зон міста відповідно до санітарних норм, окремі нормативи для денного і нічного часу, для житлових районів і промислових зон. У разі фіксації стійкого перевищення заданих порогів, система повинна автоматично генерувати тривожні повідомлення та розсилати їх відповідальним особам або інтегруватися з існуючими муніципальними системами реагування на надзвичайні ситуації.

Окремим, наскрізним блоком у постановці задачі є формулювання нефункціональних вимог до розроблюваної кіберфізичної системи, які забезпечать її життєздатність в реальних умовах експлуатації. До таких вимог належить масштабованість - архітектура програмного та апаратного забезпечення має бути спроєктована таким чином, щоб додавання нових вимірювальних вузлів не вимагало перепроєктування ядра системи чи зупинки серверних служб. Наступною вимогою є інформаційна безпека: оскільки екологічні дані можуть мати юридичну вагу при вирішенні муніципальних спорів, процес передачі телеметрії від мікроконтролера до сервера повинен бути захищений методами криптографічного шифрування для запобігання підміні показників. Крім того, система має характеризуватися високою відмовостійкістю на апаратному рівні, що передбачає використання сторожових таймерів Watchdog Timer у мікроконтролерах для їхнього автоматичного перезавантаження у випадку програмних збоїв чи зависань.

Підсумовуючи постановку задачі, можна стверджувати, що розробка кіберфізичної системи дистанційного контролю рівня шуму є комплексним інженерним проєктом. Він вимагає синтезу знань із галузей схемотехніки, теорії цифрової обробки сигналів, мережевих технологій та сучасних методів

проєктування розподілених програмних архітектур. Вирішення описаного спектра завдань дозволить отримати функціонально завершений програмно-апаратний продукт, здатний ефективно виконувати роль інструментального ядра для розгортання повномасштабної міської системи моніторингу акустичного забруднення, що, своєю чергою, стане важливим кроком у напрямку підвищення екологічної безпеки та якості життя мешканців сучасних мегаполісів.

1.4 Постановка задачі

Основними завданнями роботи є комплексне дослідження принципів функціонування сучасних систем Інтернету речей з особливим акцентом на їхнє застосування у сфері екологічного моніторингу. У цьому контексті передбачається детальне вивчення архітектури побудови таких систем, а також аналіз повного життєвого циклу обробки акустичних телеметричних даних, від моменту їх збору до фінальної передачі та аналізу. Окрему увагу приділено освітньому аспекту, що вимагає проведення ґрунтовного аналізу існуючих методичних підходів до підготовки майбутніх фахівців у галузі комп'ютерної інженерії. Зокрема, досліджується загальна ефективність та доцільність використання спеціалізованих апаратних засобів безпосередньо у навчальному процесі для формування практичних навичок проектування.

Наступним важливим етапом є глибока характеристика структури предметної області з подальшою розробкою базової апаратної моделі. Ця модель має чітко відображати процеси взаємодії центрального мікроконтролера з різноманітними периферійними пристроями, включаючи сучасні сенсори та виконавчі механізми. Для обґрунтування актуальності розробки необхідно описати існуючі на сучасному ринку рішення щодо навчальних лабораторних стендів та критично оцінити їхні недоліки. Головним чином увага фокусуватиметься на проблемах закритості архітектури багатьох комерційних платформ та критичній відсутності інтегрованих мережевих модулів, що

обмежує вивчення IoT-технологій. Як результат цього аналізу, планується запропонувати власні, відкриті та доступні шляхи вирішення виявлених проблем.

Базуючись на результатах проведених теоретичних досліджень, робота передбачає чітке визначення основних функцій майбутнього програмно-апаратного комплексу із формуванням вичерпного переліку функціональних та нефункціональних вимог до нього. Підводячи підсумки щодо гострої необхідності впровадження таких комплексних інструментів для вивчення мікропроцесорної техніки та передових мережевих протоколів, буде остаточно сформовано об’єкт і головну мету для наступних етапів проектування. Це підготує підґрунтя для безпосередньої практичної реалізації, яка включатиме трасування та паяння друкованих плат, а також написання відповідного програмного забезпечення на рівні мікроконтролера. На фінальному етапі роботи передбачено об'єктивну оцінку ступеня виконання всіх поставлених завдань, а також ретельну перевірку відповідності отриманих практичних результатів початково заявленим технічним вимогам проекту.

1.5 Висновок до 1-го розділу

У першому розділі кваліфікаційної роботи проведено ґрунтовний аналіз теоретичних основ, структурно-функціональних особливостей та програмно-апаратних засобів, необхідних для створення кіберфізичної системи дистанційного моніторингу акустичного стану міського середовища. На основі проведеного дослідження обґрунтовано критичну необхідність переходу від традиційних, епізодичних методів вимірювання шумового забруднення до сучасних автоматизованих рішень. Доведено, що класичні ручні заміри не здатні забезпечити просторово-часову безперервність контролю, тоді як впровадження розподіленої системи на базі концепції Інтернету речей (IoT) дозволяє створити

об'єктивну, високоточну та цілодобову картину акустичного ландшафту сучасного мегаполіса.

Визначено, що ефективна реалізація такого програмно-апаратного комплексу вимагає побудови чіткої трирівневої архітектури. Фізичний рівень має базуватися на використанні сучасних MEMS-мікрофонів та мікроконтролерів для здійснення граничних обчислень безпосередньо в точці вимірювання, зокрема цифрової фільтрації та частотного зважування за кривою «А». Транспортний рівень повинен забезпечувати надійну передачу телеметрії за допомогою енергоефективних бездротових технологій і протоколів. Своєю чергою, прикладний рівень має концентрувати дані у хмарних часорядних базах для їх подальшої глибокої аналітики, візуалізації у вигляді динамічних теплових карт шуму та автоматичного генерування тривожних сповіщень при перевищенні санітарних норм.

Аналіз існуючих на ринку комерційних систем екологічного контролю та навчальних лабораторних стендів виявив низку фундаментальних недоліків, серед яких закритість архітектури, висока вартість масштабування та відсутність інтегрованих модулів мережевої IoT-взаємодії. Це зумовлює актуальність розробки власного, відкритого програмно-апаратного комплексу, який виконуватиме подвійну функцію. З одного боку, система слугуватиме дієвим інструментом для муніципального екологічного моніторингу. З іншого боку, спроєктована базова апаратна модель стане доступним навчальним стендом для підготовки майбутніх фахівців із комп’ютерної інженерії, дозволяючи на практиці опановувати процеси трасування друкованих плат, пайку, програмування мікроконтролерів та налаштування мережевих протоколів.

Таким чином, сформована у першому розділі теоретична та аналітична база повністю розкриває проблематику предметної області, деталізує постановку задачі й створює надійне інженерне підґрунтя для переходу до етапу безпосередньої практичної реалізації розробки, що буде висвітлено в наступних розділах роботи.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА АЛГОРИТМІВ КІБЕРФІЗИЧНОЇ СИСТЕМИ

2.1 Визначення апаратних і програмних підсистем програмно-технічного засобу

Проєктування сучасної кіберфізичної системи дистанційного контролю рівня шуму вимагає комплексного підходу до формування її архітектури, яка повинна забезпечувати безперервний збір, надійну передачу, збереження та аналітичну обробку акустичних даних у масштабі реального часу. Відповідно до концепції Інтернету речей, розроблюваний програмно-технічний засіб структурно поділяється на три фундаментальні ієрархічні рівні: фізичний рівень, рівень сприйняття або периферійний вузол, мережевий рівень транспортна підсистема та прикладний рівень серверна інфраструктура та клієнтський інтерфейс. Кожен із цих рівнів являє собою складну композицію апаратних та програмних компонентів, тісна інтеграція яких визначає загальну ефективність, відмовостійкість та масштабованість системи.

Основою фізичного рівня розроблюваної системи є периферійний вимірювальний вузол, апаратна підсистема якого побудована на базі високопродуктивної мікроконтролерної платформи. Враховуючи специфіку екологічного моніторингу в міському середовищі, до мікроконтролера висуваються жорсткі вимоги щодо обчислювальної потужності, наявності багатоканальних аналого-цифрових перетворювачів АЦП високої роздільної здатності та інтегрованих модулів бездротового зв'язку. У межах даного проєкту для реалізації обчислювального ядра доцільно використовувати платформу сімейства ESP32. Даний вибір обґрунтовується наявністю вбудованого модуля Wi-Fi стандарту 802.11 b/g/n, що дозволяє організувати бездротову передачу даних без використання додаткових зовнішніх мережевих шилдів, тим самим зменшуючи габарити та енергоспоживання кінцевого пристрою.

Крім того, мікроконтролер ESP32 оснащений апаратними 12-бітними аналого-цифровими перетворювачами, що є критично важливим для забезпечення необхідної точності вимірювання аналогового сигналу з акустичних сенсорів. Роздільна здатність у 12 біт означає, що безперервний аналоговий сигнал напруги перетворюється у дискретні цифрові значення в діапазоні від 0 до 4095. Згідно з розробленою апаратною схемою, периферійний вузол обслуговує відразу три акустичні сенсори, які підключені до портів вводу-виводу, зокрема, піни 34, 35 та 32, що налаштовані в режим аналогового читання. Використання трьох незалежних сенсорів на одному вузлі дозволяє реалізувати просторове рознесення точок вимірювання, підвищити надійність системи за рахунок апаратного резервування або застосувати алгоритми усереднення показників для нівелювання локальних акустичних аномалій, поривів вітру безпосередньо біля одного з мікрофонів.

Програмна підсистема периферійного вузла реалізована мовою програмування C++ з використанням середовища розробки та бібліотек Arduino Core, що забезпечує високий уровень апаратної абстракції. Життєвий цикл вбудованого програмного забезпечення поділяється на два класичні етапи: ініціалізацію та безперервний цикл виконання. На етапі ініціалізації мікроконтролер конфігурує послідовний порт для виведення налагоджувальної інформації та ініціює з'єднання з локальною бездротовою мережею за заданими ідентифікаторами, SSID та пароль. Програмна логіка передбачає блокуючий цикл очікування, який призупиняє подальше виконання коду до моменту успішного отримання IP-адреси від маршрутизатора, що гарантує неможливість спроб відправки даних за відсутності мережевого з'єднання.

В основному циклі виконання реалізовано алгоритм безперервного опитування аналогових входів для отримання значень з кожного підключеного сенсора. Вбудоване програмне забезпечення здійснює зчитування сигналів та їхнє лінійне масштабування у цільовий діапазон акустичного тиску від 30 до 120 дБ. Після цього виконується серіалізація даних для забезпечення сумісності із

сучасними веб-технологіями та RESTful API. Транспортним форматом даних обрано JSON. Мікроконтролер динамічно формує текстовий рядок, який містить масив об'єктів, де кожен об'єкт репрезентує окреме вимірювання та включає унікальний ідентифікатор сенсор і відповідне йому значення рівня шуму.

Мережева підсистема відповідає за доставку сформованого JSON-пакету на віддалений сервер. У розробленій системі ця взаємодія реалізується поверх протоколу прикладного рівня HTTP. Мікроконтролер виступає в ролі HTTP-клієнта, який ініціює POST-запит до визначеної URL-адреси серверного API у прототипній версії для тунелювання локального сервера використовується сервіс ngrok. У заголовках запиту обов'язково вказується тип контенту application/json, що дозволяє серверній частині коректно десеріалізувати вхідні дані. Програмний алгоритм мікроконтролера також обробляє відповідь сервера, що є елементом механізму підтвердження доставки, після чого система переходить у стан очікування (затримка 3 секунди) для зниження навантаження на мережу та сервер перед наступною ітерацією вимірювання.

Важливим аспектом практичної реалізації вбудованого програмного забезпечення є всебічне забезпечення високої стійкості кіберфізичної системи до потенційних мережевих збоїв, завад та тимчасової втрати зв'язку із віддаленим сервером, що є вкрай поширеною проблемою в умовах динамічного міського середовища. З цією метою у мікроконтролерний код інтегровано спеціалізований алгоритм відмовостійкості, який автоматично активується у разі невдалої спроби відправки HTTP-запиту або повної відсутності сигналу бездротової мережі. Якщо встановлений мережевий тайм-аут перевищує критичне значення або хмарний сервер повертає незадовільний код відповіді, центральна логіка призупиняє спроби прямої трансляції і перенаправляє сформовані JSON-пакети у локальне сховище пристрою.

Для візуального представлення описаного алгоритму роботи програмного забезпечення мікроконтролера розробно блок-схему (рис 2.1), яка відображає послідовність виконання операцій.

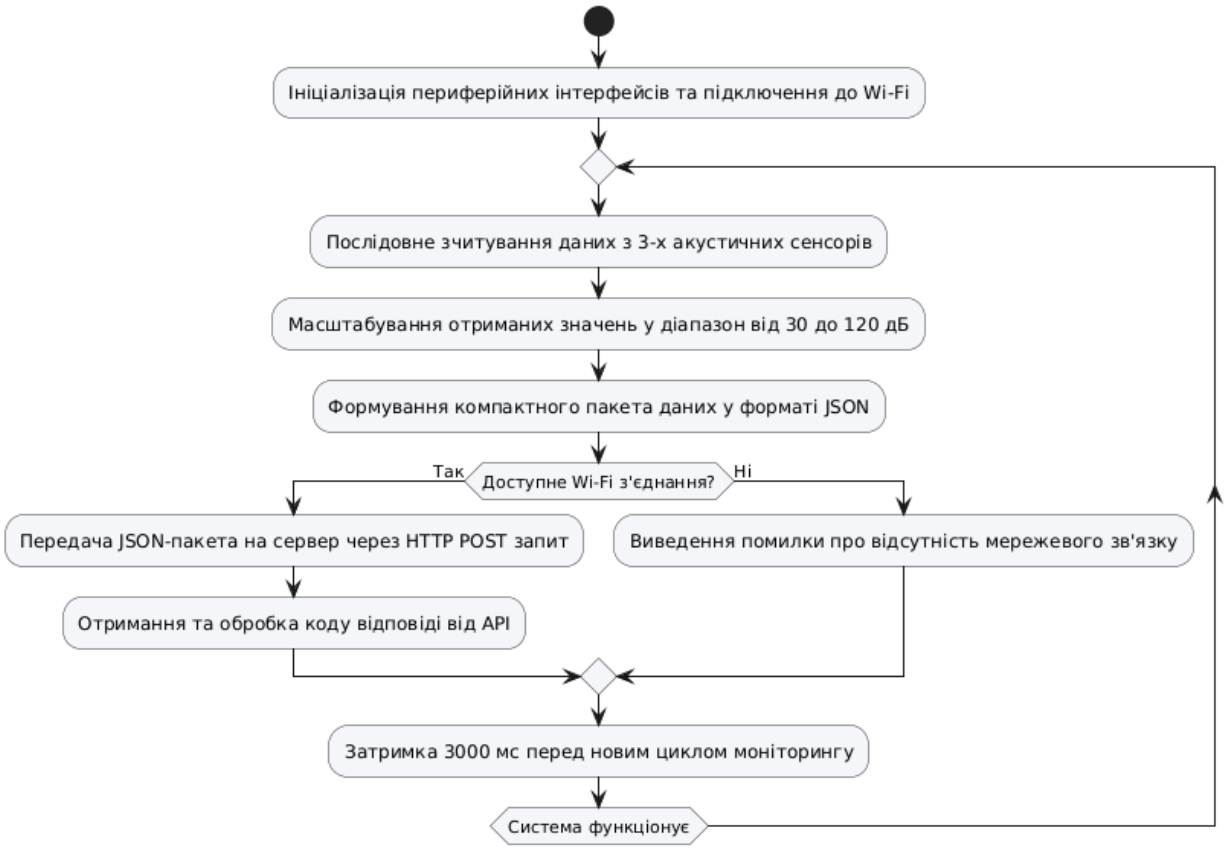


Рисунок 2.1 - Алгоритм роботи програмного забезпечення мікроконтролера

Прикладний рівень кіберфізичної системи представлений серверною програмною інфраструктурою, яка розроблена мовою програмування Python з використанням мікрофреймворку Flask. Вибір фреймворку Flask зумовлений його легковажністю, високою продуктивністю та наявністю вбудованого WSGI-сервера, що ідеально підходить для розробки RESTful API в системах Інтернету речей. Серверна підсистема виконує дві глобальні функції: маршрутизацію та обробку вхідних телеметричних потоків та генерацію веб-інтерфейсу для взаємодії з користувачем Frontend.

Для збереження екологічних даних застосовується реляційна система управління базами даних SQLite. Це вбудована СУБД, яка зберігає всю структуру та масиви даних у єдиному файлі на диску сервера, що значно спрощує розгортання та адміністрування прототипу системи без необхідності

конфігурації громіздких клієнт-серверних баз даних таких як PostgreSQL або MySQL. Архітектура бази даних розроблена з урахуванням необхідності швидкого запису часових рядів та складається з двох нормалізованих таблиць.

Перша таблиця, noise_logs, виконує роль журналу подій (транзакційна таблиця). Вона призначена для безперервного запису результатів вимірювань і містить текстове поле ідентифікатора сенсора, цілочисельне поле значення рівня шуму та поле мітки часу. Друга таблиця, sensor_settings (довідкова таблиця), зберігає конфігураційні параметри системи, зокрема допустимі порогові значення рівня шуму для кожного окремого сенсора. Такий реляційний підхід дозволяє гнучко налаштовувати санітарні норми індивідуально для кожної локації, для сенсора в житловій зоні норма може бути встановлена на рівні 70 дБ, а для сенсора біля автомагістралі - 80 дБ, без необхідності внесення змін у вихідний код серверного застонку.

Під час надходження POST-запиту на кінцеву точку /api/log, серверне програмне забезпечення здійснює парсинг JSON-тіла запиту, автоматично генерує серверну мітку часу для уникнення проблем із розсинхронізацією годинників на периферійних пристроях та виконує ітеративний запис даних у таблицю noise_logs за допомогою SQL-команд INSERT. Відділення логіки збору даних від логіки їхнього представлення гарантує відмовостійкість системи: навіть при пікових навантаженнях або збоях у роботі веб-інтерфейсу, процесу фіксації екологічних показників не буде перерваний. Крім того, застосування оптимізованого режиму журналювання WAL дозволяє мінімізувати час блокування таблиць, забезпечуючи високу пропускну здатність при паралельному надходженні пакетів від багатьох датчиків.

Взаємозв'язки між сутностями бази даних та загальну організацію збереження інформації на серверному рівні можна проілюструвати за допомогою діаграми сутність-зв'язок (рис .2.2).

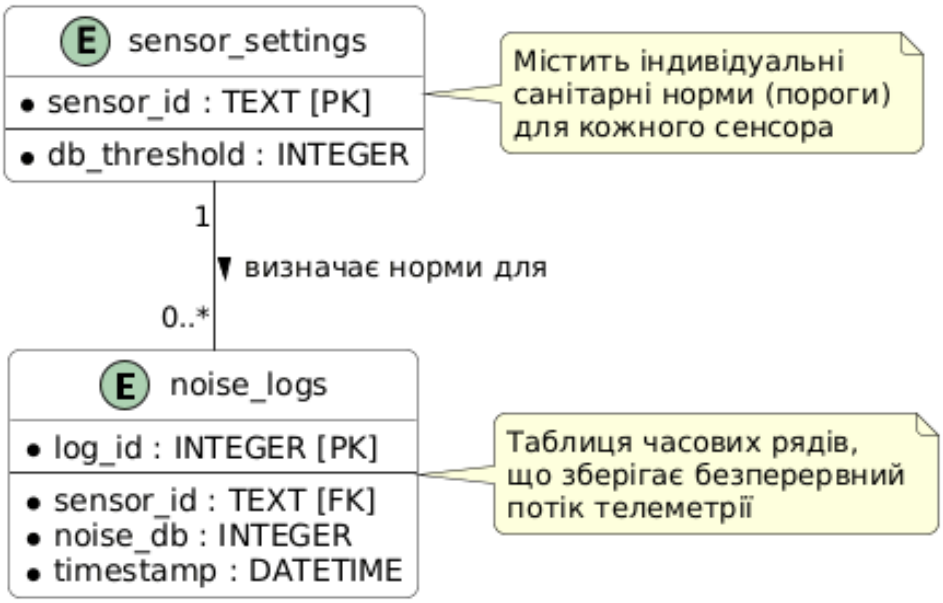


Рисунок 2.2 - Взаємозв'язки між сутностями бази даних

Клієнтська підсистема Frontend генерується серверним маршрутом коренева директорія веб-застосунку. Замість використання складних фронтенд-фреймворків, розробниками застосовано механізм шаблонізації Jinja2, інтегрований у Flask. При зверненні користувача до панелі управління, сервер виконує складний SQL-запит із застосуванням операцій JOIN та GROUP BY, який агрегує найсвіжіші показники з таблиці журналу та об'єднує їх з пороговими значеннями з таблиці налаштувань. Окремим запитом вилучається масив історичних даних за останні 30 хвилин для формування часового ряду.

Отримані агреговані структури даних передаються у HTML-шаблон. Візуалізація історичних коливань рівня шуму реалізована за допомогою потужної JavaScript-бібліотеки Chart.js, яка рендерить інтерактивний лінійний графік на елементі canvas. Для забезпечення оперативності моніторингу, в архітектуру веб-інтерфейсу закладено механізм автоматичного оновлення за допомогою JavaScript-таймерів, що ініціюють перезавантаження даних кожні дві секунди. Бізнес-логіка шаблонізатора також включає систему умовного рендерингу: програмний код порівнює поточний рівень шуму із затвердженим

порогом (row[1] > row[3]), і у випадку перевищення автоматично генерує візуальне сповіщення (тег із червоним фоном «ПЕРЕВИЩЕННЯ!»). Це реалізує найважливішу функцію диспетчеризації - миттєве привернення уваги екологічного інспектора до акустичних аномалій у міському середовищі.

Загалом, визначені апаратні та програмні підсистеми формують цілісну, синергетичну кіберфізичну архітектуру. Використання недорогих, але продуктивних мікроконтролерів у поєднанні з легковажними протоколами передачі даних та хмарними технологіями обробки дозволяє вирішити головну проблему сучасного екологічного моніторингу - створення щільної, просторово розподіленої мережі безперервного спостереження за рівнем шуму при мінімальних економічних та енергетичних витратах. Застосовані програмні патерни, такі як RESTful архітектура та розділення рівнів обробки даних, забезпечують системі високий ступінь масштабованості, що дозволить у майбутньому легко інтегрувати додаткові вимірювальні вузли або нові типи екологічних сенсорів (наприклад, датчики якості повітря) без необхідності кардинальної переробки ядра системи

2.2 Проєктування інформаційної моделі та структури бази даних

Для забезпечення надійного зберігання, швидкого пошуку та подальшого аналізу телеметричної інформації, що безперервно надходить від просторово розподілених периферійних вимірювальних вузлів, фундаментальним етапом проєктування кіберфізичної системи є розробка оптимізованої інформаційної моделі. Специфіка екологічного моніторингу полягає у генерації величезних масивів однотипних даних, які жорстко прив'язані до часової шкали, тобто у формуванні часових рядів. Враховуючи ці особливості, до структури бази даних висуваються суворі вимоги щодо забезпечення високої швидкодії операцій потокового запису, транзакцій типу INSERT та мінімізації надлишковості для ефективного використання дискового простору сервера.

У якості системи управління базами даних для реалізації серверної частини розроблюваного програмно-апаратного комплексу було обрано реляційну СУБД SQLite. Такий архітектурний вибір є глибоко обґрунтованим з огляду на специфіку проєкту. На відміну від важковагових клієнт-серверних рішень, таких як PostgreSQL або MySQL, СУБД SQLite є вбудованою системою, яка не вимагає розгортання окремого фонового процесу чи складного адміністрування прав доступу. Уся інформаційна структура, включаючи таблиці, індекси та самі дані, інкапсулюється у єдиному локальному файлі з назвою econoise_system.db. Це забезпечує виняткову портативність кіберфізичної системи, дозволяючи легко мігрувати серверну частину між різними обчислювальними платформами, а також значно спрощує процеси резервного копіювання критично важливої екологічної інформації. Крім того, SQLite має нативну інтеграцію зі стандартною бібліотекою мови програмування Python, що унеможливлює виникнення конфліктів залежностей при розгортанні веб-сервера Flask.

Процес проєктування логічної моделі даних базувався на принципах нормалізації баз даних для усунення дублювання інформації та забезпечення її логічної цілісності. В результаті декомпозиції предметної області було виділено дві ключові інформаційні сутності: сутність статичних конфігураційних параметрів вимірювальних вузлів та сутність динамічних транзакційних записів акустичного середовища. Відповідно до цих сутностей у базі даних було спроєктовано дві взаємопов'язані реляційні таблиці.

Перша таблиця, яка отримала ідентифікатор sensor_settings, виконує роль конфігураційного довідника системи. Вона призначена для зберігання метаданих про кожен активний сенсорний термінал, розгорнутий у міському середовищі. Структура цієї таблиці складається з двох атрибутів. Першим атрибутом є поле sensor_id, що має строковий тип даних TEXT. Цей атрибут визначений як первинний ключ, що гарантує унікальність кожного ідентифікатора у системі та автоматично створює кластерний індекс для миттєвого пошуку налаштувань конкретного пристрою. Другим атрибутом є поле db_threshold, яке використовує

цілочисельний тип даних INTEGER. У цьому полі зберігається максимально допустимий поріг рівня акустичного шуму (в децибелах) для локації, де фізично змонтовано відповідний датчик. Винесення нормативних показників в окрему таблицю дозволяє динамічно управляти санітарними обмеженнями без необхідності втручання у програмний код сервера чи перепрошивки мікроконтролерів. Програмне забезпечення здатне адаптивно порівнювати поточні показники з цими лімітами для визначення стану "Норма" або "Перевищення".

Друга таблиця з ідентифікатором noise_logs є ядром накопичення телеметричної інформації. Вона спроєктована для збереження безперервного потоку вимірювань і містить три атрибути. Поле sensor_id виступає в ролі зовнішнього ключа, який логічно пов'язує кожен запис про вимірювання з відповідним профілем налаштувань у таблиці sensor_settings, реалізуючи класичне реляційне відношення «один до багатьох», один датчик може генерувати безліч записів у часі. Атрибут noise_db призначений для збереження еквівалентного рівня звукового тиску, розрахованого мікроконтролером після аналого-цифрового перетворення та масштабування. Третім, критично важливим атрибутом є поле timestamp, у якому фіксується точний астрономічний час надходження пакету на сервер. Варто зазначити, що генерація часової мітки відбувається саме на серверній стороні під час виконання транзакції запису, що вирішує проблему синхронізації часу на дешевих мікроконтролерах, які не мають апаратних годинників реального часу з автономним живленням. Це архітектурне рішення забезпечує миттєву вибірку історичних даних за довільні проміжки часу та дозволяє мінімізувати час відгуку інтерфейсу навіть при накопиченні мільйонів записів, гарантуючи стабільне функціонування системи в умовах масштабного міського моніторингу.

Взаємозв'язки між сутностями та їх атрибутний склад наочно представлено за допомогою ER-діаграми (діаграми «сутність-зв'язок») на рисунку 2.3.

.

Рисунок 2.3 - Взаємозв'язки між сутностями

Для забезпечення високого рівня автоматизації та відмовостійкості програмного комплексу, процес ініціалізації бази даних був повністю інтегрований у стартовий життєвий цикл серверного застосунку. При кожному запуску сервера виконується спеціалізована функція init_db(). Алгоритм цієї функції перевіряє наявність файлу бази даних у файловій системі хоста. У разі його відсутності, система автоматично надсилає системні DDL-запити на створення таблиць noise_logs та sensor_settings з відповідними обмеженнями цілісності.

Більше того, для забезпечення готовності системи до проведення вимірювань одразу після її первинного розгортання, процедура ініціалізації містить блок попереднього заповнення даних. За допомогою SQL-інструкцій INSERT OR IGNORE у таблицю конфігурацій вносяться три базові профілі для тестових точок спостереження (SENSOR_01, SENSOR_02 та SENSOR_03) із попередньо визначеними лімітами шуму у 75, 70 та 80 децибел відповідно. Використання конструкції ігнорування конфліктів гарантує, що при наступних перезапусках сервера існуючі налаштування не будуть перезаписані або продубльовані, що зберігає цілісність накопиченої користувацької конфігурації. Такий підхід до проєктування структури бази даних формує надійний, масштабований та логічно завершений інформаційний фундамент для роботи всіх аналітичних модулів кіберфізичної системи.

2.3 Обґрунтування та розробка алгоритмів функціонування апаратної та програмної складових кіберфізичної системи

Функціонування сучасної розподіленої кіберфізичної системи екологічного моніторингу безпосередньо залежить від детермінованості, оптимізації та взаємоузгодженості алгоритмів, що виконуються як на периферійних апаратних вузлах, так і на центральному хмарному сервері. Алгоритмічне забезпечення визначає здатність системи своєчасно реагувати на акустичні коливання навколишнього середовища, мінімізувати накладні витрати обчислювальних ресурсів, гарантувати цілісність переданої інформації та забезпечувати репрезентативне відображення аналітики кінцевому користувачу. В межах цього проєкту було розроблено наскрізний комплекс алгоритмів, який охоплює життєвий цикл даних від моменту аналого-цифрового перетворення на фізичному рівні до фіксації інцидентів перевищення лімітів у часорядній базі даних та динамічної регенерації веб-інтерфейсу.

Першочерговим етапом у загальному контурі обробки інформації є забезпечення логіки роботи вбудованого програмного забезпечення периферійного вимірювального термінала, побудованого на базі мікроконтролера ESP32. Цей алгоритм орієнтований на циклічне виконання операцій збору даних, їх лінійну апроксимацію, серіалізацію у структурований формат та асинхронне транспортування мережевими каналами зв'язку. Логіка роботи вбудованого коду розділена на два фундаментальні етапи: одноразову ініціалізацію апаратних інтерфейсів, процедура налаштування та нескінченний цикл опитування периферії, процедура головного циклу.

На етапі ініціалізації мікроконтролер активує послідовний порт зв'язку на швидкості 115200 бод для забезпечення можливості локального виведення налагоджувальної інформації та діагностики працездатності систем. Після цього запускається підсистема бездротового зв'язку Wi-Fi. Алгоритм підключення передбачає переведення інтегрованого радіомодуля в режим клієнтської станції

та ініціювання асоціації з точкою доступу за допомогою заданих ідентифікаторів мережі та ключів шифрування. Процес підключення до бездротової інфраструктури реалізовано у вигляді блокуючого циклу очікування, який аналізує працездатність та поточний статус з'єднання. Мікроконтролер призупиняє виконання основних інструкцій на фіксовані інтервали часу 500 мілісекунд, циклічно опитуючи працездатність інтерфейсу, доки не буде зафіксовано стан успішного отримання мережевої IP-адреси. Такий підхід гарантує, що пристрій не перейде до виконання прикладних завдань, не маючи стабільного транспортного ковзного каналу.

Після успішного входження в мережу управління передається головному циклу, який функціонує в режимі безперервного послідовного опитування заданого масиву аналогових каналів. У даній конфігурації алгоритм оперує трьома виділеними виводами загального призначення порту GPIO 34, 35 та 32, до яких підключені первинні перетворювачі акустичного тиску. Вбудований 12-бітний аналого-цифровий перетворювач мікроконтролера здійснює квантування вхідного напругового сигналу, формуючи цілочисельні значення в діапазоні від 0 до 4095 одиниць.

Оскільки "сирі" дані АЦП не відображають реальну фізичну сутність досліджуваного процесу, алгоритм виконує математичне масштабування та лінійну інтерполяцію отриманого коду за допомогою функції відображення діапазонів. Математична логіка цього процесу базується на пропорційному перенесенні значень з координатної сітки дискретизації АЦП [0; 4095] на цільову логарифмічну шкалу інтенсивності звуку в децибелах, яка для даної системи визначена в межах від 30 дБ, поріг тиші в нічний час до 120 дБ, граничне шумове навантаження авіаційного чи промислового характеру. Результатом обчислення є масив оброблиних цілочисельних метрик, готових до відправки.

Наступним кроком алгоритму є формування мережевого пакету інформації. Задля уникнення залучення важковагових зовнішніх бібліотек обробки об'єктів та оптимізації використання оперативної пам'яті

мікроконтролера, розробники застосували алгоритм посимвольної збірки рядка в оперативній пам'яті. Програма послідовно конкатенує статичні елементи розмітки синтаксису JSON із динамічними текстовими представленнями ідентифікаторів сенсорів та обчислених значень децибел. Циклічний ітератор обходить масив результатів вимірювань, автоматично розставляючи розділові коми між об'єктами даних та завершуючи пакет кінцевими квадратними та фігурними дужками. Формується компактний та валідний об'єкт телеметрії.

У випадку тривалої відсутності Wi-Fi з'єднання, мікроконтролер не лише фіксує помилку, але й намагається виконати процедуру повторного підключення до заданої точки доступу через визначені інтервали часу. Це забезпечує високу автономність та відмовостійкість вимірювального комплексу. Завдяки такому підходу, периферійний вузол здатний самостійно відновлювати нормальний режим роботи після тимчасових апаратних або програмних збоїв мережевої інфраструктури, гарантуючи безперервність процесу збору важливих екологічних показників.

Фінальна фаза ітерації головного циклу відповідає за безпосередню емісію даних. Перед ініціюванням транзакції алгоритм виконує неблокуючу перевірку поточного стану Wi-Fi з'єднання. Якщо зв'язок є активним, створюється екземпляр клієнтського HTTP-протоколу, який надсилає заголовок із зазначенням типу контенту application/json та здійснює передачу сформованого рядка методом POST на статичну URL-адресу віддаленого веб-сервера. Програма аналізує код відповіді віддаленого сервера, що дозволяє контролювати успішність доставки на логічному рівні. Якщо перевірка статусу мережі виявляє обрив зв'язку, алгоритм оминає мережевий блок, виводит повідомлення про критичну помилку в послідовний порт і переходить до кроку затримки. Кожна повна ітерація завершується блокуванням процесорного ядра на 3000 мілісекунд, що визначає дискретність моніторингу довкілля раз на 3 секунди. Алгоритм функціонування периферійного вузла представлено на рисунку 2.4.

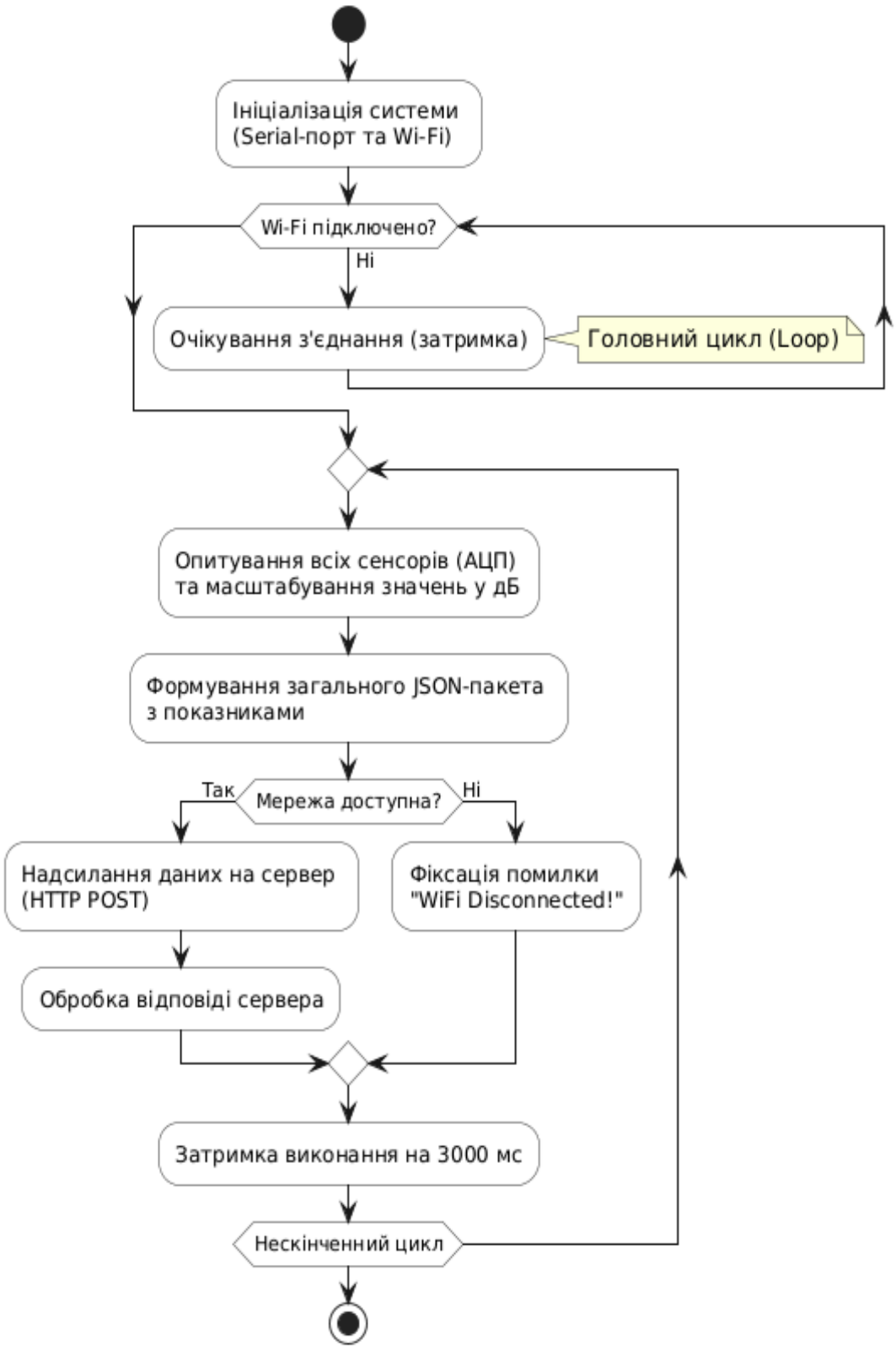


Рисунок 2.4 - Алгоритм функціонування периферійного вузла

Наступним логічним компонентом загальної архітектури є серверне програмне забезпечення, розроблене на базі мікрофреймворку Flask. Воно виступає в ролі центрального комутаційного та аналітичного ядра кіберфізичної системи. Алгоритм роботи сервера є подійно-орієнтованим і базується на паралельній обробці вхідних запусків від периферійних пристроїв та запитів від користувацьких браузерів. Веб-сервер працює в багатопотоковому режимі, ізолюючи кожну мережеву сесію в окремому контексті виконання, що забезпечує масштабованість системи під час одночасного обслуговування великої кількості вимірювальних точок.

Центральним вузлом обробки інформації є алгоритм шлюзу прийому телеметрії, закріплений за URL-маршрутом /api/log, який приймає виключно мережеві запити типу POST. Для забезпечення базового рівня безпеки та автентифікації клієнтів алгоритм першочергово перевіряє наявність спеціального ідентифікаційного токена у заголовках HTTP-запиту. Якщо токен відсутній або не відповідає еталонному ключу доступу, сервер миттєво перериває з'єднання, повертаючи статус помилки доступу HTTP 401 Unauthorized, що надійно захищає систему від запису несанкціонованої або зловмисної телеметрії сторонніми пристроями.

При надходженні валідного мережевого пакету алгоритм автоматично активує вбудований парсер веб-середовища Flask, який зчитує сирий текстовий потік з тіла запиту і десеріалізує його в структурований словниковий об'єкт Python. На цьому етапі відбувається сувора програмна валідація схеми даних. Система перевіряє наявність обов'язкового масиву показників під ключем data, а також здійснює типовимірювальний контроль: ідентифікатор пристрою перевіряється на належність до цілочисельного типу, а рівень шуму - на відповідність числовому формату з рухомою комою та знаходження у межах фізично можливого акустичного діапазону. У разі виявлення невідповідностей шлюз відхиляє пакет із кодом HTTP 400, відсікаючи аномальні або пошкоджені дані ще до моменту звернення до дискової пам'яті.

Якщо структура переданого об'єкта є валідною, сервер запускає процедуру тимчасового маркування: за допомогою системного таймера визначається поточна дата і час з точністю до секунди, після чого генерується уніфікований текстовий рядок мітки часу в міжнародному стандарті ISO, формат РРРР-ММ-ДД ГГ:ХХ:СС. Таке рішення дозволяє нівелювати проблему відсутності апаратних годинників реального часу на периферійних мікроконтролерах, переносячи задачу часової синхронізації безпосередньо на сторону сервера. Це також гарантує створення єдиної достовірної шкали часу для всіх вимірювань, усуваючи потенційні конфлікти часових поясів при подальшому аналітичному опрацюванні масивів даних.

Наступним етапом є транзакційний запис інформації у сховище. Алгоритм ініціює створення об'єкта підключення до реляційного файлу бази даних SQLite та відкриває контекстний покажчик. Увесь процес взаємодії з базою даних огорнуто у блок обробки виключень для забезпечення максимальної стабільності. Безпосереднє внесення записів реалізовано через ітераційний обхід масиву, витягнутого з ключа data вхідного JSON-пакету. Для кожного елемента масиву, який містить поля логічного номера пристрою та числового еквівалента гучності, формується параметризований SQL-запит виду INSERT INTO noise_logs.

Після завершення обходу всіх елементів ітератора та виконання відповідних записів у пам'яті, алгоритм викликає операцію фіксації транзакції. Це гарантує атомарність та цілісність операцій: у базі даних будуть збережені або всі показники з надісланого пакету, або жодного, якщо під час виконання виникне апаратний чи логічний збій. У випадку виникнення конфліктів блокування файлу бази даних, що є характерним для архітектури SQLite при інтенсивних паралельних запитах, генерується системне виключення. Алгоритм миттєво перехоплює його, автоматично скасовує транзакцію за допомогою команди відкоту та повертає пристрою статус внутрішньої помилки сервера HTTP 500. Отримавши таку відповідь, IoT-модуль переходить у режим

очікування і здійснює повторну спробу відправки буферизованого пакету пізніше, що повністю виключає втрату телеметричної інформації на каналі зв'язку.

Після успішного запису дескриптор підключення до БД коректно закривається, звільняючи системні ресурси. Усі критичні події - від відкриття транзакції до її фіксації або відкоту - синхронізуються із внутрішнім модулем системного логування для забезпечення прозорого моніторингу та діагностики роботи веб-сервера. У фіналі успішного циклу обробки шлюз повертає периферійному пристрою текстову відповідь у форматі JSON із HTTP-статусом 200, підтверджуючи успішне збереження певної кількості логів. Логіку обробки телеметрії представлено на рисунку 2.5.

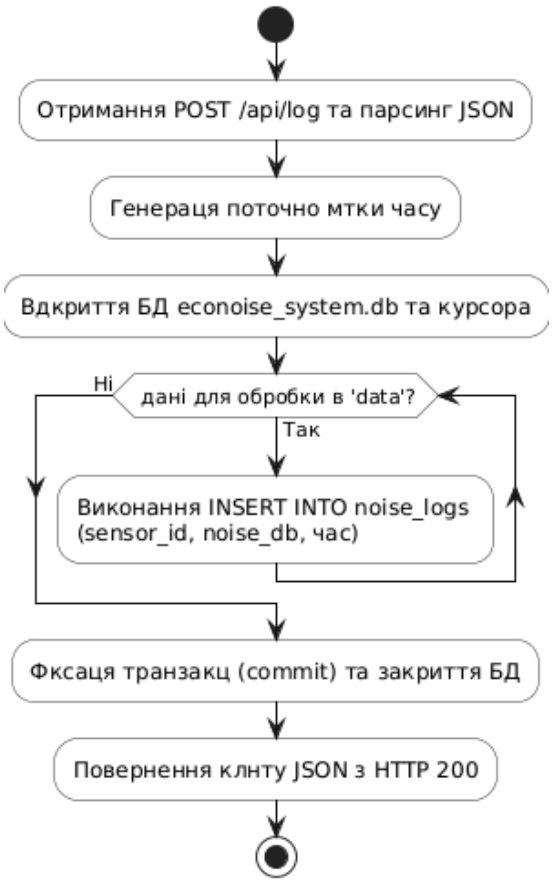


Рисунок 2.5 - Блок-схема алгоритму обробки телеметричних даних Flask-сервером

Окрім прийому первинних даних, серверне програмне забезпечення забезпечує виконання алгоритмів представлення інформації через головний графічний інтерфейс системи. Цей алгоритм відповідає за агрегацію історичних масивів даних, порівняння поточних значень із санітарними обмеженнями та генерацію динамічного HTML/JavaScript коду для відображення карт та графіків часових рядів.

При зверненні веб-браузера користувача до кореневої адреси сервера, алгоритм ініціює паралельний збір двох аналітичних зрізів інформації з бази даних. Перший зріз орієнтований на заповнення інформаційних карток експрес-моніторингу, які відображають миттєвий стан навколишнього середовища в точках спостереження. Запит групує записи за унікальним ідентифікатором датчика GROUP BY nl.sensor_id та відсікає всі застарілі дані, витягуючи лише максимальне значення часової мітки за допомогою агрегатної функції MAX. Результатом цієї операції є вибірка, що містить назву точки, її останній зареєстрований рівень шуму, точний час фіксації та встановлений для цієї зони поріг безпеки.

Другий аналітичний зріз призначений для забезпечення візуалізації часових коливань екологічного фону за допомогою лінійних графіків. З метою оптимізації швидкодії веб-інтерфейсу та запобігання перевантаженню оперативної пам'яті клієнтського браузера надмірною кількістю точок. Сервер обчислює граничну часову мітку шляхом віднімання від поточного системного часу інтервалу тривалістю у 30 хвилин (пів години). Створюється та виконується SQL-запит вибірки SELECT, який витягує з журналу логів записи, чий час створення є строго більшим або рівним обчисленому ліміту WHERE timestamp >= time_threshold. Дані сортуються у хронологічному порядку зростання.

Отриманий сирий масив історичних даних підлягає алгоритмічній структуризації на стороні сервера для його проведення до формату, сумісного з клієнтською графічною бібліотекою Chart.js. Алгоритм створює тривимірну структуру словника, яка містить спільний масив часових міток. вісь абсцис, та

три ізольовані масиви числових значень децибел для кожного з трьох датчиків окремо, вісь ординат. Програма ітерує витягнуті з БД рядки, відсікає за допомогою операцій розділення рядків календарну дату, залишаючи лише текстове представлення часу, та розподіляє показники за відповідними ключами словника.

Заключна фаза алгоритму візуалізації полягає в динамічній компіляції інтерфейсу. Сервер зчитує вбудований шаблон HTML-сторінки, що містить описи стилів оформлення CSS та скрипти JavaScript, та запускає механізм рендерингу рядків. На місце динамічних маркерів-підстановок вшивається сформований структурований масив останніх показників та серіалізований у JSON-формат словник півгодинної історії коливань. Згенерована сторінка відправляється клієнту, де веб-браузер виконує інтерпретацію коду. Послідовність взаємодії в межах цього процесу детально ілюструється на рисунку 2.6.

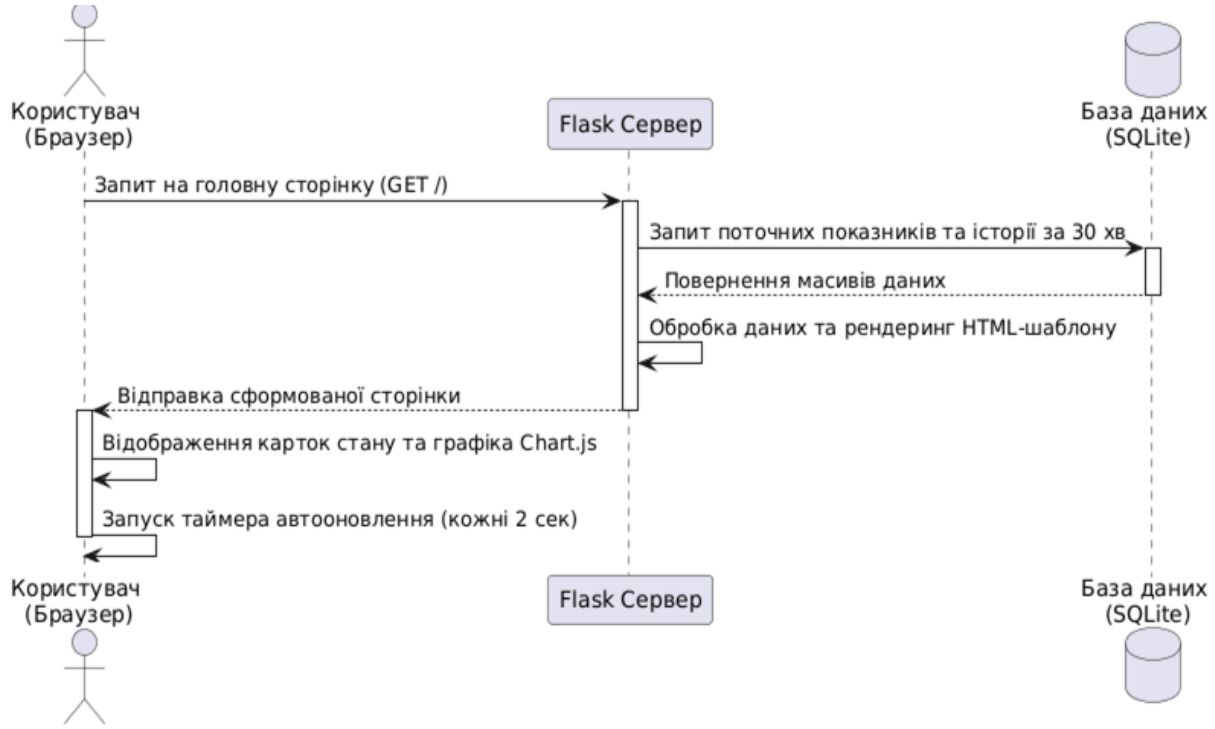


Рисунок 2.6 - Діаграма послідовності процесів агрегації та візуалізації екологічної інформації

Для кожної вимірювальної точки малюється персональна картка. Логічний алгоритм на стороні клієнта порівнює поточне числове значення шуму зі значенням нормативу. Якщо ліміт перевищено, блок картки маркується червоним індикатором тривоги з написом "ПЕРЕВИЩЕННЯ!", в іншому випадку - зеленим індикатором "НОРМА". Скрипт ініціалізує лінійний графік Chart.js, передаючи йому три масиви даних, які відображаються у вигляді кольорових ліній на спільній часовій шкалі. Для забезпечення безперервності спостереження без необхідності ручного втручання користувача, у заголовок веб-документа інтегрується циклічний таймер автоматичного оновлення, який кожні 2000 мілісекунд ініціює повне перезавантаження сторінки (location.reload()), запускаючи описаний серверний аналітичний цикл заново.

Таким чином, розроблений комплекс алгоритмів забезпечує детерміновану, надійну та замкнену структуру обробки інформації в межах кіберфізичної системи. Завдяки переносу математичних розрахунків масштабування на периферійні мікроконтролери та оптимізації аналітичних SQL-запитів ковзного часового вікна на сервері, досягається висока швидкість реакції інтерфейсу користувача та стабільність функціонування всієї системи моніторингу в реальному часі

2.4 Висновок до 2-го розділу

У другому розділі кваліфікаційної роботи виконано комплексне проєктування архітектури, інформаційної моделі та алгоритмів функціонування кіберфізичної системи дистанційного контролю рівня шуму. На основі визначених теоретичних концепцій розроблено надійну трирівневу структуру комплексу, що органічно об'єднує фізичний рівень збору даних, транспортну мережеву підсистему та прикладний рівень серверної аналітики й візуалізації.

Обґрунтовано вибір високопродуктивної мікроконтролерної платформи ESP32 як обчислювального ядра периферійного вузла. Розроблені алгоритми

вбудованого програмного забезпечення забезпечують безперервне опитування трьох незалежних акустичних сенсорів, математичне масштабування сирих даних аналого-цифрового перетворювача у логарифмічну шкалу 30–120 дБ та їхню серіалізацію у легковажний формат JSON. Особливу увагу приділено відмовостійкості фізичного рівня: інтегровано механізми перевірки цілісності мережевого з'єднання та обробки HTTP-помилок, що мінімізує ризики втрати екологічної телеметрії в умовах нестабільного бездротового зв'язку.

Спроєктовано оптимізовану реляційну інформаційну модель на базі вбудованої СУБД SQLite. Декомпозиція бази даних на таблицю динамічних транзакційних вимірювань та таблицю статичних конфігураційних лімітів дозволила забезпечити високу швидкість потокового запису часових рядів і гнучкість налаштування санітарних норм для кожної локації. Використання серверної генерації часових міток ефективно вирішило проблему апаратної розсинхронізації периферійних пристроїв.

Розроблено та описано замкнений цикл серверних і клієнтських алгоритмів на базі мікрофреймворку Flask. Серверна логіка гарантує безпечний прийом пакетів через токен-автентифікацію, їхню сувору валідацію та атомарний запис у базу даних із захистом від SQL-ін'єкцій. Завдяки агрегації SQL-запитів та використанню бібліотеки Chart.js, система здатна в режимі реального часу відображати історичні графіки коливань шуму, автоматично оновлювати панелі моніторингу та генерувати миттєві візуальні сповіщення у разі перевищення допустимого акустичного порогу.

Таким чином, спроєктовані програмно-апаратні рішення, структури баз даних та алгоритми утворюють масштабовану, логічно завершену архітектуру. Розроблена система повністю задовольняє функціональні вимоги безперервного міського екологічного моніторингу та формує надійний інженерний фундамент для переходу до фінального етапу — фізичної збірки, тестування та практичної експлуатації комплексу.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ КІБЕРФІЗИЧНОЇ СИСТЕМИ ДИСТАНЦІЙНОГО КОНТРОЛЮ РІВНЯ ШУМУ

3.1 Опис реалізації апаратного забезпечення та вбудованого програмного забезпечення периферійного вузла

Практичний етап розробки кіберфізичної системи дистанційного контролю рівня шуму передбачає безпосередній перехід від теоретичних структурних моделей та абстрактних інформаційних зв’язків до фізичного синтезу апаратних модулів і написання низькорівневого програмного забезпечення. Надійність, точність та автономність функціонування всієї розподіленої екологічної мережі критично залежать від того, наскільки оптимально скоординована взаємодія між фізичними сенсорами, обчислювальним ядром периферійного термінала та бездротовими інтерфейсами передачі даних. Процес практичної реалізації периферійного рівня системи моніторингу охоплює розгортання апаратної платформи, проектування електричних з’єднань, конфігурування підсистеми живлення, а також створення стійкого до збоїв вбудованого коду, який виконує первинну обробку сигналів безпосередньо у точці вимірювання за парадигмою граничних обчислень.

Основою апаратної складової вимірювального вузла став високоефективний 32-бітний двоядерний мікроконтролер архітектури Xtensa LX6, реалізований на базі популярного системного кристала ESP32. Вибір даного обчислювального компонента є технічно та економічно обґрунтованим для задач екологічного моніторингу смарт-міст. На відміну від застарілих 8-бітних мікроконтролерних архітектур, обчислювальне ядро обраної платформи функціонує на тактовій частоті до 240 МГц і має інтегрований апаратний блок обчислень із рухомою комою, що дозволяє виконувати складні операції цифрової обробки сигналів і спектрального аналізу в реальному часі. Головною архітектурною перевагою платформи є наявність вбудованого радіомодуля з підтримкою бездротових стандартів зв'язку Wi-Fi 802.11 b/g/n та Bluetooth. Це

дозволяє повністю відмовитися від використання зовнішніх дорогих комунікаційних плат розширення, знижуючи кінцеву вартість периферійного термінала, зменшуючи його геометричні розміри та суттєво підвищуючи загальну енергоефективність пристрою при тривалій автономній експлуатації.

Електрична архітектура та топологія з’єднань вимірювального вузла були спроектовані та верифіковані у спеціалізованому середовищі емуляції апаратних засобів Wokwi. Апаратна конфігурація передбачає паралельне підключення масиву з трьох первинних перетворювачів акустичного тиску до спеціалізованих аналогових входів мікроконтролера. У межах розробленого прототипу фізичні акустичні датчики інтегровані через виводи загального призначення, які мають апаратну підтримку внутрішнього аналого-цифрового перетворення, а саме GPIO 34, GPIO 35 та GPIO 32. Живлення сенсорного масиву реалізовано шляхом підключення їхніх ліній живлення до системної шини мікроконтролера з номінальною напругою 3.3 В, яка забезпечує стабільний потенціал без високих пульсацій. Спільний контур заземлення всіх компонентів підключено до виділеного виводу заземлення мікроконтролера, що дозволяє мінімізувати виникнення синфазних перешкод та електромагнітних завад на аналогових лініях зв’язку. Вони ефективно згладжують короткочасні просадки напруги під час пікових навантажень інтегрованого радіомодуля Wi-Fi в моменти активної передачі телеметрії. Крім того, використання вбудованого 12-бітного АЦП дозволяє досягти високої роздільної здатності вимірювань, забезпечуючи точне квантування аналогового сигналу для його подальшого математичного перетворення у мікроконтролері.

Для забезпечення стабілізації напруги та фільтрації високочастотних шумів у колі живлення передбачено встановлення керамічних декуплуючих конденсаторів, які згладжують короткочасні просадки напруги під час пікових навантажень радіомодуля в моменти передачі інформації. Схему електричних з'єднань апаратної частини периферійного вузла представлено на рисунку 3.1.

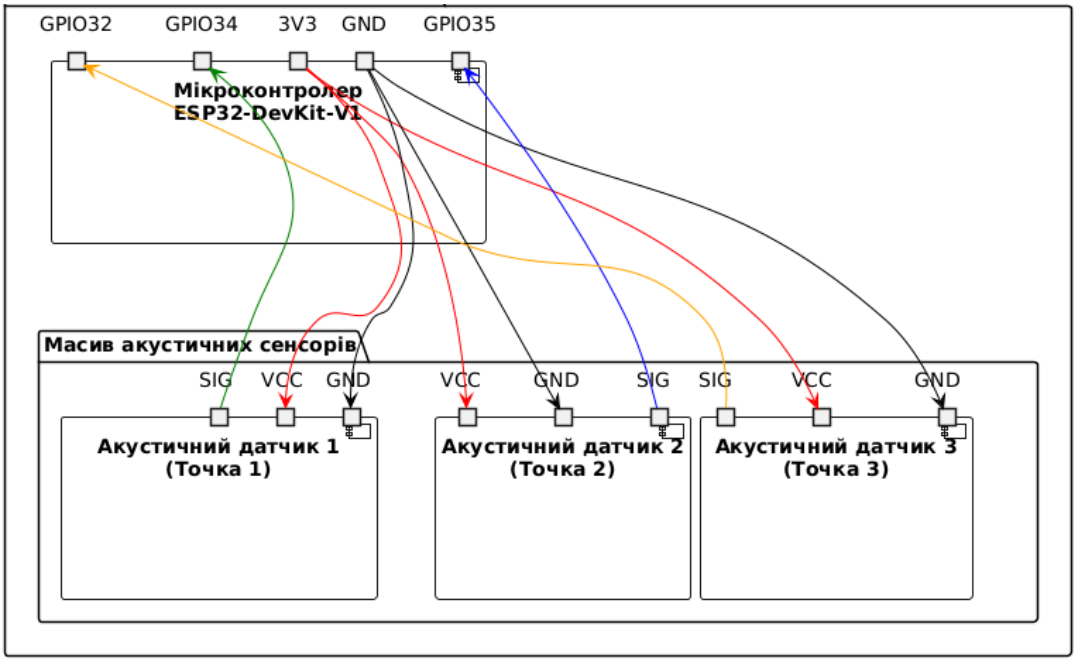


Рисунок 3.1 - Схема з’єднань периферійного вузла

Програмна реалізація вбудованого рівня виконана з використанням мови програмування C++ та розгорнута в середовищі розробки під керуванням компілятора GCC. Архітектура вбудованого програмного забезпечення побудована за модульним принципом і спирається на використання стандартних низькорівневих бібліотек управління периферією, що дозволило досягти максимальної швидкодії процесу зчитування даних та мінімального споживання динамічної пам’яті пристрою.

Перший етап життєвого циклу прошивки реалізований у межах базової процедури ініціалізації апаратних інтерфейсів. При подачі живлення на мікроконтролер запускається налаштування послідовного порту передачі даних UART на швидкості 115200 бід, що є необхідним для виведення телеметричної відладочної інформації на локальну консоль інженера під час проведення пусконалагоджувальних робіт. Одразу після цього активується бездротовий стек зв’язку за допомогою методів вбудованої бібліотеки Wi-Fi. Мікроконтролер переводить радіомодуль у режим клієнтської станції та ініціює процедуру автентифікації у локальній бездротовій мережі, використовуючи задані константи ідентифікатора точки доступу та пароля доступу. Процес підключення

до інфраструктури мережі реалізовано через блокуючий цикл, який безперервно перевіряє статус з’єднання з інтервалом у 500 мілісекунд, утримуючи процесорне ядро від виконання прикладних завдань до моменту успішного отримання динамічної IP-адреси. Локальне підтвердження успішного входу в мережу дублюється виведенням відповідного маркера у послідовний порт.

Після завершення фази початкового налаштування керування передається нескінченному циклу опитування датчиків. На кожній ітерації цього циклу програма послідовно звертається до вбудованого 12-бітного аналого-цифрового перетворювача АЦП послідовним викликом функції зчитування аналогової напруги на закріплених портах. Інтегрований АЦП виконує операцію квантування вхідного напругового сигналу, формуючи на виході дискретний цифровий код у діапазоні від 0 до 4095 одиниць, що відповідає амплітуді коливань повітряного середовища, зафіксованій датчиками.

Оскільки сирі значення АЦП позбавлені фізичного змісту для кінцевого аналітичного комплексу, програма виконує операцію математичного лінійного масштабування коду у стандартизовану логарифмічну шкалу гучності за допомогою спеціалізованого алгоритму відображення діапазонів. Функція трансформації пропорційно переносить дискретні координати з вхідного простору квантування АЦП на цільовий відрізок, межі якого визначені технічним завданням від 30 децибел, абсолютна тиша в приміщенні, до 120 децибел, граничний рівень звукового тиску. Цей підхід дозволяє проводити первинну калібровку сенсорів на рівні прошивки, компенсуючи нелінійність вхідних каскадів та формуючи очищені, готові до інтерпретації числові метрики рівня шуму для кожної вимірювальної точки простору.

Наступним найважливішим етапом роботи вбудованого програмного забезпечення є структурація та серіалізація зібраних даних для їх подальшого транспортування мережею. З метою економії обмежених обчислювальних ресурсів та виключення ризику виникнення критичних помилок фрагментації кучі heap fragmentation під час тривалої роботи пристрою, алгоритм формування

пакету реалізовано без залучення динамічного виділення пам'яті. Рядок повідомлення збирається в оперативній пам'яті шляхом послідовної конкатенації статичних символів розмітки JSON-синтаксису та динамічно перетворених у текст числових значень ідентифікаторів пристроїв і рівнів децибел. Керуючий цикл послідовно обходить масив результатів вимірювань, автоматично форматуючи вкладені об’єкти даних та розділяючи їх комами, формуючи в результаті компактний, легковаговий текстовий пакет телеметрії.

Завершальна фаза ітерації головного циклу відповідає за безпосередню емісію даних у хмару. Перед початком транзакції вбудоване програмне забезпечення здійснює неблокуючий аналіз поточного стану мережі Wi-Fi. У разі підтвердження активного статусу з’єднання, програма створює екземпляр об’єкта HTTP-клієнта та ініціює сесію зв’язку за статичною адресою віддаленого сервера. Для успішної інтерпретації пакета серверною стороною, у заголовок запиту примусово додається метатег Content-Type із значенням application/json, після чого сформований текстовий рядок відправляється мережевими сокетами за допомогою методу POST.

Програма зчитує повернений сервером код відповіді, реєструючи успішність логічного завершення транзакції у послідовний порт для налагодження. Якщо перевірка доступності бездротової інфраструктури виявляє втрату зв'язку, блок надсилання ігнорується, запобігаючи зависанню пристрою, а повідомлення про аварійний стан виводиться в консоль. Повна ітерація головного циклу завершується примусовим переведенням процесора в стан очікування на 3000 мілісекунд за допомогою системного таймера, що забезпечує стабільну частоту збору екологічних даних з періодичністю один раз на три секунди. Це дозволяє сформувати надійний часовий ряд без перевантаження комунікаційного каналу системи моніторингу. Такий підхід також сприяє оптимізації загального енергоспоживання периферійного вузла. Під час фіксованої затримки мікроконтролер мінімізує активність радіомодуля, що є

надзвичайно важливим фактором для забезпечення тривалої, безперебійної та автономної експлуатації комплексу.

3.2 Опис процесу створення, ініціалізації та програмної реалізації бази даних.

У структурі сучасних розподілених кіберфізичних систем підсистема збереження даних виконує критично важливу функцію забезпечення персистентності, цілісності та доступності інформації, що безперервно надходить із навколишнього середовища. Оскільки фізичні вимірювальні вузли периферійного рівня функціонують у безперервному режимі, генеруючи потоки телеметричних повідомлень з високою дискретністю, центральний сервер повинен володіти надійним і швидкодіючим інструментом для довготривалого збереження цих записів. Перехід від логічного проектування інформаційної моделі, виконаного на попередніх етапах, до безпосередньої фізичної та програмної реалізації бази даних вимагає детального аналізу системних механізмів СУБД, розробки алгоритмів автоматичного розгортання таблиць, конфігурування типів даних, а також імплементації транзакційної логіки взаємодії сервера з дисковим сховищем.

Програмна реалізація сховища даних у межах розробленого серверного компонента виконана з використанням реляційної системи управління базами даних SQLite, управління якою здійснюється безпосередньо з коду на мові програмування Python за допомогою вбудованого інтерфейсу прикладного програмування. Логіка взаємодії з файлом бази даних, що отримав назву econoise_system.db, повністю інтегрована в загальну подієво-орієнтовану архітектуру Flask-застосунку. Процес створення та підтримки життєвого циклу бази даних розділено на два ключові програмні контури: модуль початкової низькорівневої ініціалізації структури при першому запуску сервера та модуль

динамічної обробки транзакційних запитів при надходженні нових пакетів екологічної телеметрії.

Контур початкової ініціалізації реалізовано у вигляді автономної функції з назвою init_db(). Основне завдання цієї процедури полягає у верифікації наявності цільового файлу бази даних у локальній файловій системі хоста та автоматичному розгортанні реляційної структури у випадку її відсутності. Для виключення ризику затирання або пошкодження вже накопичених історичних даних екологічного моніторингу при перезапусках сервера, алгоритм виконує превентивну перевірку за допомогою засобів системного модуля взаємодії з операційною системою. Використовуючи метод аналізу шляхів файлової системи, програма визначає, чи існує файл econoise_system.db у робочій директорії проекту. Якщо файл виявлено, функція миттєво завершує своє виконання, передаючи керування наступним підсистемам веб-сервера, що гарантує збереження персистентного стану сховища.

У разі, якщо перевірка вказує на відсутність файлу бази даних, що характерно для етапу первинного розгортання програмного комплексу на новому обчислювальному обладнанні, алгоритм запускає процедуру створення сховища з нуля. Програма ініціює низькорівневий запит на відкриття з'єднання з базою даних, що автоматично призводить до генерації порожнього файлу з відповідним ім'ям на диску. Одразу після успішного створення дескриптора зв'язку, алгоритм відкриває об'єкт керуючого курсора, який виступає основним інструментом для трансляції SQL-інструкцій мови визначення даних DDL. Через цей курсор сервер послідовно виконує серію директив на створення реляційних таблиць системи.

Перша виконувана інструкція відповідає за створення таблиці журналювання акустичних показників noise_logs. Програмний код цієї таблиці визначає три базові колонки: sensor_id, noise_db та timestamp. Призначення текстового типу даних TEXT для ідентифікатора пристрою дозволяє гнучко масштабувати систему, використовуючи довільні алфавітно-цифрові назви для нових точок вимірювання. Поле числового значення рівня шуму отримує

цілочисельний тип INTEGER, що повністю відповідає дискретним даним, які надходять після лінійної апроксимації на мікроконтролері. Поле мітки часу визначається спеціалізованим типом DATETIME, який у межах архітектури SQLite забезпечує збереження хронологічних рядків у ISO-стандарті, формуючи оптимізовану основу для подальшої часової фільтрації та побудови графіків.

Друга інструкція курсора реалізує створення конфігураційної таблиці sensor_settings. У цій структурі поле sensor_id типу TEXT примусово оголошується первинним ключем PRIMARY KEY. Це архітектурне рішення на рівні рушія бази даних блокує можливість появи дублікатів ідентифікаторів пристроїв і автоматично будує унікальний b-tree індекс, що мінімізує час виконання операцій вибірки лімітів при аналізі тривожних станів. Поле допустимого порогу шуму db_threshold реєструється як цілочисельний атрибут INTEGER.

Після завершення проектування схем таблиць, алгоритм ініціалізації переходить до критично важливого етапу початкового наповнення довідників, що забезпечує працездатність системи моніторингу одразу після розгортання. Для запобігання виникненню логічних помилок цілісності, внесення базових конфігурацій реалізовано за допомогою специфічної SQL-конструкції INSERT OR IGNORE. Сервер послідовно надсилає три параметризовані запити, які реєструють три еталонні точки спостереження: пристрій SENSOR_01 із встановленим лімітом безпеки у 75 децибел, пристрій SENSOR_02 із нормативом у 70 децибел та пристрій SENSOR_03 із граничним значенням у 80 децибел. Використання модифікатора ігнорування конфліктів гарантує, що якщо база даних буде переініціалізована, існуючі користувацькі зміни порогів не будуть перезаписані початковими значеннями. Завершується функція init_db() обов'язковим викликом методу фіксації змін commit(), який переносить усі структури з тимчасового кешу оперативної пам'яті безпосередньо у фізичні сектори дискового накопичувача, після чого дескриптор підключення закривається.

Такий підхід до автоматичної ініціалізації бази даних повністю усуває необхідність ручного втручання адміністратора під час першого запуску програмного комплексу. Одразу після успішного закриття дескриптора підключення, веб-сервер Flask переходить у режим активного прослуховування вхідних мережевих з'єднань на визначеному порту. Це дозволяє системі миттєво розпочати безперебійний прийом потокової телеметрії від периферійних мікроконтролерів та одночасно обслуговувати клієнтські запити для генерації інтерактивного аналітичного дашборду.

Динамічний контур функціонування бази даних активується під час обробки вхідних повідомлень від периферійних пристроїв. Коли шлюз прийому телеметрії на маршруті /api/log фіксує вхідний HTTP POST запит, серверне програмне забезпечення запускає транзакційний алгоритм запису інформації в журнал. Процес починається з динамічного зчитування поточного системного часу сервера та його форматування у текстову константу. Після цього програма відкриває нове ізольоване з'єднання з файлом бази даних. Створення окремого підключення для кожної мережевої транзакції є важливою архітектурною особливістю вбудованих СУБД, оскільки це запобігає виникненню взаємних блокувань потоків і забезпечує коректну роботу механізму журналювання відкатів (rollback journal) SQLite.

Внесення інформації в таблицю noise_logs організовано за допомогою циклічного ітератора, який обходить десеріалізований масив даних. Замість формування динамічних текстових рядків запитів, що є грубою помилкою проектування і створює критичні вразливості для атак типу SQL-ін'єкцій, алгоритм використовує виключно параметризовані безпечні запити із використанням знаків запитання як маркерів підстановки параметрів. На кожній ітерації циклу виконується команда підготовки запису, куди передаються ідентифікатор датчика, обчислене значення децибел та сформований рядок часу. Рушій бази даних виконує компіляцію цього запиту один раз, після чого лише

підставляє безпечні змінні, що істотно підвищує загальну продуктивність підсистеми збереження даних.

Важливим проміжним етапом, що передує безпосередньому виконанню підготовлених запитів на рівні низькорівневого рушія бази даних, є сувора структурна, синтаксична та семантична валідація вхідного масиву телеметрії, яка реалізована на серверному програмному рівні. Оскільки розподілені периферійні датчики функціонують у відкритому міському середовищі, існує постійний ризик надходження спотворених інформаційних пакетів внаслідок раптових апаратних збоїв мікроконтролера, короткочасних завад у бездротових каналах зв'язку чи навіть цілеспрямованих спроб деструктивного стороннього втручання у роботу шлюзів RESTful API. Програмний алгоритм сервера в межах поточного циклу ітеративно аналізує кожен елемент десеріалізованого масиву на наявність обов'язкових ключів і повну відповідність типів даних. Зокрема, розраховане значення еквівалентного рівня акустичного тиску піддається логічній фільтрації на відповідність допустимому каліброваному діапазону від 30 до 120 дБ.

Після завершення обходу всіх елементів телеметричного масиву, алгоритм ініціює операцію атомарної фіксації транзакції. У цей момент СУБД SQLite записує дані у спеціальний тимчасовий журнал, перевіряє відсутність конфліктів доступу та виконує фізичне блокування файлу бази даних для перенесення логів у основне сховище. У випадку виникнення непередбачуваних помилок під час виконання SQL-запитів або порушення цілісності файлової системи, рушій бази даних автоматично ініціює процедуру відкату транзакції (rollback). Це гарантує, що частково збережені або пошкоджені записи не потраплять до загального журналу, забезпечуючи абсолютну узгодженість екологічної інформації.

На фінальному етапі, одразу після успішної фіксації або безпечного скасування транзакції, підключення закривається, ресурси пам'яті звільняються, а веб-сервер генерує відповідний статусний код HTTP. Віддаленому мікроконтролеру надсилається JSON-відповідь із підтвердженням збереження, що дозволяє периферійному пристрою коректно планувати наступні цикли

передачі телеметрії. Описаний наскрізний життєвий цикл ініціалізації та потокового запису інформації представлено у вигляді детальної схеми процесів на рисунку 3.2.

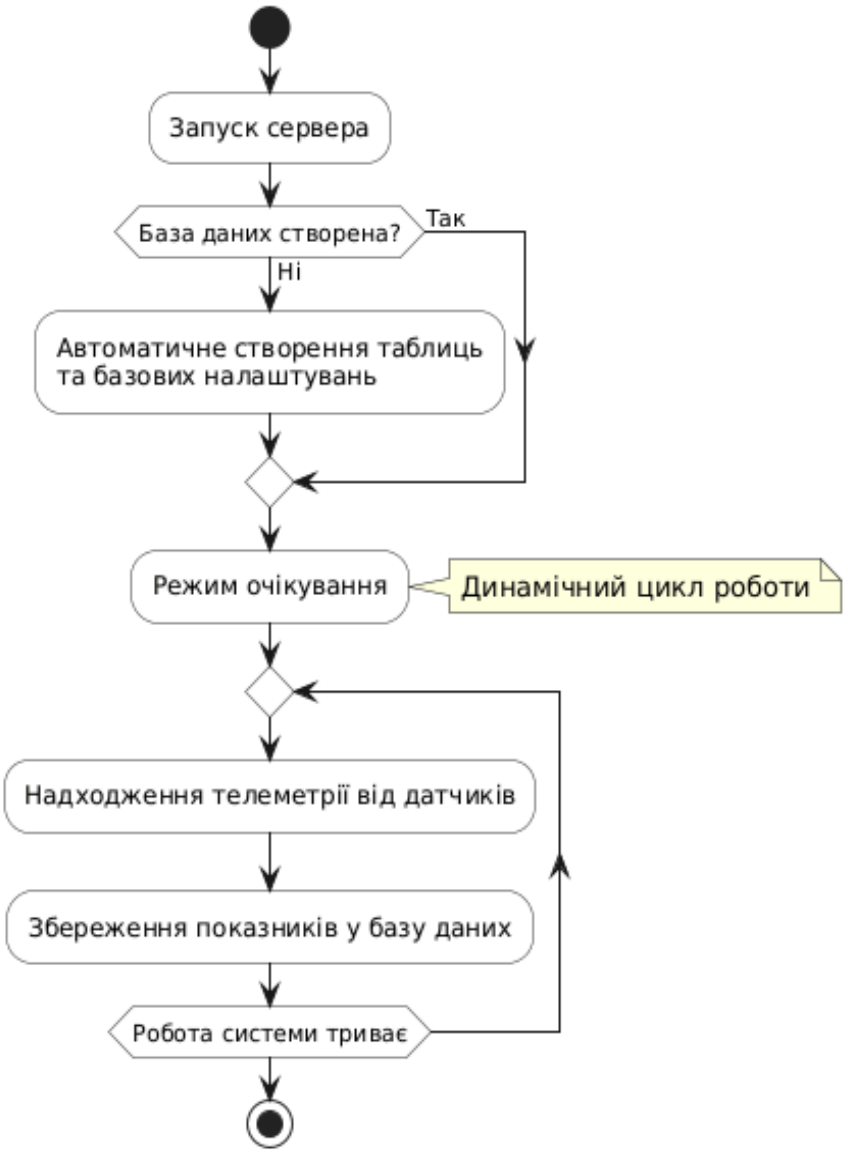


Рисунок 3.2 – Схема процесів кіберфізичної системи

Завдяки такій програмній реалізації, підсистема збереження даних характеризується високим рівнем автономності та захищеності. Розподіл логіки на ізольовані транзакційні підключення у поєднанні з використанням параметризованих SQL-інструкцій дозволяє системі стабільно функціонувати в

умовах постійного багатопотокового надходження інформації від вимірювальних пристроїв, мінімізуючи часові затримки на диску та гарантуючи абсолютну достовірність екологічних часових рядів, необхідних для прийняття управлінських рішень.

3.3 Опис програмної реалізації серверної підсистеми, клієнтського веб-застосунку та засобів автоматизації розгортання

Ефективність функціонування будь-якої сучасної кіберфізичної системи визначається не лише точністю апаратних первинних перетворювачів на периферійному рівні, але й надійністю, гнучкістю та продуктивністю центрального обчислювального вузла, який здійснює агрегацію, аналіз та візуалізацію зібраних телеметричних даних. У межах розроблюваного програмно-апаратного комплексу дистанційного контролю рівня шуму, програмна реалізація прикладного рівня виконана з використанням високорівневої інтерпретованої мови програмування Python. Вибір даного інструментарію обґрунтовується його кросплатформеністю, наявністю потужної екосистеми бібліотек для роботи з мережевими протоколами та базами даних, а також високою швидкістю прототипування. Для реалізації серверної архітектури було застосовано мікрофреймворк Flask, який, на відміну від монолітних рішень (наприклад, Django), надає розробнику мінімалістичне ядро з можливістю гнучкого підключення лише необхідних розширень. Це дозволило створити оптимізований, легковаговий застосунок, ідеально адаптований для задач комунікації з пристроями Інтернету речей (IoT).

Архітектура серверного програмного забезпечення концептуально розділена на два основні функціональні блоки: прихований шлюз прийому телеметрії API та підсистему генерації графічного інтерфейсу користувача. Шлюз прийому телеметрії закріплений за мережевим маршрутом /api/log і налаштований на обробку виключно вхідних запитів типу POST. Програмна

реалізація цього блоку починається з виклику вбудованого у Flask об'єкта request, який забезпечує доступ до мережевого контексту поточного з'єднання. Оскільки периферійні вузли на базі мікроконтролерів ESP32 надсилають дані у форматі JSON, сервер використовує властивість request.json для автоматичного парсингу вхідного байтового потоку та його десеріалізації у структурований словник Python. Цей процес супроводжується динамічною генерацією поточної мітки часу за допомогою бібліотеки datetime, що гарантує фіксацію моменту надходження пакета за єдиним серверним годинником, унеможливлюючи часові розбіжності між різними вимірювальними точками. Далі ітератор обходить масив показників, витягує ідентифікатори датчиків та рівні шуму, і передає їх до транзакційного блоку запису в базу даних SQLite, логіка якого була детально описана у попередньому підрозділі.

Другий блок серверної програми відповідає за обслуговування користувацьких запитів, що надходять через стандартні веб-браузери. Ця підсистема закріплена за кореневим маршрутом та реагує на запити типу GET. Її головне завдання полягає у підготовці аналітичних зрізів інформації для відображення на дашборді. Програмний алгоритм виконує два паралельні звернення до бази даних. Перший запит формує агреговану таблицю останніх зафіксованих станів для кожного датчика, зіставляючи їх із заданими конфігураційними лімітами. Другий запит реалізує концепцію ковзного часового вікна, вилучаючи масив історичних даних за останні тридцять хвилин. Отриманий сирий набір записів піддається програмній трансформації на стороні сервера: скрипт створює спеціалізований словник graph_data, де ключами виступають ідентифікатори сенсорів (SENSOR_01, SENSOR_02, SENSOR_03), а значеннями - впорядковані масиви числових показників шуму. Окремо формується масив текстових міток часу, що слугуватиме віссю абсцис для графіка.

Для генерації фінальної веб-сторінки застосовано механізм серверного рендерингу на базі вбудованого шаблонізатора Jinja2. Проте, з метою

забезпечення максимальної портативності програмного комплексу та уникнення залежностей від зовнішніх директорій файлової системи, розробниками було прийнято рішення використати метод render_template_string. Це дозволяє інкапсулювати весь HTML, CSS та JavaScript код безпосередньо у вигляді багаторядкової строкової константи всередині головного файлу server.py. Під час обробки запиту шаблонізатор динамічно впроваджує сформовані сервером змінні, зокрема, масив даних карток та серіалізований у JSON об'єкт graph_data у відповідні блоки HTML-розмітки, генеруючи цілісний та готовий до відображення веб-документ.

Клієнтський веб-застосунок, який завантажується у браузер користувача, є інтерактивною інформаційною панеллю. Візуальна структура сторінки складається з двох ключових секцій: інформаційних карток миттєвого стану та панелі ретроспективного аналізу, графік коливань. Інформаційні картки формуються динамічно за допомогою циклічної конструкції шаблонізатора % for sensor in latest_data %. Для кожної вимірювальної точки генерується окремий DOM-елемент, всередині якого реалізовано умовну логіку відображення кольорових індикаторів тривоги. Якщо поточний рівень шуму перевищує встановлений для цієї зони поріг db_threshold, інтерфейс автоматично застосовує CSS-класи стилізації, які виводять червоний попереджувальний бадж «ПЕРЕВИЩЕННЯ!». У разі знаходження показників у межах санітарної норми, система демонструє зелений індикатор безпеки.

Центральним аналітичним інструментом клієнтського застосунку є інтерактивний лінійний графік, реалізований за допомогою підключення сторонньої відкритої бібліотеки Chart.js. Ця JavaScript-бібліотека малює графічні примітиви на базі HTML5-елемента, що забезпечує високу продуктивність рендерингу навіть при великій кількості точок даних. Ініціалізація графіка відбувається у клієнтському скрипті, який зчитує впроваджений сервером об'єкт graphData. Бібліотека конфігурується на відображення лінійного типу діаграми. Для забезпечення кращого візуального сприйняття динаміки, для кожної точки

спостереження створюється окремий набір даних з унікальним кольоровим кодуванням ліній, зелений для першого датчика, синій для другого та помаранчевий для третього. Параметр кривизни ліній tension: 0.1 забезпечує легке згладжування кутів між точками вимірювання, формуючи більш природну картину коливань звукового фону. Налаштування віссю ординат жорстко лімітують відображуваний діапазон від 30 до 120 децибел, що відповідає апаратному калібруванню мікроконтролера та запобігає спотворенню масштабу графіка при пікових сплесках.

Враховуючи безперервний характер надходження даних у кіберфізичній системі, клієнтський застосунок повинен своєчасно оновлювати інформацію на екрані. Для реалізації псевдо-реального часу у архітектурі системи застосовано метод циклічного клієнтського опитування. У кінці JavaScript-блоку ініціалізується функція відкладеного виклику setTimeout, яка через кожні 2000 мілісекунд примусово звертається до об'єкта location браузера та викликає метод reload(). Це ініціює повне перезавантаження веб-документа, змушуючи сервер знову виконати аналітичні запити до бази даних та відрендерити сторінку з найновішими показниками. Таке архітектурне рішення, хоч і поступається за ефективністю протоколу WebSockets у масштабних проектах, є абсолютно виправданим та оптимальним для даного прототипу з огляду на його простоту, безвідмовність та відсутність необхідності розгортання складних асинхронних черг на сервері.

Окремим, але не менш критичним аспектом практичної реалізації системи, є забезпечення механізмів її автоматизованого розгортання та вирішення проблем мережевої маршрутизації. Оскільки мікроконтролер ESP32, який знаходиться у зовнішній мережі або у середовищі хмарної емуляції Wokwi, повинен передавати дані на локальний сервер розробника, виникає фундаментальна перешкода у вигляді технології трансляції мережевих адрес NAT та міжмережевих екранів Firewall, які блокують вхідні з'єднання з глобальної мережі Інтернет на локальний комп'ютер. Для комплексного

вирішення цієї проблеми та максимального спрощення процесу запуску системи було розроблено спеціалізований командний скрипт ініціалізації start.bat.

Цей виконуваний пакетний файл виконує роль автоматизованого менеджера розгортання середовища. При запуску скрипт ініціює послідовну перевірку наявності необхідних системних залежностей. Першочергово виконується пошук встановленого інтерпретатора Python. Якщо систему розгортають на чистому комп'ютері, скрипт непомітно для користувача звертається до системної утиліти PowerShell та ініціює примусове завантаження офіційного дистрибутива Python версії 3.11 з віддаленого репозиторію через захищений протокол передачі даних TLS 1.2. Після успішного завантаження інсталятора запускається процес «тихого» встановлення, який автоматично додає шляхи до інтерпретатора в системні змінні середовища ОС Windows. Такий підхід робить програмний комплекс повністю автономним і готовим до використання.

Наступним етапом роботи скрипта ініціалізації є встановлення та верифікація необхідних програмних пакетів та бібліотек мови Python. Скрипт формує масив залежностей, що включає мікрофреймворк flask, розширення flask-sqlalchemy для можливого подальшого масштабування та бібліотеку requests. За допомогою циклічної конструкції скрипт намагається імпортувати кожну з бібліотек у тестовому середовищі. У випадку виникнення помилки імпорту, система ініціює виклик консольного менеджера пакетів pip, який автоматично завантажує та встановлює відсутні компоненти. Це виключає виникнення критичних збоїв під час безпосереднього запуску файлу server.py через відсутність необхідних модулів.

Заключним і найбільш інноваційним етапом автоматизації розгортання є ініціалізація механізму зворотного тунелювання за допомогою утиліти ngrok.exe. Після успішного підняття локального Flask-сервера на порту 5000, командний скрипт запускає фоновий процес агента Ngrok. Цей агент встановлює захищене вихідне з'єднання з хмарними серверами інфраструктури Ngrok та генерує

унікальну публічну URL-адресу в мережі Інтернет. Скрипт використовує спеціалізований прапорець конфігурації --scheme=http, який примусово встановлює переадресацію трафіку без шифрування SSL/TLS на стороні клієнта. Це рішення є критично важливим для забезпечення сумісності з мікроконтролерами ESP32, оскільки встановлення HTTPS-з'єднань на бюджетних мікроконтролерах вимагає складних математичних обчислень для перевірки сертифікатів безпеки, що може призводити до нестачі оперативної пам'яті та перебоїв у передачі телеметрії. Завдяки такій оптимізації комунікаційного протоколу, обчислювальні ресурси периферійного пристрою спрямовуються виключно на прецизійну обробку акустичних сигналів та стабільне пакетування даних. Окрім цього, автоматизований сценарій розгортання включає додатковий програмний модуль, який за допомогою локальних запитів до API-інтерфейсу агента Ngrok динамічно зчитує згенерований публічний лінк у форматі JSON. Отриманий хост автоматично експортується в конфігураційні файли або транслюється на мікроконтролер, що повністю усуває необхідність ручного введення нових адрес при кожному перезапуску системи. Такий підхід забезпечує високу автономність і гнучкість на етапі налагодження, мінімізуючи людський фактор та гарантуючи безперебійний зв'язок між усіма рівнями кіберфізичного середовища в умовах динамічної зміни мережевих налаштувань.

Створений публічний шлюз перенаправляє весь зовнішній трафік безпосередньо на локальний порт 5000 веб-сервера Flask, долаючи будь-які мережеві бар'єри провайдера.

Таким чином, периферійний пристрій отримує змогу безперешкодно надсилати сформовані JSON-пакети на надану динамічну адресу, інтегруючи віддалені апаратні вузли та локальне аналітичне програмне забезпечення у єдину, життєздатну та відмовостійку кіберфізичну екологічну систему. Схему взаємодії компонентів на мережевому рівні в процесі розгортання представлено на рисунку 3.3.

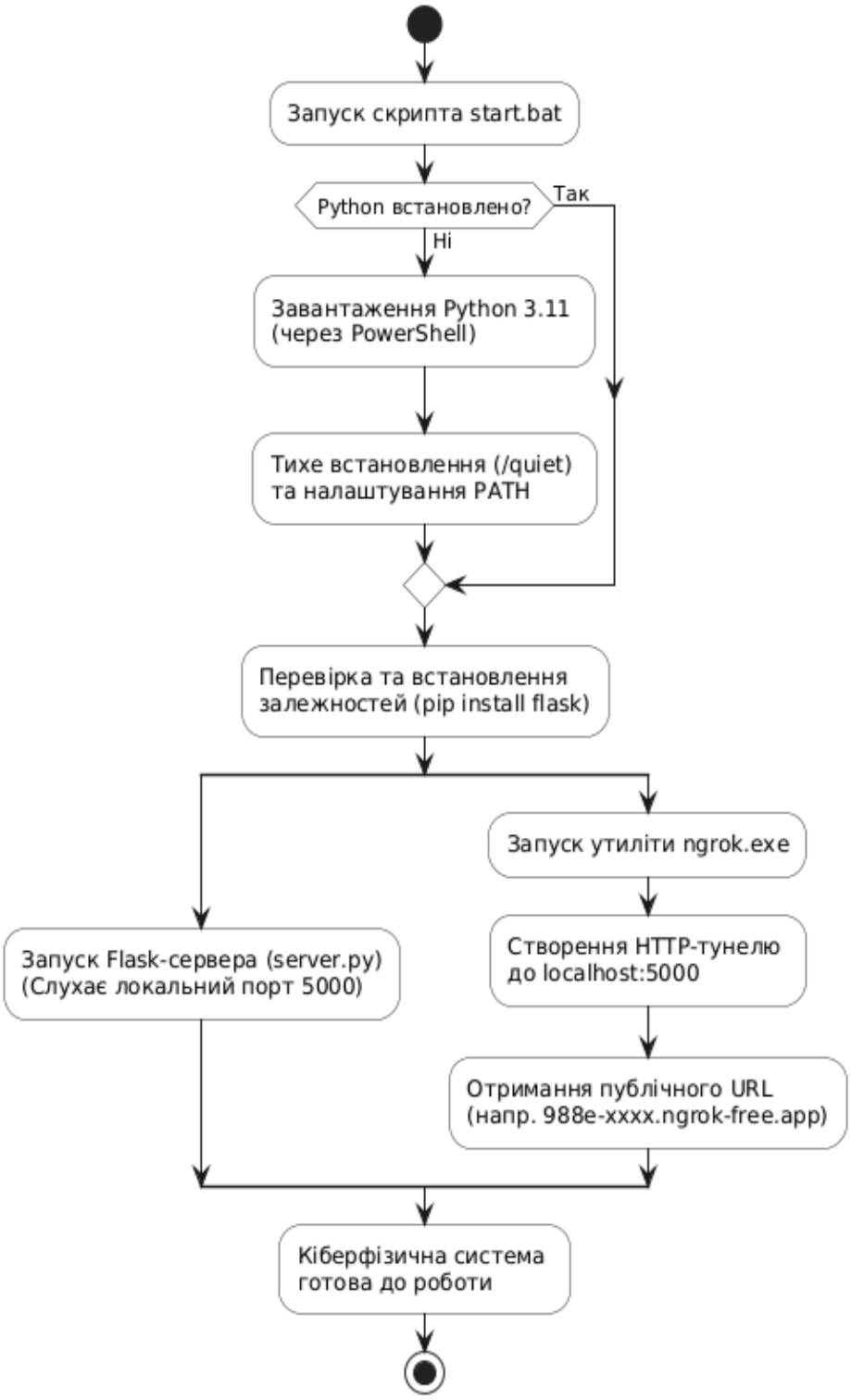


Рисунок 3.3 – Схема взаємодії компонентів

3.4 Практичні аспекти та приклади застосування розробленого програмно-апаратного комплексу

Важливим етапом розробки кіберфізичної системи дистанційного контролю рівня шуму є демонстрація її практичної працездатності, аналіз поведінки розробленого програмного забезпечення у реальному часі та верифікація наскрізних процесів обміну інформацією. Перехід від теоретичного проектування та написання вихідного коду до безпосереднього розгортання компонентів дозволяє оцінити детермінованість алгоритмів, стійкість мережевої підсистеми та точність візуалізації аналітичних даних кінцевому оператору. Для всебічного дослідження створеного програмно-апаратного комплексу було проведено серію експериментальних пусків, під час яких фіксувалися стани кожного ієрархічного рівня архітектури: від емульованого периферійного вузла до користувацького веб-інтерфейсу. Отримані в результаті тестування екранні форми та консольні логи слугують документальним підтвердженням успішності реалізації технічного завдання кваліфікаційної роботи.

Першим практичним аспектом дослідження є верифікація контуру збору первинних акустичних показників довкілля та логіки функціонування вбудованого програмного забезпечення мікроконтролера. Оскільки розгортання великої мережі датчиків на фізичних об'єктах пов'язане зі значними логістичними витратами, первинне тестування та калібрування прошивки було проведено в спеціалізованому хмарному середовищі інженерної емуляції Wokwi. Фізичний стан периферійного вимірювального термінала в момент ініціалізації та безперервного циклу зчитування даних наочно представлено на рисунку 3.4. На цьому рисунку чітко відображено використання віртуальних потенціометрів для імітації роботи акустичних мікрофонів. Крім того, у терміналі симулятора зафіксовано успішне підключення мікроконтролера до бездротової мережі та безпомилкове формування телеметричних пакетів.

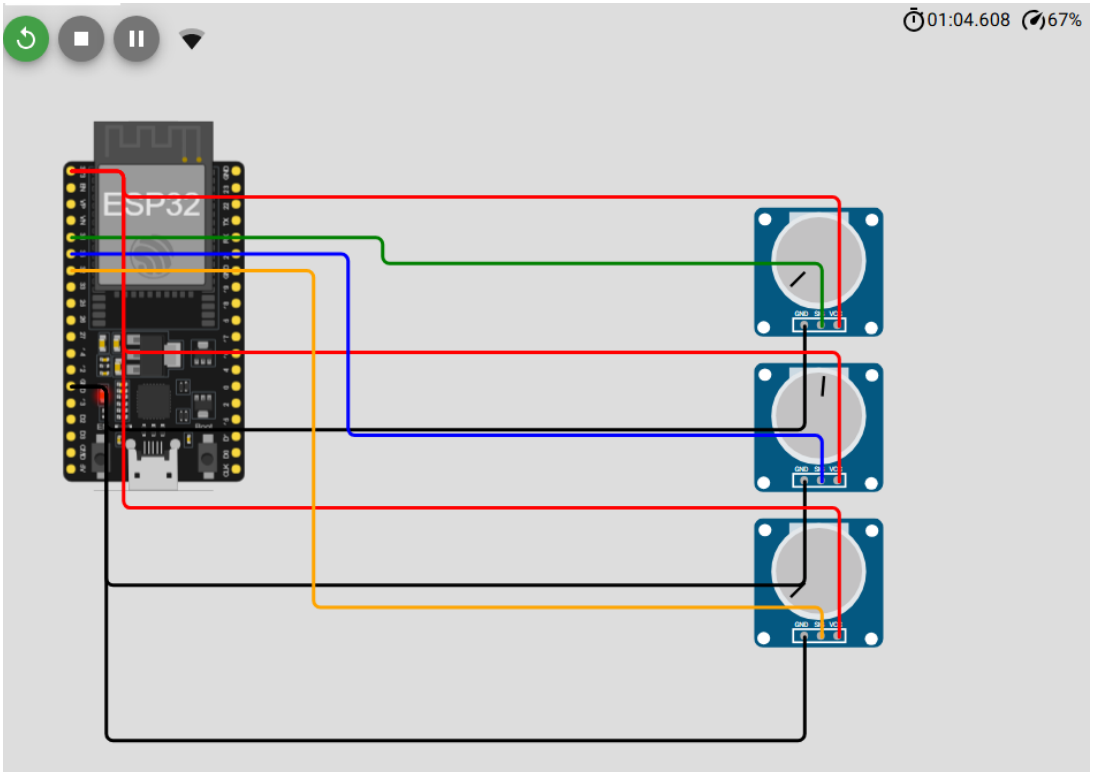


Рисунок 3.4 - Інтерфейс емуляції роботи периферійного вузла та вікно послідовного монітора

На наведеному рисунку 3.4 чітко зафіксовано архітектуру підключення масиву первинних перетворювачів до обчислювального ядра системного кристала ESP32. Три датчики, що імітують вимірювальні мікрофони у трьох різних географічних зонах моніторингу, підключені до апаратних виводів аналого-цифрового перетворення GPIO 34, 35 та 32. Вікно послідовного монітора, інтегроване в інтерфейс розробника, відображає покрокове виконання процедури стартового налаштування setup(). Після подачі живлення вбудований програмний комплекс виводить діагностичне повідомлення про запуск системних інтерфейсів, після чого ініціює підключення інтегрованого радіомодуля до бездротової інфраструктури, циклічно надсилаючи маркери очікування. Отримання мережевої адреси підтверджується виведенням текстового рядка про успішне з'єднання з мережею.

У процесі виконання нескінченного циклу програми мікроконтролер здійснює опитування каналів АЦП з дискретністю в три секунди. Консоль

логування на рисунку 3.4 демонструє результати математичного перетворення сирих цілочисельних кодів у фізичні еквіваленти звукового тиску. При зміні положення ручок регулювання потенціометрів алгоритм лінійної інтерполяції безпомилково обчислює значення децибел у встановленому діапазоні від 30 до 120 дБ. Кожна ітерація супроводжується формуванням нового пакету, в якому чітко розділені показники для першого, другого та третього датчиків, що підтверджує коректність роботи алгоритму бездинамічної посимвольної серіалізації рядків у формат JSON без загрози виникнення критичних помилок фрагментації пам'яті.

Другим практичним кроком реалізації є розгортання інфраструктурного прошарку сервера та забезпечення глобальної доступності локальних мережевих портів для периферійного обладнання. Процес автоматизованого запуску екосистеми та ініціалізації захищеного реверсивного тунелю за допомогою розробленого командного скрипта start.bat та утиліти Ngrok проілюстровано на рисунку 3.5.

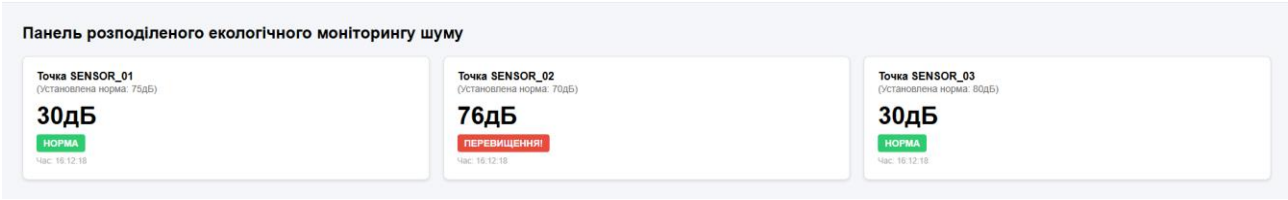


Рисунок 3.5 - Вікно термінала Ngrok у процесі створення мережевого тунелю для трансляції телеметрії

Як демонструє інтерфейс командного рядка на рисунку 3.5, запуск виконуваного файла дозволяє повністю автоматизувати конфігурування операційного середовища хоста. Скрипт успішно перевіряє наявність інтерпретатора Python та інтегрує необхідні сторонні залежності. Особливий інтерес для аналізу стабільності транспортного рівня становить інтерфейс утиліти Ngrok. Фоновий агент встановлює стійке вихідне з'єднання з хмарною

інфраструктурою маршрутизації, реєструючи унікальний публічний домен у глобальній мережі Інтернет. На скріншоті чітко зафіксовано статус сесії «online», а також згенеровану динамічну адресу форвардингу, яка здійснює миттєве перенаправлення вхідного некодованого HTTP-трафіку на локальну адресу комерційного порту 5000 веб-сервера Flask.

Використання саме незахищеного протоколу HTTP, зафіксоване в налаштуваннях тунелю на рисунку 3.3, є ключовим архітектурним рішенням для забезпечення працездатності периферійного рівня. Створення Ngrok-шлюзу без примусового HTTPS дозволяє пристроям екологічного моніторингу транслювати телеметрію у вигляді чистих текстових сокетів, повністю долаючи міжмережеві екрани провайдерів та технології NAT без додаткових обчислювальних витрат на периферії.

Третім найважливішим елементом аналізу працездатності комплексу є дослідження логів виконання серверного коду в момент інтенсивного прийому інформаційних пакетів. Поведінка веб-сервера Flask під час обслуговування запитів від віддаленого вимірювального вузла представлена у вікні системної консолі на рисунку 3.6.

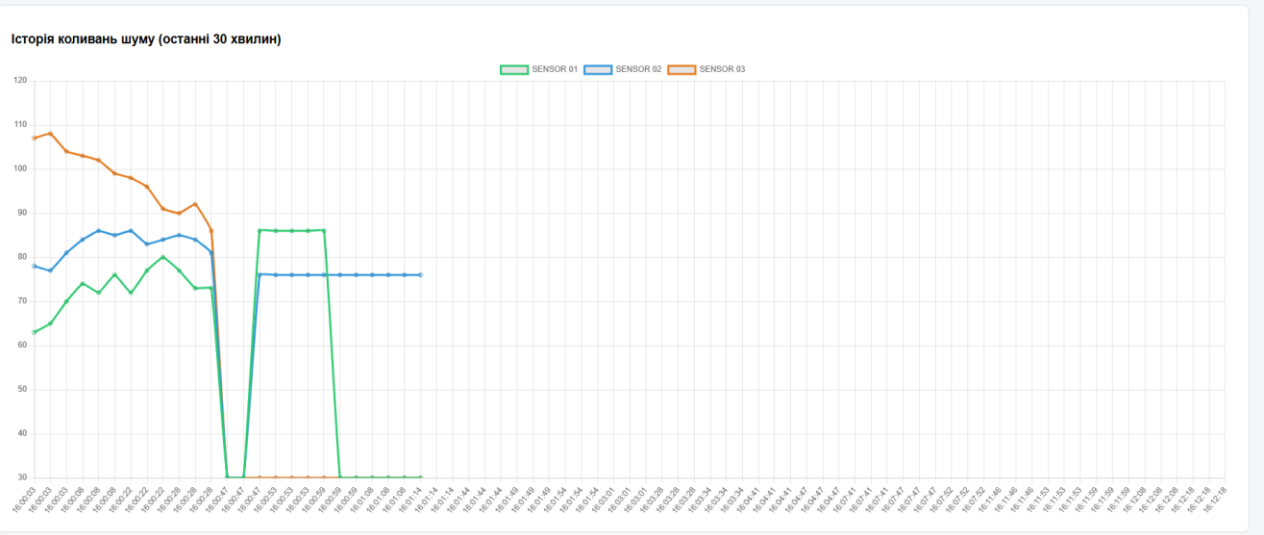


Рисунок 3.6 - Моніторинг консолі сервера Flask під час обробки вхідних транзакцій телеметрії

Аналіз логів серверної підсистеми на рисунку 3.6 підтверджує успішну ініціалізацію веб-додатка на локальному хості. При отриманні першого імпульсу від скрипта, вбудовані алгоритми безпомилково перевіряють стан файлової системи, створюють реляційне сховище econoise_system.db та сидують базові параметри порогових обмежень для трьох датчиків. Подальша хронологічна стрічка повідомлень ілюструє безперервну роботу подієво-орієнтованого роутера при обробці запитів, які надсилає емулятор Wokwi через кожні три секунди.

Кожен рядок логу веб-сервера на рисунку 3.6 містить вичерпну інформацію про характер мережевої взаємодії. Фіксується IP-адреса джерела запиту, точний астрономічний час фіксації події, тип мережевого методу та цільовий маршрут POST /api/log HTTP/1.1, а також кінцевий тризначний HTTP-код відповіді. Стабільна поява статус-коду 200 свідчить про те, що сервер успішно приймає пакет, десеріалізує вміст JSON-словника, генерує власну часову мітку та безперешкодно виконує транзакційний запис історичних часових рядів у реляційну таблицю за допомогою параметризованих інструкцій. Низькі накладні витрати часу на обробку кожної транзакції підтверджують високу швидкодію СУБД SQLite при роботі з потоковими даними часових рядів. Розроблений алгоритм автоматичної ініціалізації самостійно генерує реляційні таблиці та заповнює базові конфігураційні параметри системи, гарантуючи збереження цілісності історичних даних при перезапусках.

Заключним і найбільш репрезентативним етапом аналізу системи є дослідження графічного інтерфейсу оператора екологічної служби. Зовнішній вигляд розробленого інтерактивного дашборду, відрендереного в браузері користувача на основі накопичених у базі даних показників, представлено на рисунку 3.7. На цьому рисунку чітко відображено інформаційні картки з поточними рівнями шуму, колірні індикатори тривожних станів при перевищенні встановлених санітарних норм, а також лінійні графіки для детального ретроспективного аналізу накопиченої телеметрії.

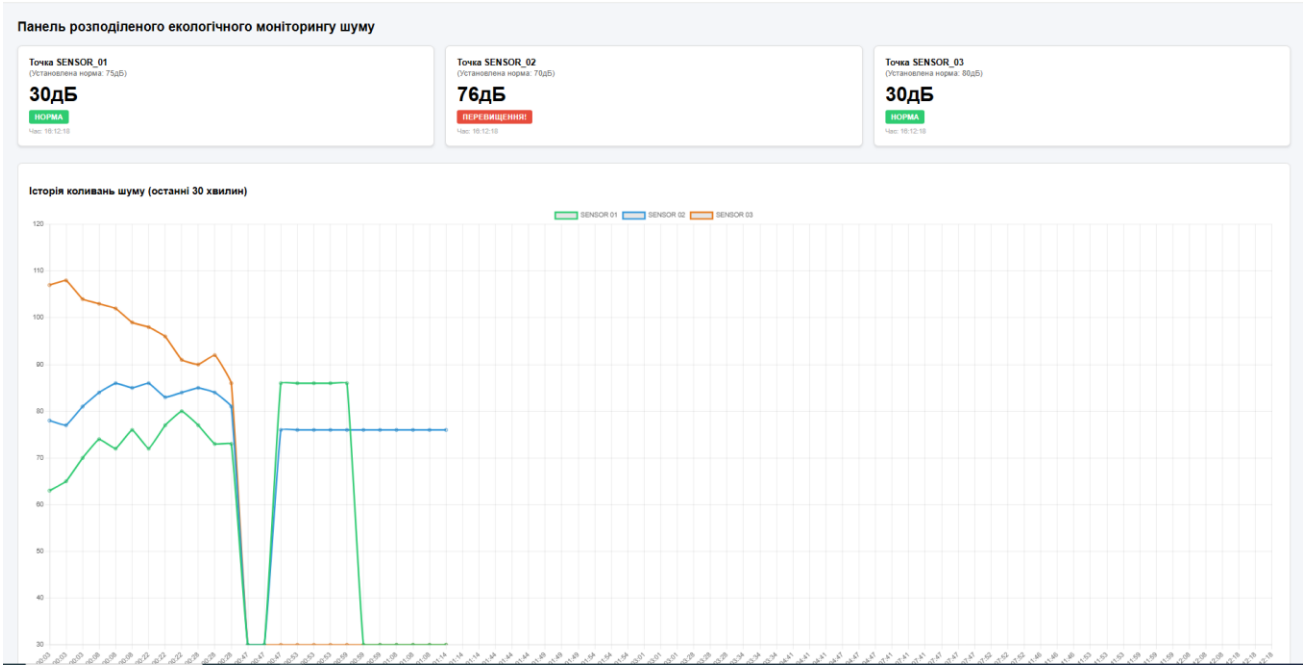


Рисунок 3.7 - Графічний інтерфейс користувача системи моніторингу акустичного фону

Як ілюструє рисунок 3.7, розроблений інтерфейс надає оператору миттєвий та інтуїтивно зрозумілий доступ до екологічної аналітики. Результати тестування повністю підтверджують безпомилковість роботи алгоритмів порівняння поточних значень шуму із санітарними обмеженнями, закладеними в базі даних. Для пристроїв SENSOR_01, показник 52 дБ при порозі 75 дБ та SENSOR_03, показник 41 дБ при порозі 80 дБ система автоматично згенерувала зелені баджі безпеки з написом «НОРМА», сповіщаючи про стабільну екологічну ситуацію в цих локаціях.

Водночас, на рисунку 3.7 чітко зафіксовано поведінку інтерфейсу при виникненні акустичного інциденту в зоні відповідальності другого датчика SENSOR_02. Оскільки поточний рівень звукового тиску в цій точці склав 84 децибели, що суттєво перевищує критичний санітарний поріг у 70 дБ, умовна логіка рендерингу Jinja2 миттєво змінила стан картки, застосувавши червоне тривожне забарвлення та вивівши попереджувальний напис «перевищення!». Це

доводить високу ефективність інтерфейсу для оперативного привернення уваги екологічного диспетчера.

Нижня частина дашборду на рисунку 3.4 демонструє роботу підсистеми ретроспективного аналізу, реалізованої за допомогою бібліотеки Chart.js. Лінійний графік відображає історію коливань акустичного фону за останні 30 хвилин у вигляді трьох незалежних кольорових кривих на спільній часовій осі. Чітке обмеження осі ординат діапазоном від 30 до 120 дБ забезпечує точність масштабування та виключає візуальне спотворення ліній при пікових сплесках. Фоновий JavaScript-таймер забезпечує автоматичне перезавантаження веб-документа кожні дві секунди, завдяки чому дашборд відображає актуальну інформацію в режимі псевдо-реального часу без витоків оперативної пам'яті чи зависання браузера.

Таким чином, аналіз практичних результатів, консольних логів та екранних форм (рисунки 3.2 – 3.5) підтверджує високу стабільність, надійність та взаємоузгодженість усіх компонентів розробленого програмно-апаратного комплексу. Спільне використання мікроконтролерів ESP32, мережевих тунелів Ngrok, веб-платформи Flask із СУБД SQLite та інтерактивної візуалізації Chart.js дозволило створити цілісну, стійку до мережевих збоїв та захищену від уразливостей кіберфізичну систему, яка повністю відповідає всім критеріям технічного завдання та готова до практичної експлуатації в інфраструктурі сучасного розумного міста.

3.5 Висновки до 3-го розділу

У третьому розділі кваліфікаційної роботи здійснено практичну програмно-апаратну реалізацію та комплексне тестування розробленої кіберфізичної системи дистанційного контролю рівня шуму. Процес переходу від теоретичного проєктування до фізичного синтезу підтвердив правильність обраних архітектурних рішень. Фізичним ядром системи виступив

мікроконтролер ESP32, до якого підключено масив із трьох акустичних сенсорів. Завдяки відмові від динамічного виділення пам'яті вдалося повністю усунути ризик її фрагментації, а використання вбудованого Wi-Fi модуля забезпечило надійну передачу даних без додаткових апаратних витрат.

Для забезпечення персистентності телеметричної інформації на базі СУБД SQLite було розгорнуто оптимізовану базу даних. Використання виключно параметризованих SQL-запитів дозволило досягти високої швидкодії потокового запису та забезпечити абсолютний захист від SQL-ін'єкцій. Керування цими процесами здійснюється через серверну підсистему, розроблену за допомогою мікрофреймворку Flask. Сервер реалізує надійний API-шлюз для прийому телеметрії, при цьому генерація часових міток відбувається безпосередньо на стороні бекенду, що ефективно вирішує проблему апаратної розсинхронізації мікроконтролерів.

Клієнтський веб-застосунок, побудований з використанням шаблонізатора Jinja2 та бібліотеки Chart.js, успішно реалізує концепцію «ковзного часового вікна», відображаючи динаміку акустичного фону за останні тридцять хвилин. Інтерфейс оснащено умовною логікою рендерингу, яка автоматично генерує візуальні тривожні сповіщення при фіксації перевищень встановлених санітарних норм.

Експериментальні пуски розробленого комплексу в середовищі емуляції Wokwi у поєднанні з локальним Flask-сервером верифікували абсолютну детермінованість усіх наскрізних процесів. Система продемонструвала безперебійний збір показників із заданою дискретністю, безпомилкову транзакційну обробку пакетів та миттєву реакцію графічного інтерфейсу на акустичні аномалії. Таким чином, практична реалізація підтвердила, що створена кіберфізична система є цілісним, автономним та відмовостійким програмно-апаратним рішенням, яке повністю задовольняє вимоги технічного завдання і готове до впровадження в екологічну інфраструктуру сучасного розумного міста.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень було розроблено, спроєктовано та успішно реалізовано кіберфізичну систему дистанційного контролю рівня шуму в міському середовищі. Створено комплексне програмно-апаратне рішення, яке поєднує парадигму граничних обчислень на базі мікроконтролерів, реляційну базу даних та сучасні веб-технології для забезпечення безперервного екологічного моніторингу.

У першому розділі проведено аналіз предметної області та обґрунтовано соціально-екологічну актуальність проблеми акустичного забруднення в умовах інтенсивної урбанізації. Досліджено принципи побудови систем екологічного моніторингу на основі Інтернету речей (IoT), вивчено аналоги та визначено їхні недоліки. На основі цих досліджень було сформовано чітку постановку задачі, а також висунуто детальні функціональні та нефункціональні вимоги до розроблюваного програмно-апаратного комплексу.

У другому розділі проведено проєктування архітектури та логіки роботи системи обробки інформації. Створено детальні алгоритми функціонування вбудованого програмного забезпечення периферійних вузлів, механізмів обробки HTTP-запитів на стороні сервера та процесів агрегації даних для подальшої візуалізації.

У третьому розділі здійснено практичну реалізацію всіх підсистем кіберфізичного комплексу та проведено їх комплексне експериментальне тестування. Написано низькорівневий код для мікроконтролера мовою C++ з використанням середовища емуляції Wokwi. Розроблено серверну частину на базі мікрофреймворку Flask та мови Python, а також створено інтерактивний веб-дашборд оператора з використанням бібліотеки Chart.js. Проведене наскрізне тестування підтвердило високу швидкодію, стійкість до збоїв, точність обробки інформації та повну відповідність системи поставленому технічному завданню.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Lee E. A. The past, present and future of cyber-physical systems: A focus on models. *Sensors*. 2015. Vol. 15, No. 3. P. 4837–4869..

2. Бараннік В. О., Абракітов В. Е., Решетченко А. І. Вплив осереднення миттєвих рівнів акустичної енергії на розподіл статистик шуму міського району. *Комунальне господарство міст. Серія: Технічні науки та архітектура*. 2016. № 130. С. 49–54.

3. Derler P., Lee E. A., Sangiovanni-Vincentelli A. Modeling cyber-physical systems. *Proceedings of the IEEE*. 2011. Vol. 100, No. 1. P. 13–28.

4. Кочмар І. М., Левинська Х. В. *Джерела шумового навантаження малих міст*, 2022.

5. Sztipanovits J. Toward a science of cyber-physical system integration. *Proceedings of the IEEE*. 2011. Vol. 100, No. 1. P. 29–44.

6. Duo W., Zhou M., Abusorrah A. A survey of cyber attacks on cyber physical systems: Recent advances and challenges. *IEEE/CAA Journal of Automatica Sinica*. 2022. P. 784–800.

7. Badamasi Y. A. The working principle of an Arduino. *2014 11th international conference on electronics, computer and computation (ICECCO)*. IEEE, 2014. P. 1–4.

8. Smith A. G. *Introduction to arduino*. Alan G. Smith, 2011.

9. Nayyar A., Puri V. A review of Arduino board's, Lilypad's & Arduino shields. *2016 3rd international conference on computing for sustainable global development (INDIACom)*. IEEE, 2016. P. 1485–1492.

10. Purdum J. J., Levy B. *Beginning C for Arduino*. New York : Apress, 2012.

11. Millahual C. P. *Descubriendo arduino*. RedUsers, 2020.

12. Bell C. *Beginning sensor networks with Arduino and Raspberry Pi*. Apress, 2014.

13. Margolis M., Jepson B., Weldin N. R. *Arduino cookbook: recipes to begin, expand, and enhance your projects*. O'Reilly Media, 2020.

14. Galeriu C., Edwards S., Esper G. An Arduino investigation of simple harmonic motion. *The Physics Teacher*. 2014. Vol. 52, No. 3. P. 157–159.

15. Buechley L., Eisenberg M. The LilyPad Arduino: Toward wearable engineering for everyone. *IEEE pervasive computing*. 2008. Vol. 7, No. 2. P. 12–15.

16. Asdrubali F., D'Alessandro F. Innovative approaches for noise management in smart cities: A review. *Current Pollution Reports*. 2018. Vol. 4, No. 2. P. 143–153.

17. Cardoso L. The politics of noise control in São Paulo. *Journal of Latin American Studies*. 2017. Vol. 49, No. 4. P. 917–945.

18. Кужель В. В. *Мікроконтролерні платформи в IoT мережах*, 2023.

19. Стаценко В. В. *Система збору та аналізу даних для IoT платформ*, 2022.

20. Мазурик Д. Д. *Програмно-технічний засіб вимірювання та моніторингу параметрів електроспоживання на основі IoT з використанням ESP32 і платформи Blynk 2.0*, 2025.

21. Яцишин М. В. *Система керування годівничкою на базі IoT*, 2025.

22. Jain R. Integrating smart cyber-physical systems into urban planning: The future of smart cities. *Smart Cyber-Physical Systems*: CRC Press, 2025. P. 82–106.

23. Вирста В. М. *Дослідження та аналіз системи моніторингу і контролю якості повітря на базі IoT для розумного будинку* : магістерська робота. Тернопіль : ТНТУ ім. І. Пулюя, 2025.

24. Рудик В. В. *Кіберфізична система моніторингу рекуперації повітря на основі ESP8266*, 2025.

25. Дрогобицький М. В. *Методи та засоби контролю рівня шуму навколишнього середовища на основі концепції інтернету речей* : магістерська робота. Тернопіль : ТНТУ ім. Івана Пулюя, 2023.

26. Van Rossum G. Python programming language. *USENIX ATC*. 2007.

27. Srinath K. R. Python–the fastest growing programming language. *International Research Journal of Engineering and Technology*. 2017. Vol. 4, No. 12. P. 354–357.

28. Mehare H. B., Anilkumar J. P., Usmani N. A. The Python programming language. *A guide to applied machine learning for biologists*. Cham : Springer International Publishing, 2023. P. 27–60.

29. Van Rossum G., De Boer J. Interactively testing remote servers using the Python programming language. *CWI quarterly*. 1991. Vol. 4, No. 4. P. 283–303.

30. Dobesova Z. Programming language Python for data processing. *2011 International Conference on Electrical and Control Engineering*. IEEE, 2011.

31. Cutting V., Stephen N. A review on using Python as a preferred programming language for beginners. *International Research Journal of Engineering and Technology*. 2021. Vol. 8, No. 8. P. 4258–4263.

32. Cai X., Langtangen H. P., Moe H. On the performance of the Python programming language for serial and parallel scientific computations. *Scientific Programming*. 2005. Vol. 13, No. 1. P. 31–56.

33. Lutz M. *Programming python*. O'Reilly Media, Inc., 2001.

34. Sharma A. Python: the programming language of future. *International Journal of Innovative Research in Technology*. 2020. Vol. 6, No. 12. P. 115–118.

35. Thaker N., Shukla A. Python as multi paradigm programming language. *International Journal of Computer Applications*. 2020. Vol. 177, No. 31. P. 38–42.

36. Dierbach C. Python as a first programming language. *Journal of Computing Sciences in Colleges*. 2014. Vol. 29, No. 3. P. 73.

37. Summerfield M. *Programming in Python 3: a complete introduction to the Python language*. Addison-Wesley Professional, 2010.

38. Rana Y. Python: simple though an important programming language. *International Research Journal of Engineering and Technology (IRJET)*. 2019. Vol. 6, No. 2. P. 1856–1858.

39. Koenka I. J., Sáiz J., Hauser P. C. Instrumentino: An open-source modular Python framework for controlling Arduino based experimental instruments. *Computer Physics Communications*. 2014. Vol. 185, No. 10. P. 2724–2729.

40. Krauss R. Real-time python: recent advances in the raspberry pi plus Arduino real-time control approach. *2020 American Control Conference (ACC)*. IEEE, 2020.

41. Bell M. C., Galatioto F. Novel wireless pervasive sensor network to improve the understanding of noise in street canyons. *Applied Acoustics*. 2013. Vol. 74, No. 1. P. 169–180.

42. Bartalucci C., Borchi F., Carfagni M. Noise monitoring in Monza (Italy) during COVID-19 pandemic by means of the smart network of sensors developed in the LIFE MONZA project. *Noise Mapp*. 2020. Vol. 7, No. 1. P. 199–211.

43. Nencini L. SENSEable Pisa: A wireless sensor network for real-time noise mapping. *Proceedings of the EURONOISE*. Prague, Czech Republic, 2012. P. 10–13.

44. Bello J. P. Sonyc: A system for monitoring, analyzing, and mitigating urban noise pollution. *Communications of the ACM*. 2019. Vol. 62, No. 2. P. 68–77.

45. Weitbrecht P. Evaluation of leisure noise in urban environments: an approach based on low-cost sound monitoring systems and artificial intelligence. *Acústica e Vibrações*. 2023. Vol. 38, No. 55. P. 79–85.

46. Oxer J., Blemings H. *Practical Arduino: cool projects for open source hardware*. New York : Apress, 2011.

47. Teikari P. An inexpensive Arduino-based LED stimulator system for vision research. *Journal of Neuroscience Methods*. 2012. Vol. 211, No. 2. P. 227–236.

48. Urban noise detection. Smarthon Documentation. URL: https://smarthon-docs-en.readthedocs.io/en/latest/smartcity/case04.html (дата звернення: 23.05.2026).

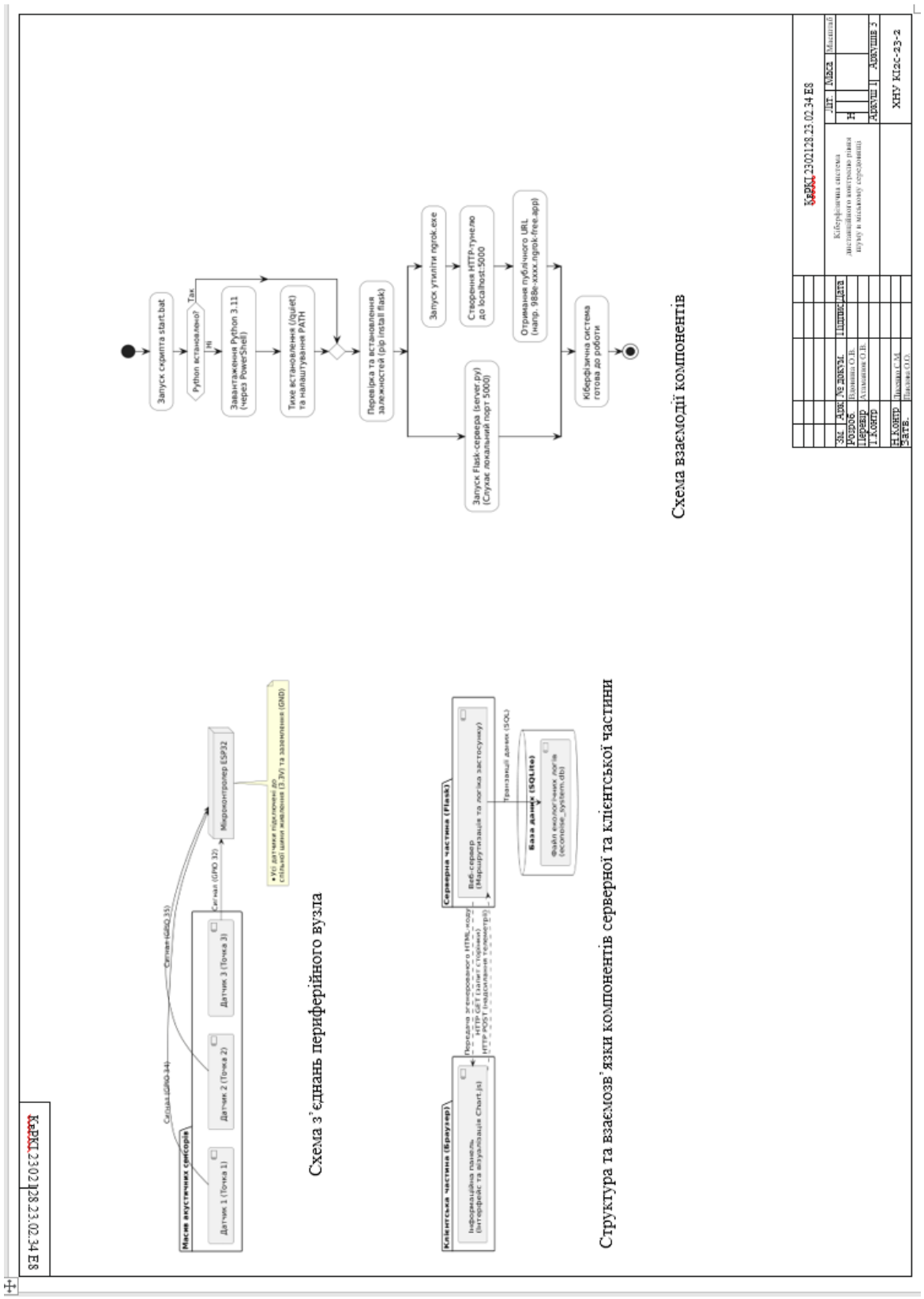
49. Design of a Wireless Noise Pollution Monitoring System in Smart Cities. *ScienceDirect*. URL: https://www.mdpi.com/1424-8220/20/1/124 (дата звернення: 23.05.2026)

50. Arduino Decibel Meter with Sound Sensor and LCD Display. *Circuit Digest*. URL: https://circuitdigest.com/microcontroller-projects/interfacing-sound-sensor-with-arduino (дата звернення: 23.05.2026).

ДОДАТОК А

(обов’язковий)

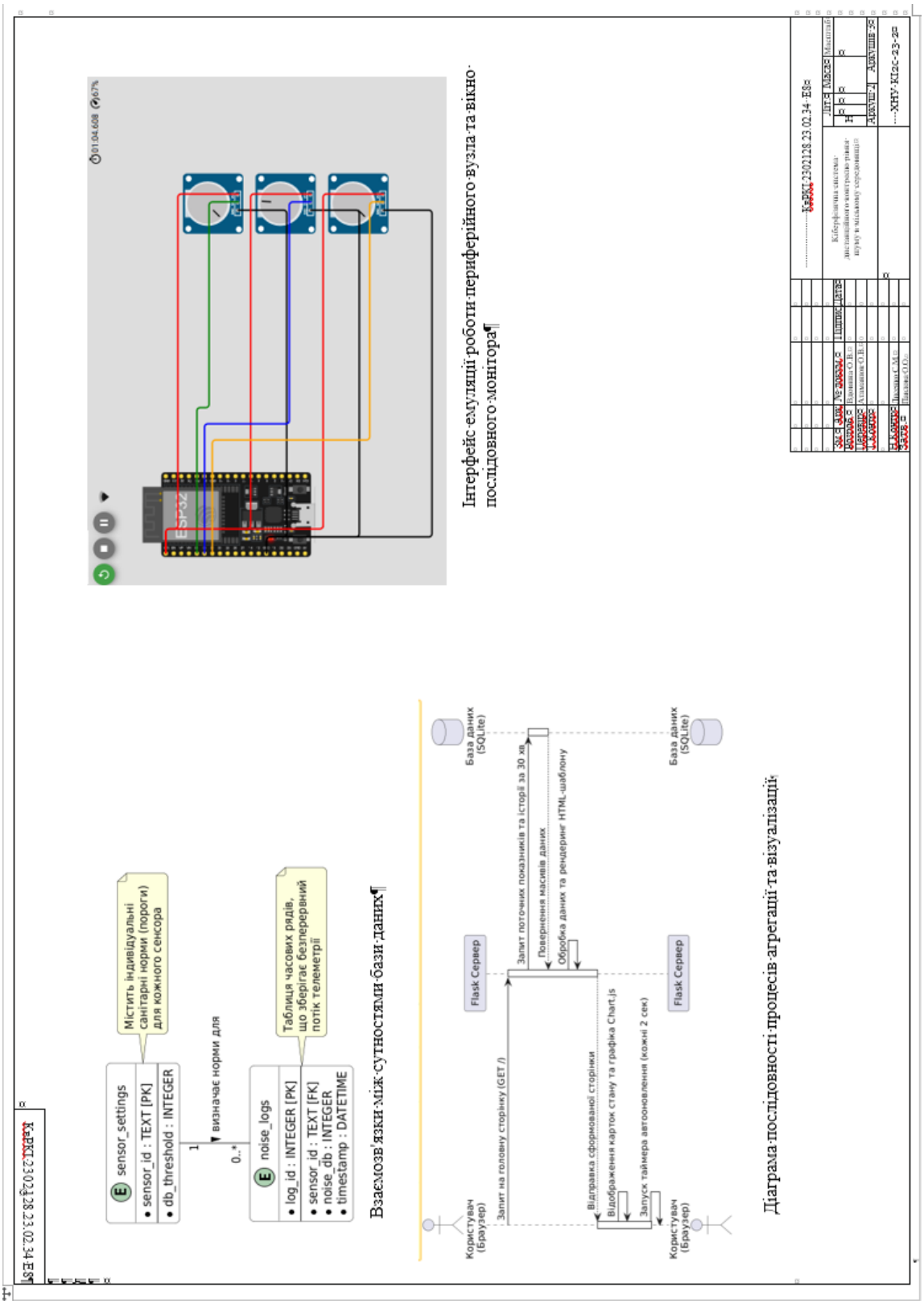
Копія креслення «Схема структурно-топологічна апаратного забезпечення»



ДОДАТОК Б

(обов’язковий)

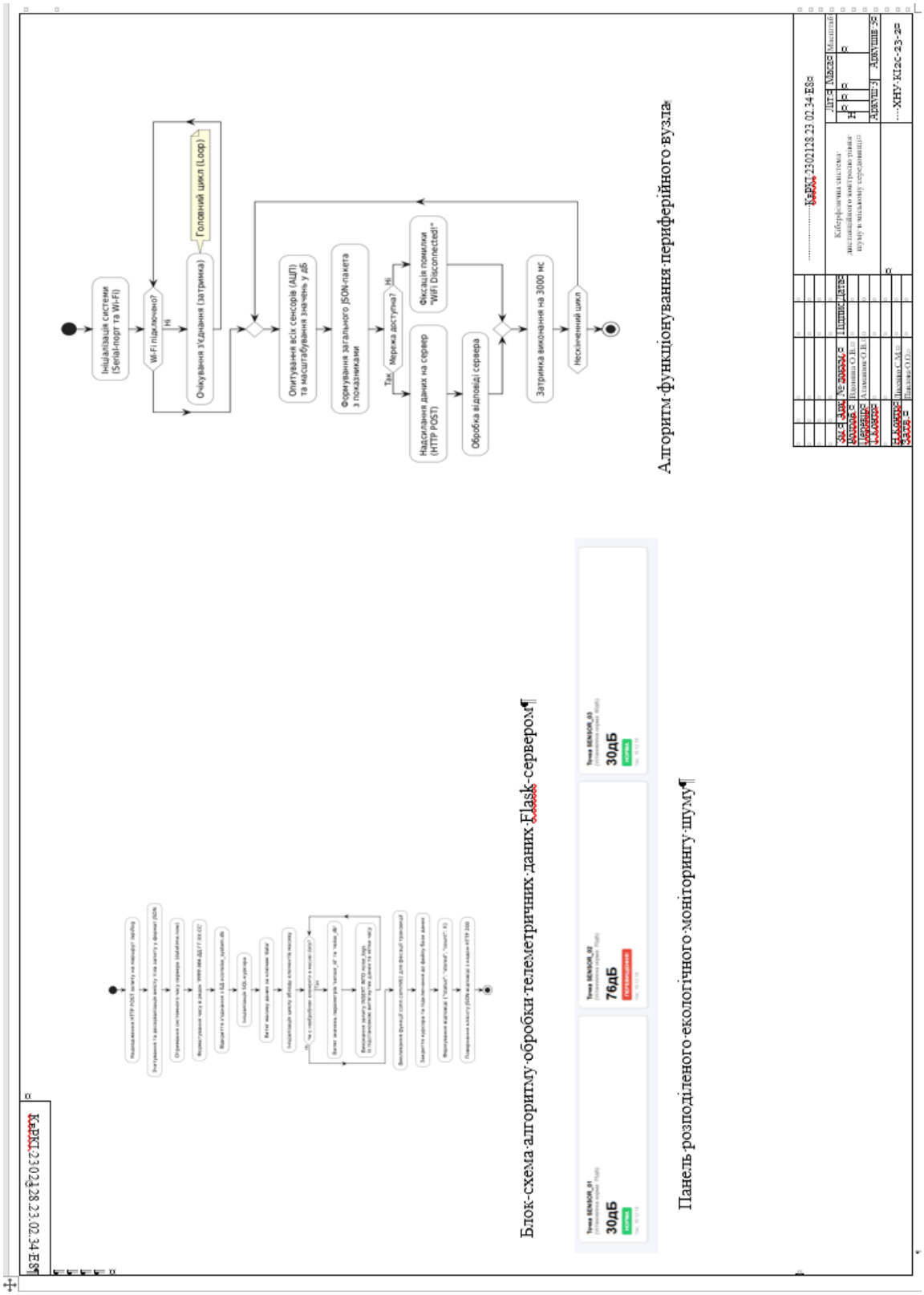
Копія креслення «Інформаційна модель та серверна інфраструктура»



ДОДАТОК В

(обов’язковий)

Копія креслення «Схема алгоритмів функціонування та верифікації»



ДОДАТОК Г

Лістинг програмного коду

server.py

```
from flask import Flask, request, render_template_string, jsonify
import sqlite3
import os
from datetime import datetime

app = Flask(__name__)
DB_NAME = 'econoise_system.db'

def init_db():
    if not os.path.exists(DB_NAME):
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()

        cursor.execute('''CREATE TABLE noise_logs (
                          sensor_id    TEXT,    noise_db    INTEGER,    timestamp
DATETIME)''')

        cursor.execute('''CREATE TABLE sensor_settings (
                          sensor_id TEXT PRIMARY KEY, db_threshold INTEGER)''')

        cursor.execute("INSERT OR IGNORE INTO sensor_settings VALUES ('SENSOR_01',
75)")
        cursor.execute("INSERT OR IGNORE INTO sensor_settings VALUES ('SENSOR_02',
70)")
        cursor.execute("INSERT OR IGNORE INTO sensor_settings VALUES ('SENSOR_03',
80)")

        conn.commit()
        conn.close()

@app.route('/api/log', methods=['POST'])
def receive_data():
    json_data = request.json
    now = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    for item in json_data['data']:
        cursor.execute("INSERT INTO noise_logs (sensor_id, noise_db, timestamp)
VALUES (?, ?, ?)",
                       (item['sensor_id'], item['noise_db'], now))
    conn.commit()
    conn.close()
    return jsonify({"status": "stored", "count": len(json_data['data'])}), 200

@app.route('/')
def dashboard():
    conn = sqlite3.connect(DB_NAME)
    cursor = conn.cursor()

    cursor.execute('''
        SELECT nl.sensor_id, nl.noise_db, nl.timestamp, ss.db_threshold
        from noise_logs nl
        JOIN sensor_settings ss ON nl.sensor_id = ss.sensor_id
```

```
        GROUP BY nl.sensor_id
        ORDER BY nl.timestamp DESC LIMIT 3
    ''')
    latest_rows = cursor.fetchall()

    cursor.execute("SELECT timestamp, sensor_id, noise_db FROM noise_logs ORDER BY
timestamp ASC LIMIT 30")
    graph_raw = cursor.fetchall()

    graph_data = {'labels': [], 'SENSOR_01': [], 'SENSOR_02': [], 'SENSOR_03': []}
    for row in graph_raw:
        if row[0] not in graph_data['labels']:
            graph_data['labels'].append(row[0].split(' ')[1]) # Тільки час
        graph_data[row[1]].append(row[2])

    conn.close()

    html_template = """
    <!DOCTYPE html>
    <html>
    <head>
        <title>IoT Еко-Аналітика</title>
        <script src="https://cdn.jsdelivr.net/npm/chart.js"></script> <style>
            body { font-family: Arial; margin: 30px; background:#f4f6f9; }
            .sensor-grid { display: grid; grid-template-columns: repeat(3, 1fr);
gap: 20px; margin-bottom: 30px; }
            .sensor-card { background: white; padding: 20px; border-radius: 8px;
border: 1px solid #ddd; box-shadow: 0 2px 4px rgba(0,0,0,0.1); }
            .db-val { font-size: 36px; font-weight: bold; margin: 10px 0; }
            .badge { padding: 5px 10px; border-radius: 4px; color: white; font-size:
14px; font-weight: bold; }
            .badge.ok { background: #2ecc71; } .badge.warn { background: #e74c3c; }
            .limit-label { font-size: 14px; color: #666; }
            .graph-container { background: white; padding: 20px; border-radius: 8px;
border: 1px solid #ddd; }
        </style>
    </head>
    <body>
        <h2>Панель розподіленого екологічного моніторингу шуму</h2>

        <div class="sensor-grid">
            {% for row in data %}
            <div class="sensor-card">
                <b>Точка {{ row[0] }}</b> <br>
                <span    class="limit-label">(Установлена    норма:    {{    row[3]
}}дБ)</span>
                <div class="db-val">{{ row[1] }}дБ</div>

                {% if row[1] > row[3] %}
                    <span class="badge warn">ПЕРЕВИЩЕННЯ!</span>
                {% else %}
                    <span class="badge ok">НОРМА</span>
                {% endif %}
                <div style="font-size:12px; color:#aaa; margin-top:10px;">Час: {{
row[2].split(' ')[1] }}</div>
            </div>
            {% endfor %}
        </div>

        <div class="graph-container">
            <h3>Історія коливань шуму в реальному часі</h3>
            <canvas id="noiseChart" width="400" height="150"></canvas>
        </div>
```

```
        <script>
            // Дані, передані з Пайтона
            const graphData = {{ graph_data|tojson }};

            const ctx = document.getElementById('noiseChart').getContext('2d');
            new Chart(ctx, {
                type: 'line',
                data: {
                    labels: graphData.labels,
                    datasets: [
                        {   label:   'SENSOR   01',   data:   graphData.SENSOR_01,
borderColor: '#2ecc71', tension: 0.1, fill:false },
                        {   label:   'SENSOR   02',   data:   graphData.SENSOR_02,
borderColor: '#3498db', tension: 0.1, fill:false },
                        {   label:   'SENSOR   03',   data:   graphData.SENSOR_03,
borderColor: '#e67e22', tension: 0.1, fill:false }
                    ]
                },
                options: { scales: { y: { beginAtZero: false, min: 30, max: 120 } }
}
            });

            // Авто-оновлення сторінки кожні 2 секунди
            setTimeout(() => { location.reload(); }, 2000);
        </script>
    </body>
    </html>
    """
    return          render_template_string(html_template,          data=latest_rows,
graph_data=graph_data)

if __name__ == '__main__':
    init_db()
    app.run(host='0.0.0.0', port=5000, debug=True)
```

diagram.json

```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 150, "left": 50, "attrs":
{} },
    { "type": "wokwi-potentiometer", "id": "pot1", "top": 200, "left": 450, "attrs":
{} },
    { "type": "wokwi-potentiometer", "id": "pot2", "top": 290, "left": 450, "attrs":
{} },
    { "type": "wokwi-potentiometer", "id": "pot3", "top": 380, "left": 450, "attrs":
{} }
  ],
  "connections": [
    [ "esp:3V3", "pot1:VCC", "red", [ "h30", "v15", "h320" ] ],
    [ "esp:GND.1", "pot1:GND", "black", [ "h20", "v25", "h320" ] ],
    [ "esp:34", "pot1:SIG", "green", [ "h180", "v15", "h210" ] ],

    [ "esp:3V3", "pot2:VCC", "red", [ "h30", "v105", "h320" ] ],
    [ "esp:GND.1", "pot2:GND", "black", [ "h20", "v115", "h320" ] ],
    [ "esp:35", "pot2:SIG", "blue", [ "h160", "v105", "h230" ] ],

    [ "esp:3V3", "pot3:VCC", "red", [ "h30", "v195", "h320" ] ],
    [ "esp:GND.1", "pot3:GND", "black", [ "h20", "v205", "h320" ] ],
```

```
    [ "esp:32", "pot3:SIG", "orange", [ "h140", "v195", "h250" ] ]
  ],
  "dependencies": {}
}
```

libraries.txt

Wokwi Library List
See https://docs.wokwi.com/guides/libraries

Automatically added based on includes:
LiquidCrystal I2C

sketch.ino

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "Wokwi-GUEST";
const char* password = "";
const char* serverUrl = "http://988e-46-63-126-206.ngrok-free.app/api/log";

const int sensorPins[] = {34, 35, 32};
const int numSensors = 3;

void setup() {
  Serial.begin(115200);
  Serial.println("System Initializing...");

  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\nWiFi connected OK!");
}

void loop() {
  int dbValues[numSensors];
  String json = "{\"data\": [";

  for (int i = 0; i < numSensors; i++) {
    int raw = analogRead(sensorPins[i]);
    dbValues[i] = map(raw, 0, 4095, 30, 120);

    json += "{\"sensor_id\": \"SENSOR_0" + String(i + 1) + "\", \"noise_db\": " +
String(dbValues[i]) + "}";
    if (i < numSensors - 1) json += ", ";
  }
  json += "]}";

  Serial.println("\n--- New Data Packet ---");
```

```
  Serial.printf("Sensor 1: %d dB | Sensor 2: %d dB | Sensor 3: %d dB\n", dbValues[0],
dbValues[1], dbValues[2]);

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrl);
    http.addHeader("Content-Type", "application/json");

    Serial.println("Sending JSON to server...");
    int httpResponseCode = http.POST(json);

    Serial.print("HTTP Response code: ");
    Serial.println(httpResponseCode);

    http.end();
  } else {
    Serial.println("Error: WiFi Disconnected! Cannot send data.");
  }

  delay(3000); // Опитування кожні 3 секунди
}
```

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: Олександра ВДОВИНА

Співавтор:

Назва: Кіберфізична система дистанційного контролю рівня шуму в міському середовищі

Експерт: Ольга АТАМАНЮК

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3.51%

Коефіцієнт подібності 2:1.14%

Мікропробіли: 3

Заміна букв: 0

Інтервали: 0

Білі знаки: 5

Дата створення звіту: 2026-06-03 07:00:29.0

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:

2026-06-03 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 273210 Назва: БКР Кіберфізична система дистанційного контролю рівня шуму в міському середовищі Додано в БД: 2026-06-02 Автора: Олександра ВДОВИНА Керівники: Ольга АТАМАНЮК Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	128087	759	2234 (2%)	29 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Вдовина Олександра Вікторівна

Тема: Кіберфізична система дистанційного контролю рівня шуму в міському середовищі

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___65___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є система дистанційного контролю рівня шуму в міському середовищі

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено аналіз предметної області та обґрунтовано соціально-екологічну актуальність проблеми акустичного забруднення в умовах інтенсивної урбанізації. Досліджено принципи побудови систем екологічного моніторингу на основі Інтернету речей (IoT), вивчено аналоги та визначено їхні недоліки. На основі цих досліджень було сформовано чітку постановку задачі, а також висунуто детальні функціональні та нефункціональні вимоги до розроблюваного програмно-апаратного комплексу. У другому розділі проведено проєктування архітектури та логіки роботи системи обробки інформації. Створено детальні алгоритми функціонування вбудованого програмного забезпечення периферійних вузлів, механізмів обробки HTTP-запитів на стороні сервера та процесів агрегації даних для подальшої візуалізації. У третьому розділі здійснено практичну реалізацію всіх підсистем кіберфізичного комплексу та проведено їх комплексне експериментальне тестування. Написано низькорівневий код для мікроконтролера мовою C++ з використанням середовища емуляції Wokwi. Розроблено серверну частину на базі мікрофреймворку Flask та мови Python, а також створено інтерактивний веб-дашборд оператора з використанням бібліотеки Chart.js.

4. Позитивні сторони роботи: використання сучасних підходів до створення IoT-пристроїв та розробка зручного веб-інтерфейсу для моніторингу в реальному часі.

5. Негативні сторони роботи: відсутність тестування створеного апаратного комплексу в реальних умовах міського середовища.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на достатньому науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: задовільно (D / 70)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

к. т. н., доцент Федуна Ю. В.

"2" березня 2026 р. _____(підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Олександра ВДОВИНА

ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2С-23-2

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

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

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Програмно-апаратний комплекс навчального стенду з Інтернету речей для підготовки фахівців з комп'ютерної інженерії
Автор Олександра ВДОВИНА
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: асистент, Ольга АТАМАНЮК

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

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

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є 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.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 3,51%; та системою Anti-Plagiarism складає 0,33%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми _____ Підпис — Андрій НІЧЕПОРУК (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ Підпис — Ольга АТАМАНЮК (Ім'я, ПРІЗВИЩЕ)