Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

СИСТЕМА МОНІТОРИНГУ ЗАБРУДНЕННЯ ПОВІТРЯ З АНАЛІЗОМ ВПЛИВУ НА ЗДОРОВ'Я ЛЮДЕЙ

AIR POLLUTION MONITORING SYSTEM WITH HEALTH IMPACT ANALYSIS

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІ-41
Пась Вадим Володимирович

(підпис)

Керівник:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Кваліфікаційну роботу
допущено до захисту
« 04 » червня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Лавренчук Світлана Василівна

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Тарас ТЕРЛЕЦЬКИЙ
« 10 » 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Пасю Вадиму Володимировичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Система моніторингу забруднення повітря з аналізом впливу на здоров'я людей*

Керівник роботи *к.т.н., доцент Лавренчук Світлана Василівна*

затверджені наказом закладу вищої освіти від «04» січня 2025 року № 11/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.06.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області та різні інтернет-ресурси технічного спрямування.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Огляд існуючих рішень та постановка задач
Апаратна частина
Розробка програмного забезпечення
Висновки

5. Перелік графічного (ілюстративного) матеріалу:
Електрична схема системи збору інформації
Технології розробки програмного забезпечення
Функціональна схема
Інтерфейс розробленої системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Огляд існуючих рішень та постановка задач	*Лавренчук С.В., доцент*		
Апаратна частина	*Лавренчук С.В., доцент*		
Розробка програмного забезпечення	*Лавренчук С.В., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Лавренчук С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І.,ст.викладач*		

7. Дата видачі завдання 10.01.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми, аналіз предметної області та наявних рішень*	до 10.02.2025 р.	Виконано
2.	*Вибір апаратної та програмної бази для проєкту*	до 02.03.2025 р.	Виконано
3.	*Проєктування та тестування системи моніторингу*	до 02.04.2025 р.	Виконано
4.	*Висновки та пропозиції*	до 10.04.2025 р.	Виконано
5.	*Формування списку використаних джерел*	до 15.04.2025 р.	Виконано
6.	*Формування додатків*	до 02.05.2025 р.	Виконано
7.	*Оформлення ілюстративного матеріалу*	до 10.05.2025 р.	Виконано
8.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	до 15.05.2025 р.	Виконано
9.	*Нормоконтроль*	до 30.05.2025 р.	Виконано
10.	*Інструментальна перевірка на академічний плагіат*	до 03.06.2025 р	Виконано
11.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	до 11.06.2025 р.	Виконано

Здобувач вищої освіти _____ Пась В.В.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Лавренчук С.В.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Пась В. В. Система моніторингу якості повітря з аналізом впливу на здоров’я людей. Рукопис.

Кваліфікаційна робота бакалавра ОП «Комп’ютерна інженерія» спеціальності 123 Комп’ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел і додатків.

У першому розділі обґрунтовано актуальність дослідження, розглянуто вплив забрудненого повітря на здоров’я населення, визначено ключові екологічні показники, що формують індекс якості повітря. Проведено огляд існуючих рішень у сфері моніторингу (European Air Quality Index, AirVisual, PurpleAir, Sensor.Community), а також проаналізовано їх переваги й обмеження.

У другому розділі здійснено вибір апаратних і програмних засобів для реалізації системи. Розглянуто роботу сенсорів MQ-135, DHT22 та PMS5003, обґрунтовано використання мікроконтролера ESP32 як основного вузла обробки й передачі даних. Проведено моделювання та тестування системи у середовищі Wokwi, що дозволило перевірити функціональність усіх компонентів без фізичного прототипу. Також описано принцип функціонування апаратної частини та можливості її модульного розширення.

У третьому розділі розроблено програмне забезпечення системи: створено бекенд на Flask, реалізовано збереження даних у базі PostgreSQL, побудовано фронтенд на React.js із інтерактивною картою. Інтегровано модель машинного навчання для оцінки ризику здоров’я залежно від рівня забруднення. Проведено тестування REST API, візуалізації даних і функції сповіщень через Telegram.

Ключові слова: ESP32, якість повітря, сенсори MQ-135, DHT22, Wokwi, Flask, PostgreSQL, React.js, Leaflet, IoT, машинне навчання.

ANNOTATION

Pas V. Air quality monitoring system with analysis of impact on human health. Manuscript.

Bachelor’s qualification work OP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The qualification work consists of an introduction, three chapters, conclusions, a list of sources used and appendices.

The first section substantiates the relevance of the study, considers the impact of polluted air on public health, identifies key environmental indicators that form the air quality index. A review of existing monitoring solutions (European Air Quality Index, AirVisual, PurpleAir, Sensor.Community) is conducted, and their advantages and limitations are analyzed.

In the second section, hardware and software tools are selected for the implementation of the system. The operation of MQ-135, DHT22 and PMS5003 sensors is considered, and the use of the ESP32 microcontroller as the main data processing and transmission node is justified. The system was modeled and tested in the Wokwi environment, which allowed us to check the functionality of all components without a physical prototype. The principle of operation of the hardware part and the possibilities of its modular expansion are also described.

In the third section, the system software was developed: a backend was created on Flask, data storage was implemented in the PostgreSQL database, and a frontend was built on React.js with an interactive map. A machine learning model was integrated to assess health risk depending on the level of pollution. REST API, data visualization, and notification functions via Telegram were tested.

Keywords: ESP32, air quality, MQ-135 sensors, DHT22, Wokwi, Flask, PostgreSQL, React.js, Leaflet, IoT, machine learning.

ЗМІСТ

ВСТУП

«Забруднення атмосферного повітря є однією з головних екологічних проблем сучасності. За даними Всесвітньої організації охорони здоров'я (ВООЗ), понад 90 % населення світу дихає повітрям, яке не відповідає нормативам якості. Висока концентрація шкідливих речовин у повітрі, зокрема твердих частинок (PM2.5, PM10), оксидів азоту (NO_x), озону (O_3) та чадного газу (CO), суттєво підвищує ризик розвитку захворювань дихальної, серцево-судинної та імунної систем» [7].

У зв'язку з цим виникає потреба у створенні доступних і ефективних рішень для локального моніторингу якості повітря, які можуть застосовуватися як у міському середовищі, так і в сільській місцевості. Сучасні технології Інтернету речей (IoT) відкривають нові можливості для збору, аналізу та візуалізації екологічних даних у реальному часі. Крім того, інтеграція методів машинного навчання дозволяє проводити первинну оцінку ризику для здоров'я, ґрунтуючись на фактичних екологічних показниках.

Метою даної кваліфікаційної роботи є розробка інтегрованої системи моніторингу якості повітря з базовим аналізом впливу екологічних показників на здоров'я людини. Система має забезпечувати збір даних за допомогою сенсорів, їх обробку на мікроконтролері ESP32, локальне виведення інформації, а також збереження та візуалізацію даних через веб-інтерфейс.

Об'єктом дослідження є інформаційно-аналітична система моніторингу екологічних параметрів середовища.

Предметом дослідження є програмно-апаратний комплекс для моніторингу якості повітря та аналізу його впливу на здоров'я людей, побудований з використанням сенсорів, мікроконтролера ESP32 та сучасних засобів програмної інженерії.

Для досягнення поставленої мети необхідно виконати такі завдання:

– підібрати мікроконтролер та інші складові системи збору даних;

– розробити програмну модель прототипу у середовищі Wokwi для перевірки працездатності логіки без використання фізичних компонентів;

– спроєктувати серверну частину для зберігання екологічних даних у базі даних PostgreSQL з використанням Flask API;

– інтегрувати алгоритм машинного навчання, що класифікує рівень екологічного ризику для здоров’я людини;

– візуалізувати дані у фронтенді, побудованому на React.js та Leaflet.js;

– запропонувати можливості модульного розширення системи, у тому числі за рахунок нових сенсорів і каналів передачі даних.

Модель може бути впроваджена в навчальний процес як демонстраційний стенд для моніторингу довкілля.

Матеріали кваліфікаціної роботи обговорювалися на міжнародній науково-практичній конференції молодих вчених та студентів «Програмне та апаратне забезпечення в інформаційних технологіях» [31], яка відбулася 6 травня 2025 року в місті Луцьку.

Публікації. За результатами дослідження опубліковано тези [32] (додаток А).

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Актуальність теми

«Якість повітря є одним із найважливіших чинників, що впливають на здоров’я людей, стан довкілля та загальний рівень життя. За даними Всесвітньої організації охорони здоров’я, щороку понад 7 мільйонів людей у світі помирають внаслідок хвороб, спричинених забрудненням повітря» [19]. Найбільше шкоди негативні фактори можуть завдати дітям, людям літнього віку та тим, хто страждає на хронічні захворювання дихальної або серцево-судинної систем.

«Особливу тривогу викликає ситуація в містах, де рівень викидів твердих частинок (PM2.5 та PM10), діоксиду азоту (NO_2), озону (O_3), чадного газу (CO) часто перевищує безпечні норми, що встановлені ВООЗ та іншими екологічними агентствами» [1, 2]. Ці сполуки (рисунок 1.1) безпосередньо впливають на органи дихання, провокують загострення астми й бронхіту, підвищують ризик серцевих нападів і послаблюють імунну систему.

Рисунок 1.1 – Ситуація в містах [4]

В умовах змін клімату, урбанізації та зростання інтенсивності дорожнього руху питання контролю якості повітря набуває особливої актуальності. Своєчасний та ефективний моніторинг дає змогу виявляти небезпечні рівні забруднення, оцінювати потенційні загрози для здоров’я населення й формувати відповідні заходи захисту.

Однак традиційні методи спостереження за якістю повітря часто є фінансово затратними та малодоступними, особливо для громад із обмеженими ресурсами. У зв’язку з цим зростає інтерес до бюджетних, гнучких систем, які забезпечують локальний моніторинг у режимі реального часу.

«Сучасні сенсори та мікроконтролери забезпечують нові можливості для збору даних, а аналітичне програмне забезпечення дає змогу оцінювати потенційний вплив на здоров’я, що раніше було доступно лише у спеціалізованих лабораторіях чи дослідницьких установах» [3].

Отже, створення системи моніторингу повітря з аналітичними функціями, спрямованими на оцінку впливу на здоров’я людини, є важливим і актуальним напрямом дослідження. Такий підхід об’єднує екологічну інженерію, інформаційні технології та медико-статистичний аналіз, що надає йому високої практичної значущості для суспільства.

1.2 Аналіз існуючих рішень

На сьогодні існує велика кількість рішень для спостереження за якістю повітря, які різняться між собою за рівнем точності, вартісними характеристиками, технічною складністю та можливістю інтеграції з іншими інформаційними системами. Серед них вирізняються класичні державні платформи, зокрема ті, що функціонують у межах Європейської системи звітності про стан повітря (European Air Quality e-Reporting), які забезпечують високу надійність результатів завдяки застосуванню стандартизованого та каліброваного обладнання. Проте значні витрати на обслуговування та технічна складність впровадження таких систем ускладнюють їх застосування на місцевому рівні, особливо в громадах із обмеженими фінансовими ресурсами.

«Комерційні рішення, як-от AirVisual (IQAir), PurpleAir або Clarity Node-S, пропонують доступні за ціною сенсори, які можна легко встановити на локальному рівні. Ці пристрої забезпечують збір даних у реальному часі та передають інформацію через хмарні сервіси» [26].

«Популярність також здобули проєкти з відкритим кодом, як-от Luftdaten (тепер Sensor.Community), що дозволяють користувачам самостійно створювати та підключати сенсори до глобальної мережі» [27].

Попри те, що подібні системи є більш доступними з фінансової точки зору, точність їхніх показників може знижуватись, особливо за несприятливих метеоумов, що зумовлено обмеженнями дешевих сенсорних елементів.

Серед наявних комерційних рішень у сфері моніторингу якості повітря особливу популярність мають сервіси AirVisual (рисунок 1.2) та Plume Labs (рисунок 1.3). Вони поєднують стаціонарні або мобільні датчики з веб- або мобільними інтерфейсами для візуалізації даних та формування рекомендацій. Основою їхньої аналітики є індекс якості повітря (AQI), що дозволяє користувачам оперативно оцінити загальний рівень забруднення повітря у певній місцевості.

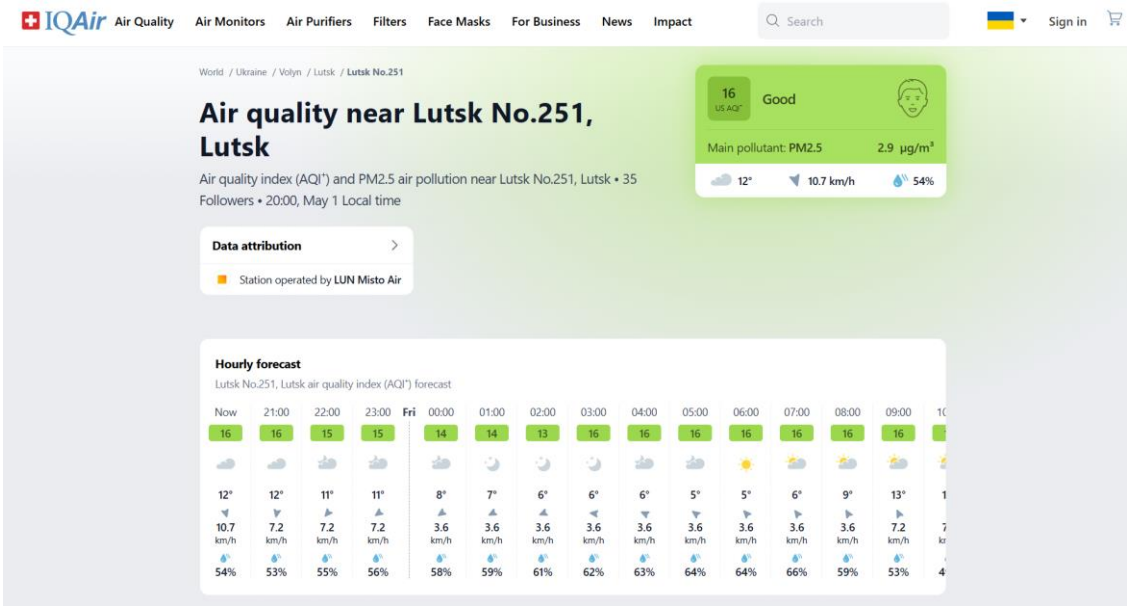

Рисунок 1.2 – AirVisual

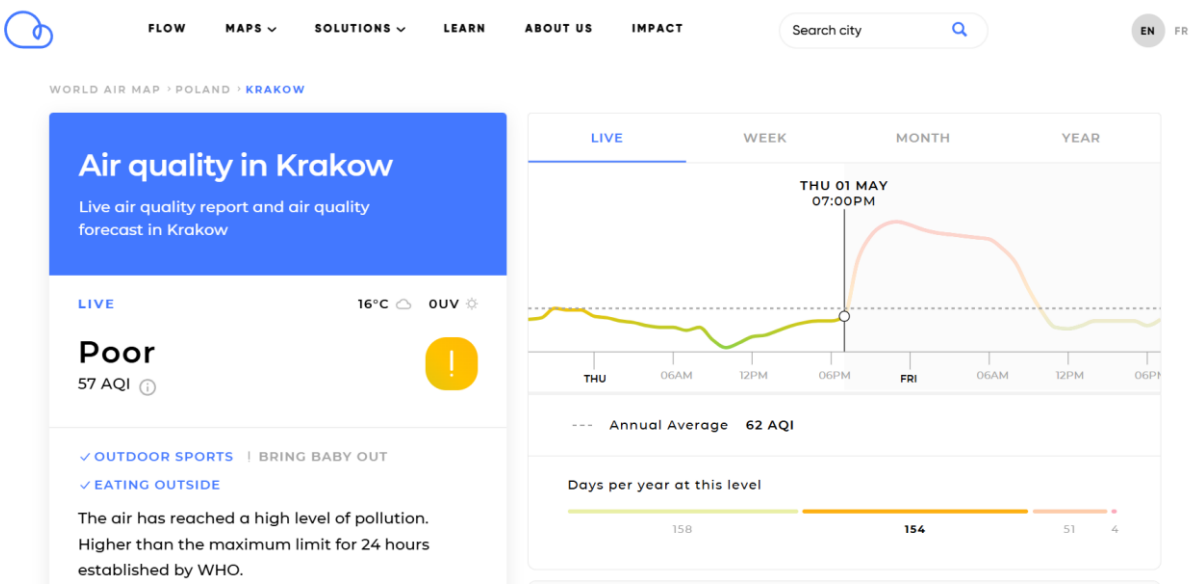

Рисунок 1.3 – Plume Labs

Однак такі платформи, як правило, орієнтуються на узагальнені рекомендації, не враховуючи індивідуальні особливості здоров'я користувачів або їхню чутливість до окремих забруднювачів. Це обмежує ефективність таких систем у контексті персоналізованої профілактики, особливо для вразливих груп населення (людей з хронічними респіраторними захворюваннями, дітей, літніх людей тощо).

Враховуючи зазначене, постає потреба в створені системи яка аналізує оцінку потенційного ризику для здоров'я на основі реальних екологічних параметрів (PM2.5, PM10, температура, вологість), що обробляються за допомогою моделі машинного навчання. Хоча персональні медичні дані наразі не інтегруються, програмна архітектура системи є відкритою до подальшого розширення у напрямку персоніфікованого аналізу. Такий підхід формує основу для створення адаптивних систем попередження, здатних враховувати індивідуальні особливості користувача.

1.3 Постановка задачі

«Погіршення якості атмосферного повітря у поєднанні з поширенням хронічних захворювань дихальної та серцево-судинної системи посилює потребу в

нових підходах до моніторингу довкілля, які мають враховувати не лише екологічні аспекти, а й безпосередній вплив на здоров’я людини» [5]. Більшість сучасних систем моніторингу не орієнтовані на індивідуальні особливості користувачів і вимагають значних фінансових витрат на встановлення та обслуговування, що обмежує їх застосування у малобюджетних громадах.

У цьому контексті метою даної кваліфікаційної роботи є створення доступної за вартістю системи моніторингу якості повітря, яка не лише фіксуватиме екологічні показники в режимі реального часу, але й здійснюватиме первинну оцінку потенційної загрози для здоров’я населення. Запропонована система повинна поєднувати недорогі сенсорні модулі, засоби бездротової передачі даних, аналітичні алгоритми для визначення ризиків і персоналізовані рекомендації, що формуються на основі медико-статистичних моделей.

Для реалізації поставленої мети необхідно виконати такі завдання:

– здійснити огляд наявних сенсорів, здатних вимірювати ключові забруднювальні речовини (PM2.5, PM10), з аналізом їхніх технічних параметрів та точності вимірювань;

– змоделювати систему збору інформації, використовуючи економічні компоненти з можливістю бездротової передачі даних;

– створити програмний модуль для обробки, зберігання та графічного подання зібраної інформації про якість повітря;

– вбудувати до системи алгоритми первинної оцінки впливу зафіксованих рівнів забруднення на стан здоров’я людини;

– реалізувати механізм генерування персоналізованих порад для користувачів, базуючись на поточному стані атмосферного повітря та медичних даних.

У результаті очікується створення інтелектуальної системи, яка виконуватиме функцію інструмента раннього попередження, підвищуватиме рівень екологічної обізнаності населення та сприятиме зниженню ризиків для здоров’я завдяки своєчасному реагуванню на забруднення повітря.

1.4 Теоретичні основи моніторингу якості повітря

Моніторинг стану атмосферного повітря є важливою складовою системи екологічного контролю та дозволяє оцінювати ступінь впливу довкілля на здоров’я населення. Теоретична база цього напряму охоплює вивчення основних забруднювальних компонентів, методів їх вимірювання та аналітичних моделей, що використовуються для оцінки потенційних ризиків для людини.

До основних забруднювальних речовин, що підлягають постійному спостереженню, належать:

– тверді частинки PM2.5 і PM10 – дрібнодисперсні частинки пилу, сажі та диму, які можуть проникати в нижні відділи дихальної системи і навіть у кровоносне русло, створюючи серйозні ризики для здоров’я;

– оксиди азоту (NO та NO_2) – переважно виникають унаслідок експлуатації автомобільного транспорту та викидів промислових підприємств;

– озон (O_3) – вторинний забруднювач, що утворюється в атмосфері в результаті фотохімічних реакцій між іншими забруднюючими речовинами під впливом сонячного світла;

– чадний газ (CO) – безбарвний і без запаху газ, що з’являється в процесі неповного згоряння пального і становить загрозу через здатність блокувати транспорт кисню в організмі;

– діоксид сірки (SO_2) – виникає переважно при спалюванні викопного палива, зокрема вугілля та нафтопродуктів.

«Для інтегральної оцінки рівня забруднення використовується індекс якості повітря (Air Quality Index, AQI). Цей показник розраховується на основі концентрацій кількох основних забруднювачів і подається у вигляді шкали з кольоровим кодуванням: від «добре» (зелений) до «небезпечне» (темно-червоний). AQI дозволяє просто та доступно інформувати населення про рівень небезпеки в конкретний момент часу» [6] (рисунок 1.4).

AQI	Якість повітря
0-50	Чисте
50-100	Прийнятне
100-150	Нездорове для чутливих груп
150-200	Нездорове
200-300	Дуже нездорове
від 300	Небезпечне

Рисунок 1.4 – Індекс якості повітря AQI [6]

Вимірювання показників якості атмосферного повітря можуть здійснюватися різними методами, які відрізняються за точністю, вартістю та практичністю використання:

– гравіметричні методи характеризуються високою точністю результатів, але є ресурсозатратними як у фінансовому, так і в часовому плані;

– сенсорно-аналітичні пристрої забезпечують мобільність і доступність завдяки невисокій вартості, проте поступаються за точністю і часто вимагають періодичного калібрування для коректної роботи;

– лазерні сенсори, зокрема моделі на кшталт SDS011 або Plantower, використовують метод розсіювання світла для визначення концентрації зважених частинок у повітрі, що дозволяє проводити вимірювання у режимі реального часу.

Оцінка шкідливого впливу забрудненого повітря на здоров’я людини ґрунтується на епідеміологічних моделях, що враховують комплекс факторів, зокрема:

– тривалість впливу шкідливих речовин;

– концентрацію забруднюючих компонентів у повітрі;

– індивідуальні характеристики людини, такі як вік, стать та наявність хронічних захворювань;

– сезонні коливання та метеорологічні умови.

«Одним із таких інструментів є AirQ+ – програмний продукт, розроблений ВООЗ для розрахунку здоров'я, пов'язаного з впливом забруднення» [4]. «Ще один приклад – BenMAP (Environmental Benefits Mapping and Analysis Program), який розроблений Агентством з охорони довкілля США (EPA) і дає змогу оцінити економічні та медичні наслідки впливу забруднення повітря» [10] (рисунок 1.5).

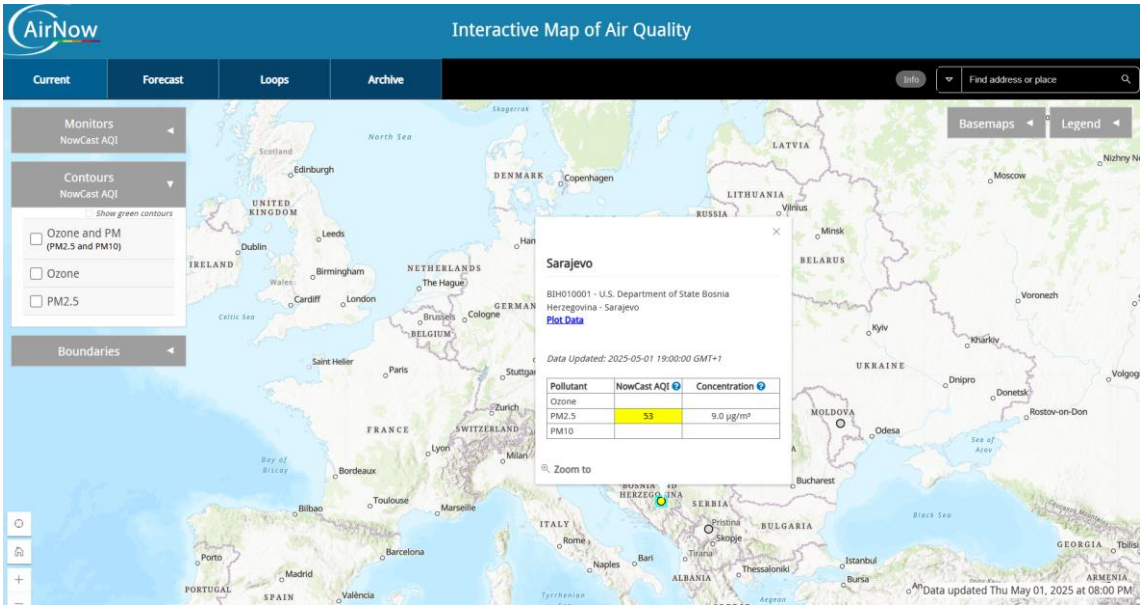

Рисунок 1.5 – BenMAP [3]

Сучасні системи моніторингу якості повітря дедалі частіше інтегрують екологічні дані з індивідуальними характеристиками користувачів для формування персоналізованих оцінок ризику. Завдяки цьому можливо надавати не лише загальні рекомендації, а й цільові поради для осіб із підвищеною чутливістю – зокрема, для людей, які мають астму або захворювання серцево-судинної системи. Такий підхід сприяє впровадженню інтелектуальних систем підтримки прийняття рішень у сфері охорони громадського здоров'я.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ ЗБОРУ ІНФОРМАЦІЇ

2.1 Підбір складових системи збору інформації

На підставі вивчення технічних вимог до системи контролю якості повітря було сформовано перелік сенсорів, які забезпечують вимірювання ключових показників стану атмосферного повітря, а також супутніх метеорологічних величин. До основних сенсорів належать:

– MQ-135 (рисунок 2.1) – це газовий сенсор, здатний виявляти кілька типів забруднюючих речовин, зокрема аміак (NH_3), вуглекислий газ (CO_2), бензольні пари та оксиди азоту (NO_x).;

– PMS5003 (рисунок 2.2) – лазерний датчик, призначений для виявлення та вимірювання концентрації твердих частинок пилу у повітрі, зокрема фракцій PM2.5 і PM10;

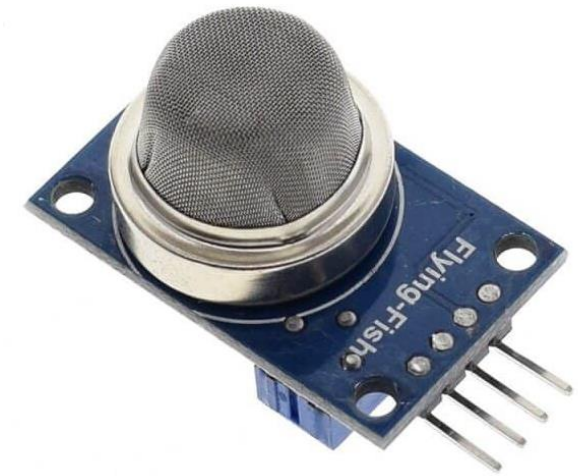

Рисунок 2.1 – Сенсор MQ-135 [15]

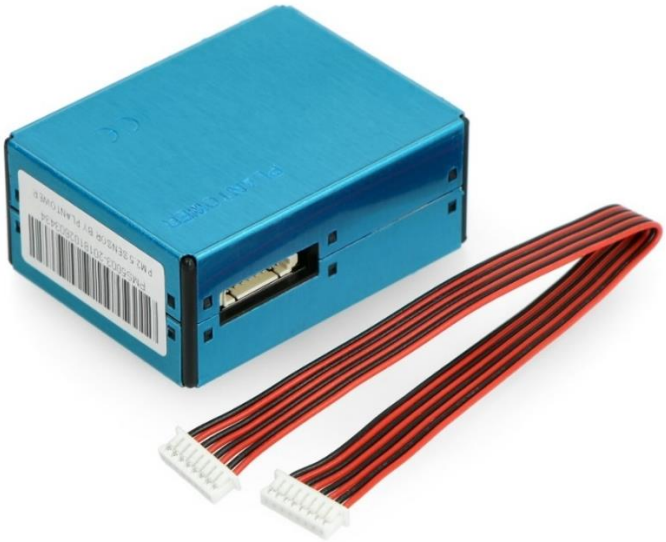

Рисунок 2.2 – Датчик PMS5003 [20]

– «DHT22 – цифровий датчик для вимірювання температури та відносної вологостію» [10] (рисунок 2.3).

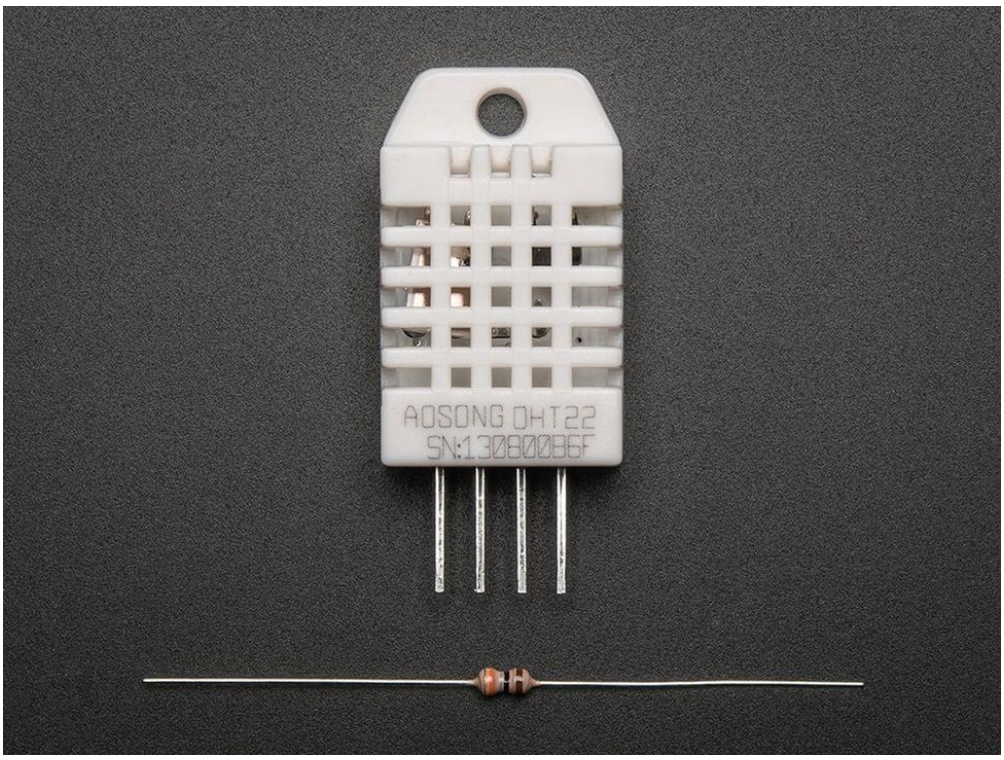

Рисунок 2.3 – Цифровий датчик DHT22 [8]

Сенсори під’єднуються до мікроконтролера ESP32 (рисунок 2.4), який виконує роль основного вузла для збору та надсилання даних. Вибір ESP32 зумовлений його вбудованою підтримкою Wi-Fi, високою продуктивністю та популярністю в проєктах Інтернету речей (IoT).

Рисунок 2.4 – Мікроконтролер ESP32 [11]

2.2 Розробка та тестування моделі в середовищі Wokwi

Для тестування функціональності системи збору даних було створено апаратну модель у середовищі Wokwi (рисунок 2.5) – онлайн-симуляторі електронних схем. У цьому середовищі було змодельовано взаємодію мікроконтролера ESP32 із сенсорами MQ-135, DHT22 та OLED-дисплеєм.

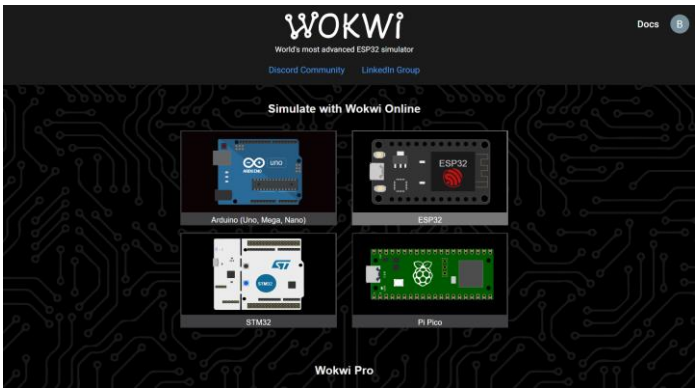

Рисунок 2.5 – Середовище Wokwi

За допомогою симуляції вдалося перевірити:

– коректність з’єднання компонентів (включаючи GPIO та живлення);

– здатність системи зчитувати дані з датчиків у режимі реального часу;

– відображення показників температури, вологості та рівня забруднення на OLED-дисплеї;

– реакцію системи на зміну вхідних даних;

– інтеграцію керування виконавчими пристроями за допомогою кнопок і реле.

Wokwi надав можливість проводити тестування без необхідності у фізичних компонентах, що значно прискорило процес розробки та налагодження.

Для підтвердження працездатності запропонованої системи у середовищі Wokwi було створено повноцінну інтерактивну симуляційну модель (рисунок 2.6), яка точно відтворює ключові елементи апаратної частини. У складі макета – мікроконтролер ESP32, цифровий датчик DHT22, газовий сенсор MQ-135, OLED-дисплей, кнопки управління та релейні модулі. Ця модель забезпечує не лише

зчитування, обробку та виведення даних, а й дозволяє змоделювати поведінку системи у відповідь на зміни в навколишньому середовищі.

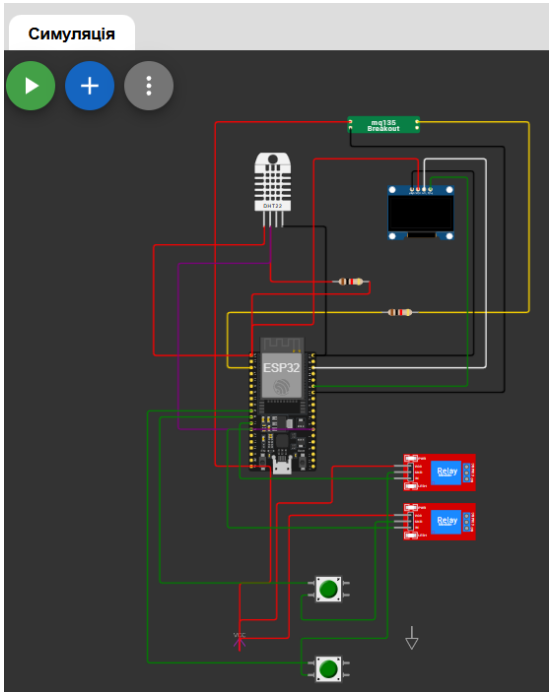

Рисунок 2.6 – Модель у Wokwi [30]

Цей симулятор надає можливість наочно дослідити алгоритм роботи системи, перевірити правильність взаємодії між програмними та апаратними компонентами, а також простежити зміну показників із сенсорів у динаміці. Окрім цього, він дозволяє провести повноцінну перевірку програмної логіки без потреби у фізичному обладнанні. Таким чином, модель, реалізована у середовищі Wokwi, відіграє ключову роль у процесі тестування та поетапного вдосконалення проєкту, а також є ефективним засобом демонстрації його функціональних можливостей під час захисту кваліфікаційної роботи.

2.3 Підбір технологій для програмної частини систми збору даних

Програмне забезпечення системи моніторингу якості повітря включає два ключові рівні: вбудований рівень, що працює на мікроконтролері ESP32, та серверно-клієнтський рівень, який відповідає за отримання, обробку, збереження й візуалізацію зібраної інформації.

Для програмування мікроконтролера ESP32 було застосовано Arduino-сумісний підхід із використанням мови C++ у середовищі Wokwi. Програмна реалізація включає такі основні компоненти (рисунок 2.7):

– бібліотека DHT.h – забезпечує зчитування значень температури та вологості з сенсора DHT22;

– бібліотеки Adafruit_GFX.h та Adafruit_SSD1306.h – відповідають за виведення інформації на OLED-дисплей;

– Wire.h – використовується для реалізації I²C-зв'язку з дисплеєм;

– функція analogRead() – дозволяє зчитувати аналоговий сигнал із газового сенсора MQ-135 через пін GPIO 34.

Прошивка мікроконтролера реалізує зчитування даних із сенсорів, умовне обчислення концентрації CO_2 в одиницях ppm, відображення результатів на OLED-дисплеї, а також діагностичний вивід через серійний порт. Програмний код передбачає обробку можливих помилок зчитування (зокрема, за допомогою перевірки функцією isnan()), виконує ініціалізацію дисплея та забезпечує періодичне оновлення даних з інтервалом у 3 секунди.

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

#define DHTPIN 4
#define DHTTYPE DHT22
#define MQ135_PIN 34  // аналоговий пін

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
```

Рисунок 2.7 – Основні компоненти

У разі повної апаратної реалізації проєкту (поза середовищем Wokwi) передбачалося використання таких технологій:

– «Python + Flask – веб-фреймворк для створення REST API, через яке дані від ESP32 передаються на сервер для подальшого оброблення» [29] (рисунок 2.8);

Рисунок 2.8 – Python + Flask [9]

– «PostgreSQL – високопродуктивна реляційна СУБД, придатна для зберігання великого обсягу історичних даних» [22] (рисунок 2.9);

Рисунок 2.9 – PostgreSQL [23]

– «MQTT – протокол обміну повідомленнями, який дозволяє передавати дані з ESP32 у режимі реального часу через брокер HiveMQ » [16] (рисунок 2.10);

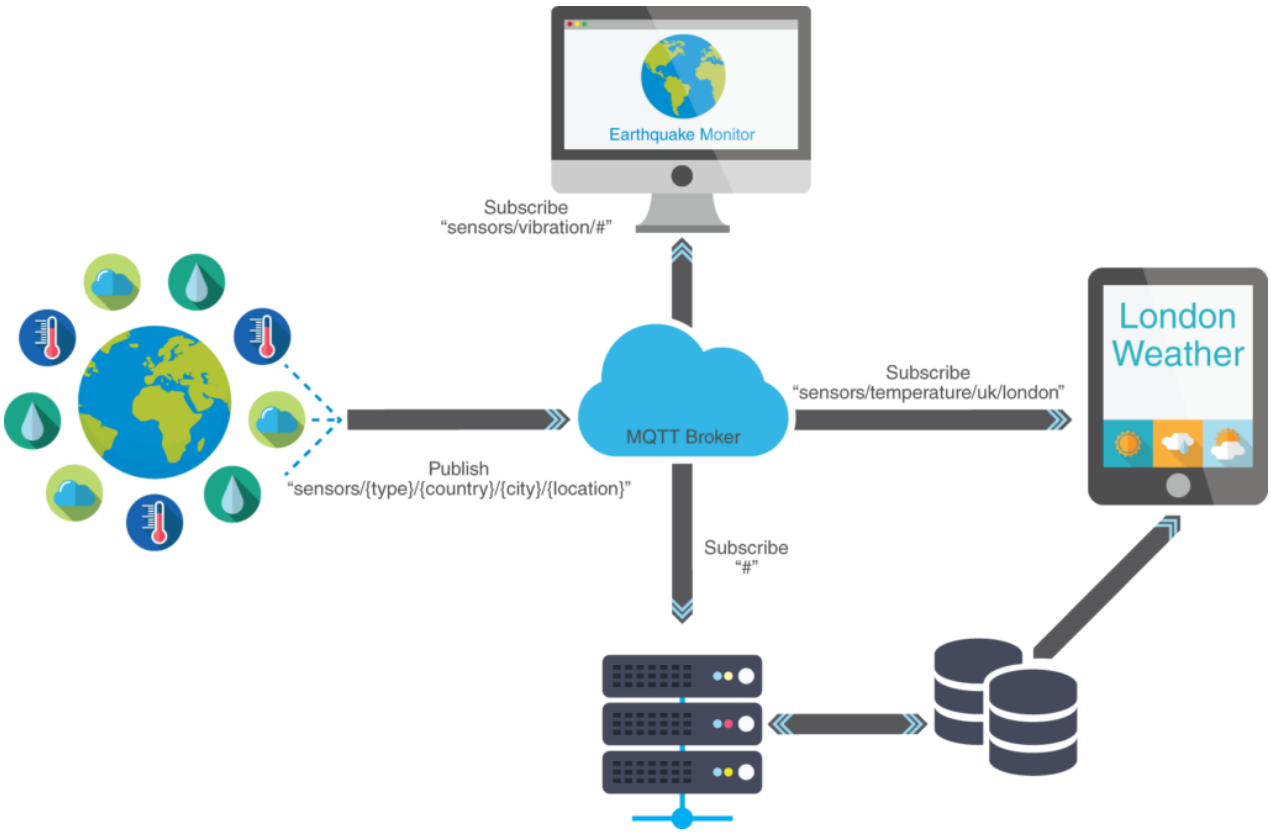

Рисунок 2.10 – MQTT [17]

– «React.js + Leaflet.js – клієнтський інтерфейс, що дозволяє будувати адаптивні веб-сторінки з інтерактивними картами, на яких відображаються поточні значення якості повітря у різних точках» [14] (рисунок 2.11).

Рисунок 2.11 – React.js + Leaflet.js [18]

Етапи взаємодії компонентів:

– ESP32 зчитує сенсорні дані та надсилає їх через MQTT або HTTP;

– Flask-сервер приймає запити та записує дані у базу;

– PostgreSQL зберігає записи та дозволяє виконувати аналітичні запити;

– React-клієнт отримує дані через API та візуалізує їх для користувача.

Технології які були використані зображені на рисунку 2.12.

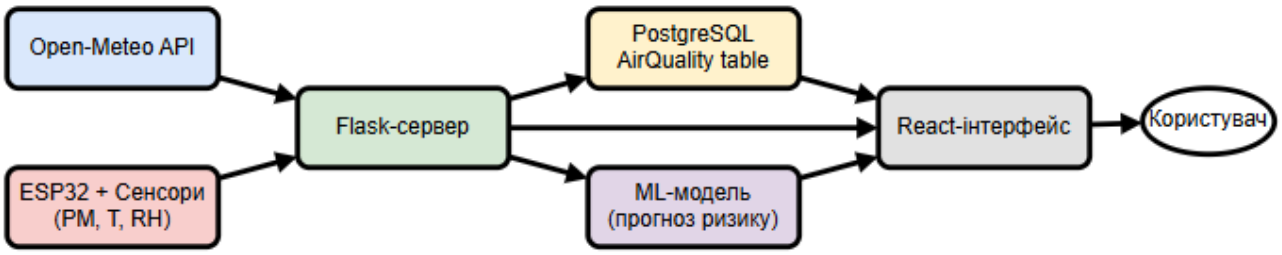

Рисунок 2.12 – Технології розробки програмного забезпечення

Варто зазначити, що під час тестування системи в середовищі Wokwi підключення до серверної частини не здійснювалося. Було зосереджено увагу на реалізації локальної логіки: зчитування значень сенсорів, їх обробка та виведення на OLED-дисплей. Передача даних через MQTT або HTTP, а також збереження в базі даних PostgreSQL та відображення на фронтенді залишаються як потенційні етапи для подальшого розширення системи в умовах реального впровадження.

2.4 Аналіз обраних рішень

У ході розробки системи моніторингу якості повітря було обрано набір апаратних і програмних засобів, що забезпечують ефективну, доступну та гнучку реалізацію поставлених задач. Підбір компонентів здійснювався з урахуванням балансу між точністю вимірювань, простотою реалізації, вартістю та потенціалом для подальшого масштабування.

Основу апаратної частини становить мікроконтролер ESP32, який поєднує високу обчислювальну потужність, підтримку бездротових з’єднань та сумісність з великою кількістю сенсорів. Для збору екологічних даних використано сенсори MQ-135 (оцінка якості повітря) і DHT22 (температура й вологість), а для

локального відображення – OLED-дисплей, що забезпечує зручний вивід інформації для користувача.

Як основне середовище для створення прототипу було обрано Wokwi – онлайн-симулятор електроніки, який дозволив зібрати та протестувати логіку роботи системи без фізичного обладнання. Інтерактивний інтерфейс Wokwi забезпечив можливість у реальному часі змінювати вхідні дані сенсорів, спостерігати реакцію системи, налагоджувати код та перевіряти функціонування дисплея, кнопок і реле. Це суттєво скоротило час розробки й знизило ризик технічних помилок.

Програмну частину реалізовано мовою C++ з використанням Arduino-сумісних бібліотек. У прошивці передбачено зчитування та базову обробку даних, виведення показників на дисплей, а також діагностичну інформацію через серійний порт. Хоча на етапі симуляції не реалізовано підключення до серверної частини, структура коду дозволяє легко інтегрувати функції надсилання даних – наприклад, через MQTT або HTTP.

Проєкт має високу гнучкість: при наявності фізичних модулів розроблену систему можна швидко перенести з віртуального середовища у реальне. Усі обрані компоненти є доступними, з відкритою документацією та підтримкою спільноти, що робить рішення придатним для подальшого розвитку, масштабування або використання в освітніх цілях.

У підсумку, обрана архітектура цілком відповідає завданням кваліфікаційної роботи на етапі симуляції та створює надійну основу для майбутньої реалізації системи в реальних умовах.

2.5 Принцип функціонування апаратної частини

Апаратна частина системи моніторингу відповідає за циклічне зчитування екологічних показників, їх попередню обробку та оперативне відображення результатів користувачу. Центральним елементом є мікроконтролер ESP32, до якого підключено як цифрові (DHT22), так і аналогові (MQ-135) сенсори, OLED-дисплей, кнопки керування та релейні модулі.

Послідовність роботи системи:

– після подачі живлення ESP32 ініціалізує OLED-дисплей, сенсор DHT22 та готується до зчитування аналогового сигналу з MQ-135, встановлюється серійне з’єднання для діагностики;

– кожні 3 секунди мікроконтролер виконує зчитування температури та вологості з DHT22, а також рівня забруднення повітря з MQ-135 (в умовних одиницях, що умовно відповідають ppm);

– отримані значення температури та вологості напряму передаються на дисплей, аналоговий сигнал з MQ-135 обробляється й конвертується в умовний числовий показник за спрощеною формулою оцінки рівня забруднення;

– дані виводяться на OLED-дисплей у зручному для користувача форматі (рисунок 2.13);

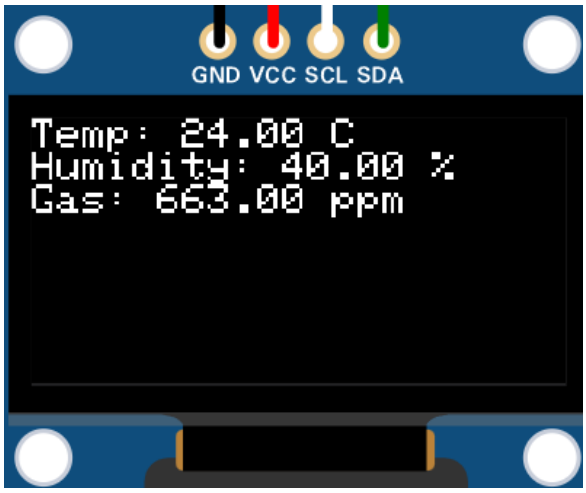

Рисунок 2.13 – Вивід даних

– за допомогою кнопок, підключених до ESP32 (рисунок 2.14), користувач може змінювати режими відображення або вмикати/вимикати реле, наприклад для керування вентиляцією або сигналізацією;

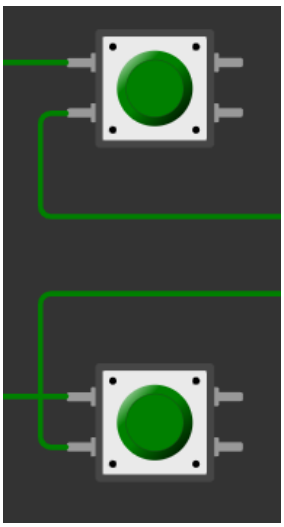

Рисунок 2.14 – Кнопки, підключені до ESP32

– усі значення дублюються в серійному моніторі (рисунок 2.15), що дозволяє вести журнал подій і спрощує процес налагодження під час розробки чи тестування.

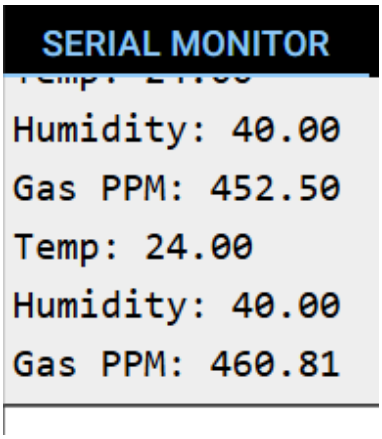

Рисунок 2.15 – Серійний монітор

2.6 Можливість модульного розширення апаратної частини

Однією з ключових переваг розробленої системи є її модульна архітектура, яка дозволяє легко додавати нові компоненти без необхідності повної реконструкції схеми або програмної логіки. Це забезпечує високу гнучкість і адаптивність системи до змін у вимогах, середовищі експлуатації або цільовому застосуванні.

Мікроконтролер ESP32, що використовується у проєкті, має значний запас обчислювальних ресурсів та велику кількість вільних GPIO-входів/виходів.

Це відкриває можливості для підключення:

– додаткових сенсорів: GPS-модуля (рисунок 2.16), датчика тиску BMP280, ультразвукових датчиків для оцінки рівня забруднення поверхонь;

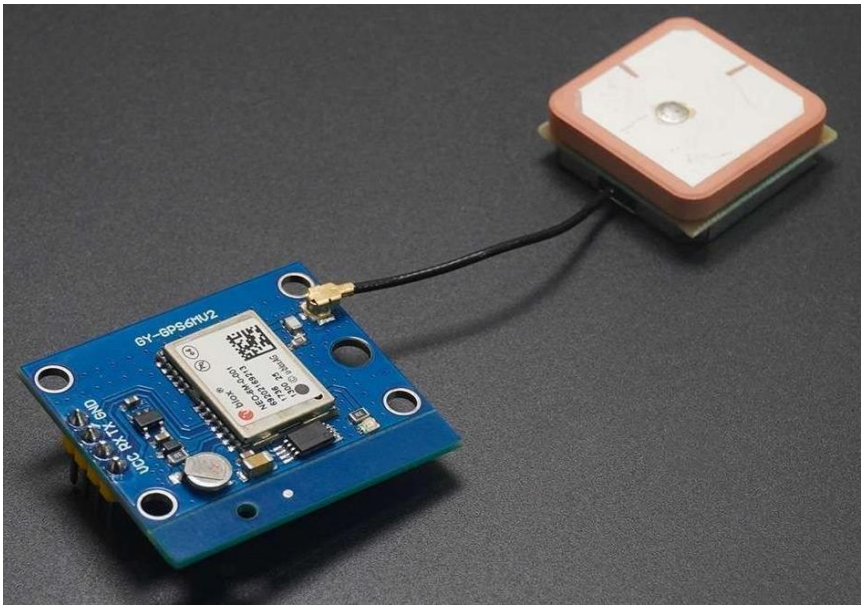

Рисунок 2.16 – GPS-модуль [13]

– сенсорів шуму, освітленості, ультрафіолету, пилу (рисунок 2.17), які можуть бути використані для комплексного моніторингу довкілля;

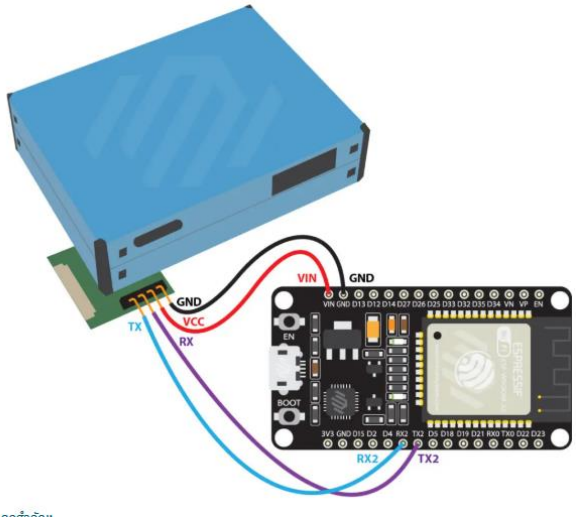

Рисунок 2.17 – Схема підєднання PMS7003 до мікроконтроллера [21]

– виконавчих пристроїв, таких як: вентилятори, клапани, сирени, світлові індикатори, які можна керувати через реле або транзистори;

– інтерфейсів взаємодії з користувачем – наприклад, матричних клавіатур, кнопкових панелей, додаткових OLED/LCD-дисплеїв;

– радіомодулів для бездротового зв’язку (LoRa, GSM/3G, Zigbee), що дозволяє розгорнути систему в місцях без Wi-Fi.

Завдяки підтримці I²C, SPI, UART, PWM, ADC і DAC інтерфейсів, ESP32 здатен паралельно обробляти велику кількість підключених модулів. Розширення системи не потребує зміни основної архітектури – лише внесення змін до прошивки, яка вже структурована відповідно до модульного підходу (функції, таймери, події).

Також варто відзначити, що всі компоненти, обрані для моделі, мають відкриту документацію, активну спільноту підтримки, сумісність з Arduino-середовищем і численні готові бібліотеки. Це спрощує інтеграцію навіть нестандартних модулів.

Таким чином, створена система моніторингу забруднення повітря може служити базою для побудови більш складних екологічних платформ, мобільних станцій або систем автоматичного реагування. Її архітектура відповідає принципам гнучкої IoT-інженерії та може адаптуватися до майбутніх завдань без значних витрат.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Загальна структура програмного забезпечення

У процесі розробки системи моніторингу якості повітря основний акцент було зроблено на створенні програмного забезпечення, яке охоплює повний цикл обробки даних: від їх отримання – до збереження, аналізу та візуального подання користувачеві. Система забезпечує збір інформації, її аналітичну обробку, збереження у базі даних, оцінку потенційних ризиків для здоров'я, а також представлення результатів у зручному веб-інтерфейсі. Функціональна схема зображена на рисунку 3.1.

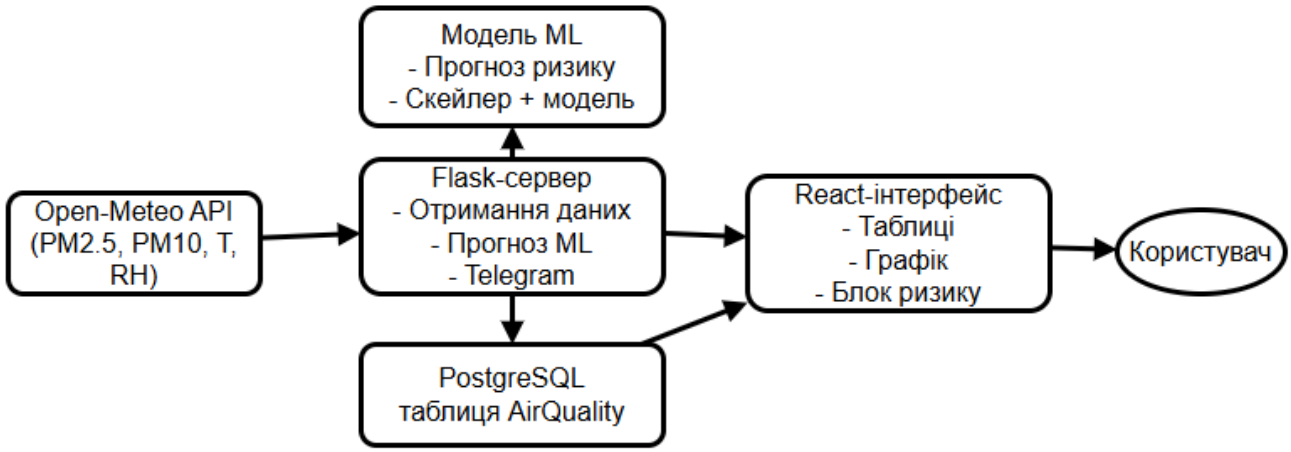

Рисунок 3.1 – Функціональна схема

Відмінною рисою запропонованого рішення є відхід від традиційної IoT-моделі, яка базується на фізичних сенсорах, підключених до мікроконтролера. Замість цього в системі реалізовано інтеграцію з відкритим метеорологічним API – Open-Meteo (рисунок 3.2), що надає актуальні дані про стан повітря, температуру та вологість у режимі реального часу. Такий підхід дозволив спростити апаратну частину, зберігаючи при цьому інформативність та функціональність системи.

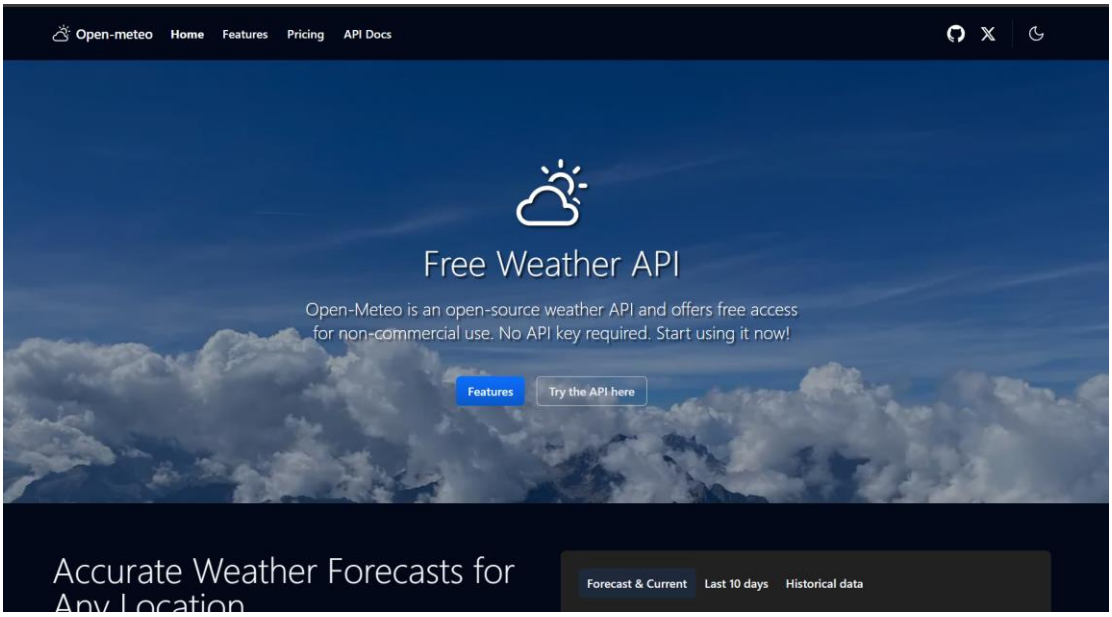

Рисунок 3.2 – Open-Meteo [12]

Запропонований підхід має низку переваг, зокрема:

– зниження витрат завдяки відмові від фізичних сенсорів;

– доступ до стабільних та масштабованих даних з відкритого джерела;

– можливість географічної фільтрації – у даному проєкті реалізовано отримання даних для м. Луцьк;

– швидке розгортання прототипу аналітичної системи, що значно спрощує процес розробки та тестування.

Архітектура програмного забезпечення (рисунок 3.3) поділена на дві частини:

– бекенд, реалізований за допомогою Flask – відповідає за обробку запитів, взаємодію з API Open-Meteo, збереження даних у базі та реалізацію логіки аналізу впливу на здоров’я;

– фронтенд, побудований на основі React.js – забезпечує користувацький інтерфейс для перегляду та аналізу даних;

– обмін між цими частинами здійснюється через HTTP-запити до REST API, що дозволяє гнучко масштабувати або змінювати окремі компоненти системи в майбутньому.

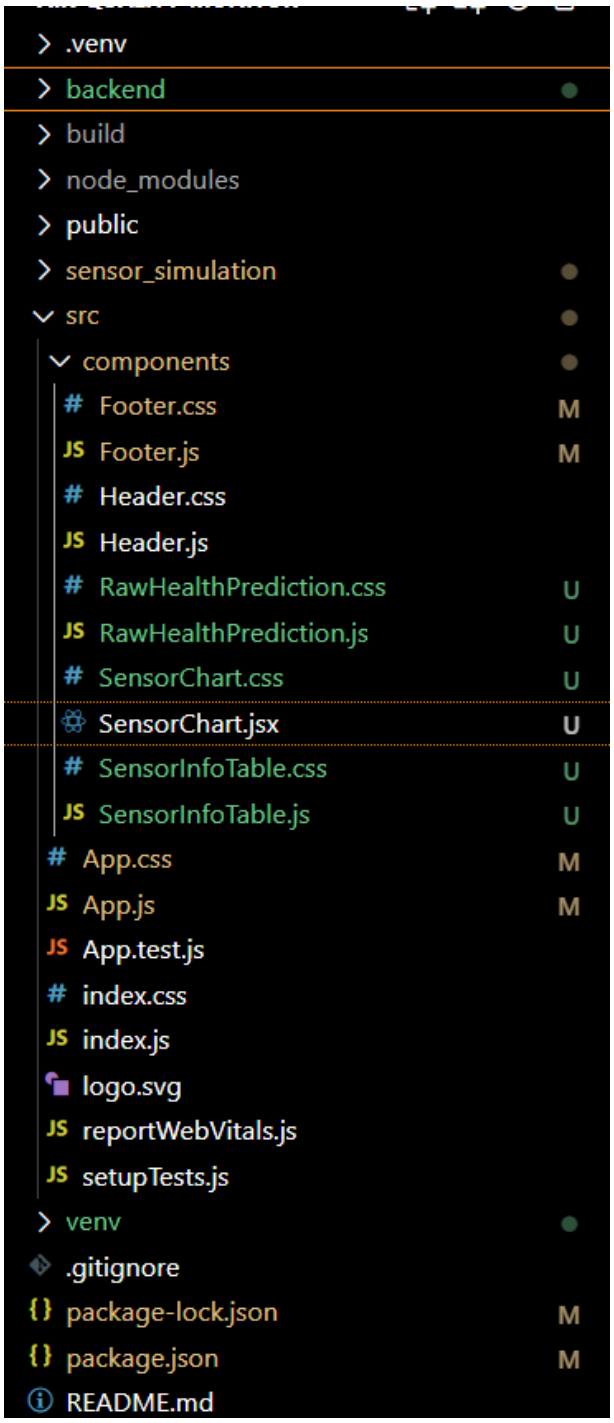

Рисунок 3.3 – Архітектура

3.2 Серверна частина: Python + Flask + PostgreSQL

Серверна частина системи реалізована з використанням мови Python 3.11 та мікрофреймворку Flask (додаток Б). Структура проєкту включає наступні модулі:

– server.py (рисунок 3.4) – головний серверний скрипт, що містить усі маршрути API;

```python
from flask import Flask, jsonify, Response, request
from flask_sqlalchemy import SQLAlchemy
from flask_cors import CORS
from apscheduler.schedulers.background import BackgroundScheduler
import requests
import logging
from datetime import datetime
import pytz
from io import StringIO
import csv
import joblib
import numpy as np
from notify import send_alert  # 🆕 Telegram alerts

app = Flask(__name__)
CORS(app)

app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:555@localhost/air_quality'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

class AirQuality(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    timestamp = db.Column(db.DateTime, nullable=False)
    address = db.Column(db.String(120), nullable=False)
    sensor_type = db.Column(db.String(50), nullable=False)
    value = db.Column(db.Float, nullable=True)
    unit = db.Column(db.String(20), nullable=True)

    def to_dict(self):
        return {
            'timestamp': self.timestamp.strftime("%Y-%m-%d %H:%M:%S"),
            'address': self.address,
            'sensor_type': self.sensor_type,
            'value': self.value,
            'unit': self.unit
```

Рисунок 3.4 – Фрагмент коду server.py

– health_risk_model.pkl – збережена модель машинного навчання;

– health_risk_scaler.pkl – скейлер для попередньої обробки вхідних даних;

– notify.py – (рисунок 3.5) модуль Telegram-сповіщень;

– база даних PostgreSQL через SQLAlchemy (рисунок 3.6).

```python
import requests

# 🔐 Дані твого Telegram бота
TELEGRAM_TOKEN = "8196897420:AAEGovM0SxLYZCWMutg81TSjW4b_5vJWV3g"
CHAT_ID = "885623897"

def send_alert(message):
    url = f"https://api.telegram.org/bot{TELEGRAM_TOKEN}/sendMessage"
    payload = {
        "chat_id": CHAT_ID,
        "text": message,
        "parse_mode": "HTML"
    }

    try:
        response = requests.post(url, json=payload)
        if response.status_code == 200:
            print("✅ Сповіщення надіслано!")
        else:
            print(f"⚠️ Помилка надсилання: {response.text}")
    except Exception as e:
        print(f"❌ Виняток під час надсилання: {e}")
```

Рисунок 3.5 – Компонент для оповіщення в телеграмі notify.py

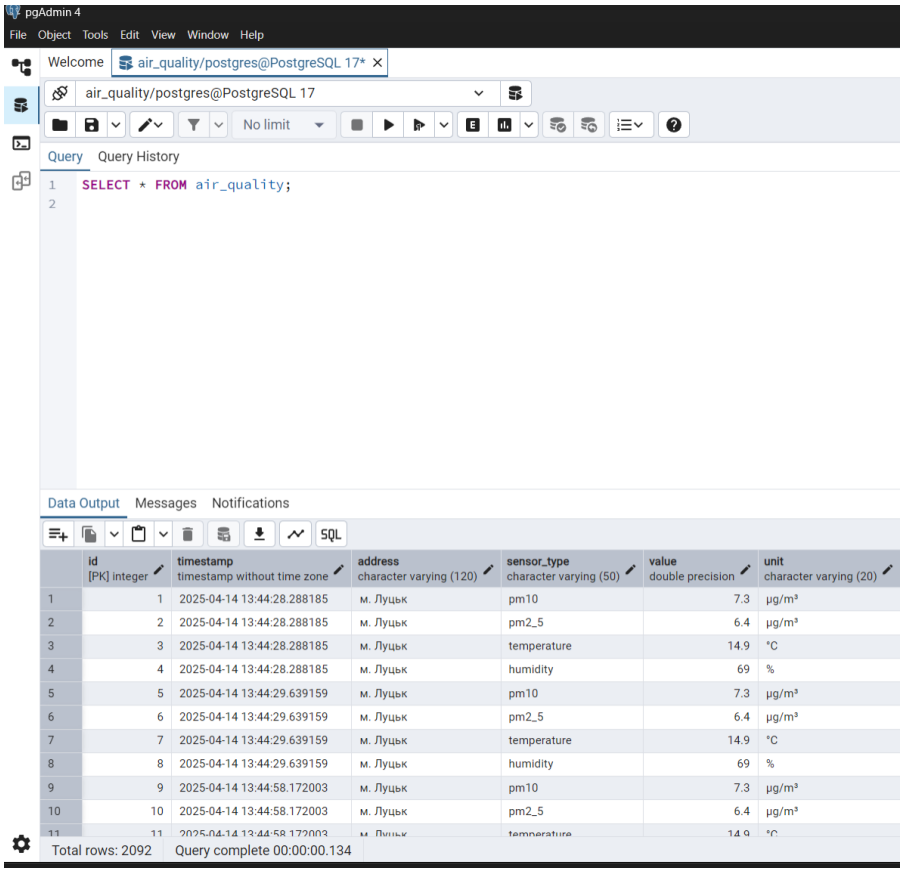

Рисунок 3.6 – База даних PostgreSQL

Інформація про якість повітря (PM2.5, PM10), температуру та вологість автоматично отримується з API Open-Meteo з інтервалом у 30 секунд. Для реалізації цієї функціональності у бекенді використовується бібліотека requests для надсилання HTTP-запитів (рисунок 3.7), а періодичне виконання запланованих задач забезпечується за допомогою APScheduler. Такий підхід дозволяє організувати безперервний збір даних у фоновому режимі без потреби у фізичних сенсорах.

```
try:
    aq_response = requests.get(
        "https://air-quality-api.open-meteo.com/v1/air-quality",
        params={
            "latitude": 50.7472,
            "longitude": 25.3254,
            "current": "pm10,pm2_5"
        }
    )
    aq_response.raise_for_status()
    aq_data = aq_response.json().get("current", {})
```

Рисунок 3.7 – Використання requests

Після отримання JSON-відповіді з Open-Meteo дані проходять етап об’єднання та нормалізації, після чого зберігаються у базі даних PostgreSQL. Кожен запис у базі містить такі атрибути:

– мітку часу, з урахуванням часового поясу Europe/Kyiv;

– тип параметра (наприклад, PM10, PM2.5, температура, вологість);

– значення вимірювання;

– одиницю виміру;

– географічну прив’язку – у цьому випадку м. Луцьк.

Збережені дані слугують не лише для архівування, а й активно використовуються у подальшому для аналітичної обробки, візуалізації трендів та

побудови графіків у веб-інтерфейсі, що забезпечує користувачеві наочне уявлення про зміни якості повітря в динаміці.

3.3 Інтеграція моделі інформування про ризик для здоров'я

Особливістю проєкту є аналіз впливу показників якості повітря на здоров'я людини. Для цього була використана модель машинного навчання, попередньо натренована у середовищі Python (наприклад, з використанням бібліотеки sklearn).

Модель класифікує стан якості повітря за п'ятибальною шкалою (рисунок 3.8):

– низький ризик;

– помірний;

– підвищений;

– високий;

– критичний.

```
const healthImpactMessages = {
  1: "✅ Низький ризик: Повітря чисте, ризик мінімальний.",
  2: "⚠️ Помірний ризик: Легкий вплив на чутливі групи людей.",
  3: "🚰 Високий ризик: Рекомендується обмежити активність на відкритому повітрі.",
  4: "🔴 Дуже високий ризик: Може впливати на всіх. Уникайте активностей на вулиці.",
  5: "⚠️ 🚨 Небезпечний рівень: Високий ризик для здоров'я. Потрібні термінові заходи!"
};
```

Рисунок 3.8 – Класифікація за п'ятибальною шкалою

Для забезпечення точної класифікації вхідні дані (PM2.5, PM10, температура, вологість) перед передачею в модель проходять масштабування за допомогою скейлера.

При кожному новому запиті з інтерфейсу модель повертає рівень ризику, який відображається у відповідному кольорі (зелений, жовтий, червоний). Якщо рівень ≥4 – сервер автоматично викликає функцію send_alert() і надсилає повідомлення через Telegram.

Таким чином, система виконує роль інструмента раннього попередження, який може бути розширено з урахуванням персональних медичних даних.

3.4 Користувацький інтерфейс: React.js + Recharts

Фронтенд-система реалізована з використанням бібліотеки React.js. Вона виконує функції:

– регулярного запиту до /api/data кожні 30 секунд;

– візуалізації останніх та історичних значень сенсорних параметрів (рисунок 3.9);

Дані сенсорів

Час	Адреса	Температура (°C)	Вологість (%)	PM2.5 (μg/m³)	PM10 (μg/m³)
2025-05-16 14:57:16	м. Луцьк	12.8	46	4.4	5
2025-05-16 14:57:06	м. Луцьк	12.8	46	4.4	5
2025-05-16 14:56:46	м. Луцьк	12.8	46	4.4	5
2025-05-16 14:56:36	м. Луцьк	12.8	46	4.4	5
2025-05-16 14:56:16	м. Луцьк	12.8	46	4.4	5

Рисунок 3.9 – Візуалізація значень

– відображення ризику для здоров’я (рисунок 3.10);

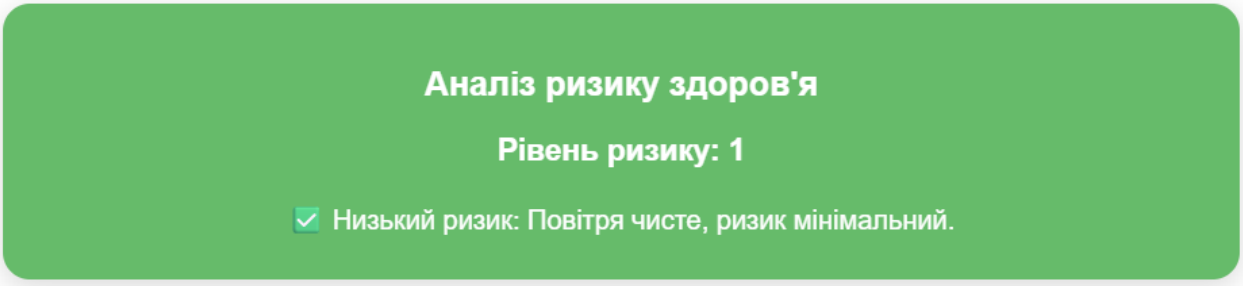

Рисунок 3.10 – Ризик здоров’я

– генерації графіків за останній тиждень (рисунок 3.11);

– підтримки темної теми оформлення;

Рисунок 3.11 – Графік зміни параметрів

– експорту CSV-файлів.

У фронтенд-частині проєкту реалізовано зручний та інформативний інтерфейс для відображення даних моніторингу. Основні компоненти:

– SensorChart – використовує бібліотеку Recharts для побудови лінійних графіків, які наочно демонструють динаміку змін параметрів якості повітря;

– SensorInfoTable – виводить дані у табличному форматі, забезпечуючи швидкий перегляд значень за добовими інтервалами.

Дані попередньо групуються за типами параметрів (PM2.5, PM10, температура, вологість) та агрегуються з розбиттям по днях. Для форматування та локалізації дати використовується бібліотека date-fns з підтримкою української мови (uk), що дозволяє підписувати вісь часу назвами днів тижня (наприклад: Пн, Вт, Ср).

Додаткові особливості:

– адаптивний дизайн – інтерфейс коректно відображається на різних пристроях (ПК, планшети, смартфони);

– комунікація з бекендом – здійснюється за допомогою бібліотеки axios (рисунок 3.12), яка забезпечує зручний обмін даними через HTTP-запити.

Загалом, інтерфейс орієнтований на користувача та адаптований для перегляду як оперативної інформації, так і історичних даних у зручному візуальному форматі.

```
const response = await axios.post("http://127.0.0.1:5000/predict_health", payload);
```

Рисунок 3.12 – Використання бібліотеки axios

3.5 Безпека, масштабування та подальший розвиток

На поточному етапі система функціонує на локальному сервері, однак її архітектура побудована з урахуванням майбутнього масштабування до хмарного середовища. API легко розгортається на таких платформах як Heroku (рисунок 3.13) або Railway (рисунок 3.14), а клієнтську частину можна розмістити на Vercel чи Netlify, що забезпечує швидкий перехід до продуктивного розгортання без суттєвих змін у коді.

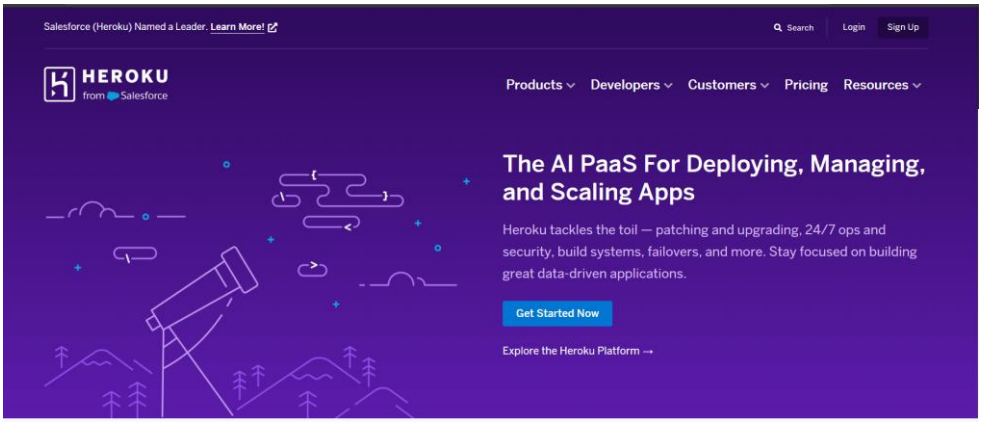

Рисунок 3.13 – Платформа Heroku [28]

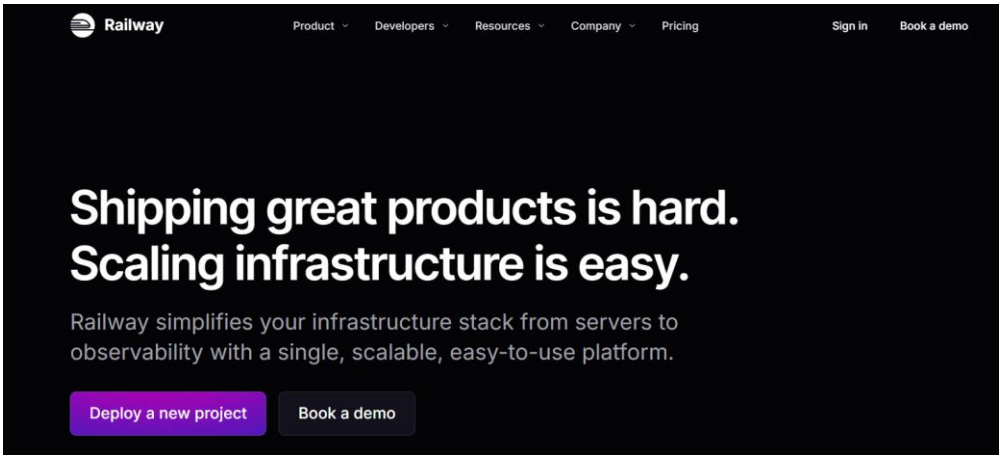

Рисунок 3.14 – Платформа Railway [25]

Заходи безпеки, передбачені в архітектурі:

– обмеження доступу до API за допомогою токенів авторизації;

– логування HTTP-запитів для відстеження активності та налагодження;

– моніторинг працездатності сервера, що дозволяє оперативно реагувати на збої.

Можливі напрями подальшого розвитку:

– інтеграція геолокації користувача, що дозволить автоматично визначати місцезнаходження та підбирати відповідні регіональні дані про якість повітря;

– підключення мобільного додатку, який може слугувати альтернативним каналом доступу до системи та отримання сповіщень;

– індивідуальні користувацькі профілі з можливістю налаштування особистих порогів ризику, наприклад, для людей із хронічними захворюваннями;

– Push-сповіщення через Web Push API або Firebase Cloud Messaging – для своєчасного інформування про критичні зміни стану повітря.

Усі ці елементи створюють передумови для розвитку системи в повноцінну платформу екологічного моніторингу з персоналізованими функціями та високою масштабованістю.

3.6 Тестування програмного забезпечення

Один із важливих етапів розробки інформаційної системи – тестування, яке дає змогу перевірити коректність роботи окремих модулів, загальну стабільність системи та її відповідність очікуваній логіці взаємодії з користувачем.

У межах проєкту з моніторингу якості повітря тестування охоплювало серверну частину, базу даних і клієнтський інтерфейс. Зважаючи на те, що джерелом даних виступає зовнішній API – Open-Meteo, особливу увагу було зосереджено на наступних питаннях:

– коректність отримання та парсингу API-відповідей;

– обробка потенційних помилок (наприклад, відсутність підключення, недоступність ресурсу, порожні або некоректні відповіді);

– стійка робота фонових задач, які періодично звертаються до API й не повинні викликати збоїв у роботі сервера.

Таке тестування дозволило виявити та усунути критичні помилки ще на етапі розробки, що суттєво підвищує надійність і готовність системи до розгортання у продуктивному середовищі.

Для тестування REST API системи моніторингу було використано інструмент Postman (рисунок 3.15), який дозволив зручно виконувати HTTP-запити до локального сервера, перевіряти коректність відповідей у форматі JSON, а також оцінювати стабільність роботи серверних маршрутів.

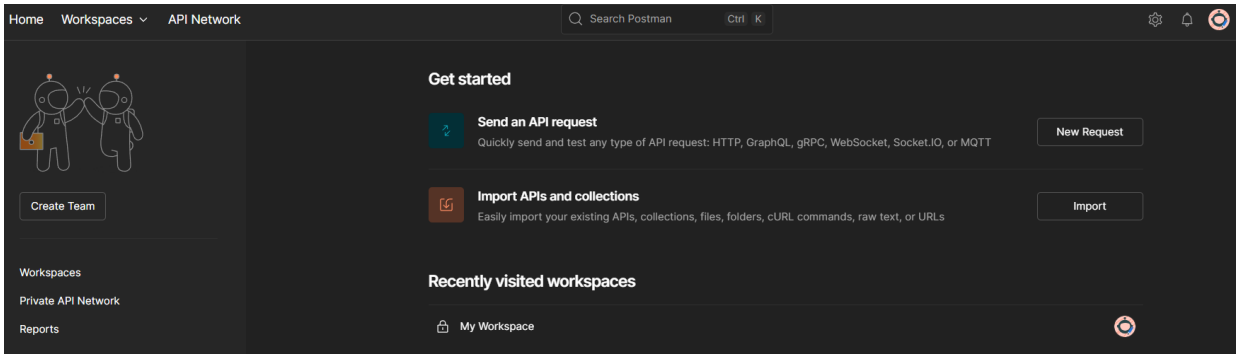

Рисунок 3.15 – Інструмент Postman [24]

Було протестовано такі ключові кінцеві точки:

– GET /api/data – повертає останні 500 записів із бази даних для перегляду поточних та історичних значень;

– GET /api/export_csv – забезпечує експорт даних у форматі CSV, що зручно для подальшого аналізу або збереження;

– POST /predict_health – приймає на вхід параметри якості повітря (PM2.5, PM10, температура, вологість) і повертає показник рівня ризику для здоров’я на основі моделі машинного навчання.

Фронтенд додатку перевірявся вручну в таких браузерах:

– Google Chrome;

– Microsoft Edge.

Було протестовано:

– перемикання теми (світла/темна);

– оновлення даних без перезавантаження сторінки;

– відображення поточних показників у таблицях;

– роботу графіка за останні 7 днів;

– активацію сповіщення при високому ризику (рисунок 3.16);

– коректність відображення нульових або відсутніх значень;

– адаптивність на мобільних пристроях (шрифт, блоки, прокрутка);

– функціонування кнопки експорту CSV.

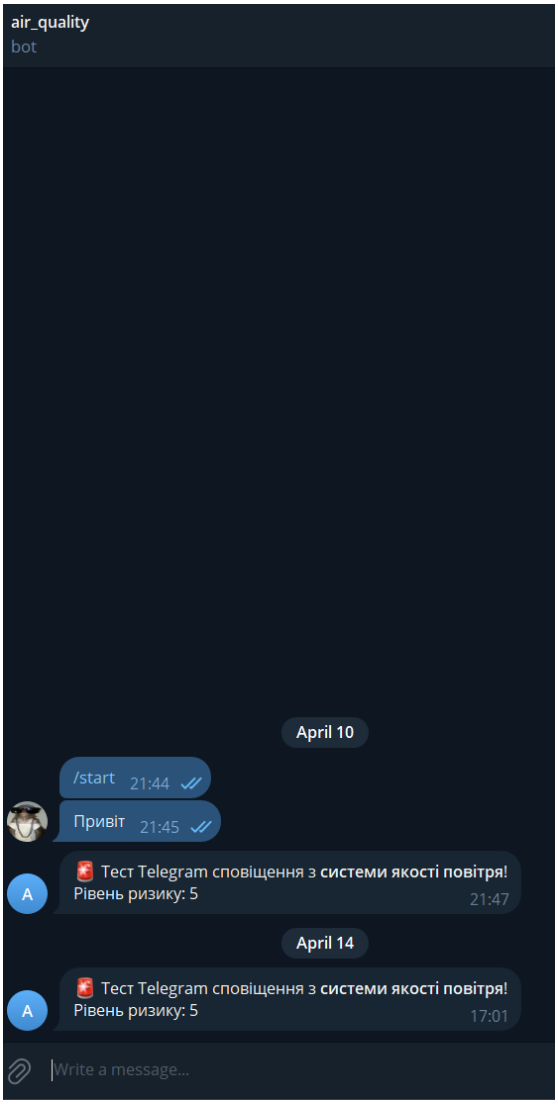

Рисунок 3.16 – Сповіщення при високому ризику

ВИСНОВКИ

Спроєктовано систему збору екологічних даних. Забезпечено коректне зчитування температури, вологості та умовного рівня забруднення повітря. Отримані значення відображаються на OLED-дисплеї, що дає змогу проводити локальний моніторинг у режимі реального часу.

Створено симуляційну модель системи в середовищі Wokwi, що дозволила протестувати апаратну й програмну частину без використання фізичних компонентів. Модель продемонструвала працездатність логіки збору, обробки та відображення даних, а також надала можливість перевірити зміну значень сенсорів і реакцію системи на зміну умов.

На основі технічних характеристик, підтримки інтерфейсів, обсягу пам’яті й продуктивності обґрунтовано вибір ESP32 як основи для реалізації системи. Його переваги відкривають можливість масштабування й додавання нових функцій.

Спроєктовано серверну частину системи з використанням Python, Flask та PostgreSQL. Реалізовано REST API для приймання та обробки даних, організовано збереження в базі з урахуванням часових міток та геолокації. Передбачено можливість подальшого розширення функціоналу – зокрема, шляхом інтеграції мобільних додатків або зовнішніх клієнтів.

Інтегровано модель машинного навчання для оцінки ризику здоров’я на основі екологічних параметрів. Модель класифікує рівень небезпеки за п’ятибальною шкалою та, за потреби, генерує Telegram-сповіщення при перевищенні критичних значень. Це забезпечує функціональність системи раннього попередження.

Розроблено фронтенд на основі React.js і Leaflet, що забезпечує візуалізацію даних у вигляді графіків, таблиць та інтерактивної карти. Передбачено підтримку темної теми, адаптивного інтерфейсу та експорту даних у форматі CSV. Це забезпечує зручну взаємодію з користувачем і підвищує доступність інформації.

Запропоновано архітектурне рішення, що дозволяє легко розширювати систему, додаючи нові сенсори (шум, тиск), виконавчі пристрої та канали зв’язку

(LoRa, GSM). Архітектура побудована за модульним принципом, що забезпечує високу адаптивність до змін і можливість використання системи в різних умовах.

Розроблену систему можна використовувати як основу для створення мобільних або стаціонарних станцій моніторингу повітря. Вона має потенціал для подальшого вдосконалення в напрямках персоналізації оцінок, автоматизованого реагування та глибшої аналітики впливу на здоров’я населення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Air pollution. *World Health Organization (WHO)*. URL: https://www.who.int/health-topics/air-pollution#tab=tab_1 (дата звернення: 01.05.2025).

2. Air quality concentrations. *European Environment Agency*. URL: https://www.eea.europa.eu/themes/air/air-quality-concentrations (дата звернення: 01.05.2025).

3. AirNow Interactive Map. *Folder*. URL: https://surl.li/ffiwph (дата звернення: 01.05.2025).

4. AirQ+. *World Health Organization (WHO)*. URL: https://www.who.int/tools/airq (дата звернення: 01.05.2025).

5. Ambient (outdoor) air pollution. *World Health Organization (WHO)*. URL: https://surli.cc/ddtywc (дата звернення: 01.05.2025).

6. AQI Basics. *AirNow.gov*. URL: https://www.airnow.gov/aqi/aqi-basics/ (дата звернення: 01.05.2025).

7. Billions of people still breathe unhealthy air: new WHO data. *World Health Organization (WHO)*. URL: https://www.who.int/news/item/04-04-2022-billions-of-people-still-breathe-unhealthy-air-new-who-data (дата звернення: 24.05.2025).

8. DHT22 temperature-humidity sensor + extras. Adafruit Industries, Unique & fun DIY electronics and kits. URL: https://www.adafruit.com/product/385 (дата звернення: 13.05.2025).

9. DL G. D. A Quick Tutorial on Deploying a Python Flask Application Using Docker. *Medium*. URL: https://surl.li/qicrwx (дата звернення: 14.05.2025).

10. Environmental Benefits Mapping and Analysis Program - Community Edition (BenMAP-CE). *US EPA*. URL: https://www.epa.gov/benmap (дата звернення: 01.05.2025).

11. Esp32 Dev Module Download. User Guide and Engine Fix Collection. URL: https://surl.li/wheqxf (дата звернення: 13.05.2025).

12. Free Open-Source Weather API. *Open-Meteo.com*. URL: https://open-meteo.com/ (дата звернення: 16.05.2025).

13. GPS Module Interfacing with NodeMCU. *ElectronicWings. Hardware Developers Community*. URL: https://www.electronicwings.com/nodemcu/gps-module-interfacing-with-nodemcu (дата звернення: 24.05.2025).

14. Leaflet an open-source JavaScript library for interactive maps. *Leaflet a JavaScript library for interactive maps*. URL: https://leafletjs.com/ (дата звернення: 14.05.2025).

15. MQ135 Air Quality Sensor Module. *Cytron Technologies*. URL: https://www.cytron.io/p-mq135-air-quality-sensor-module (дата звернення: 13.05.2025).

16. MQTT Essentials: Your 2025 Learning Hub for IoT & IIoT Data Streaming. *HiveMQ. Unlock the value of your data with MQTT.* URL: https://www.hivemq.com/mqtt/ (дата звернення: 14.05.2025).

17. MQTT für Dummies. *Business -Software- und IT-Blog - Wir gestalten digitale Wertschöpfung*. URL: https://blog.doubleslash.de/software-technologien/mqtt-fuer-dummies (дата звернення: 14.05.2025).

18. Ndichu T. Getting Started with Leaflet.js and React: Rendering a Simple Map. *Medium*. URL: https://surl.li/jnohrj (дата звернення: 14.05.2025).

19. Nearly 50 million people sign up call for clean air action for better health. *World Health Organization (WHO)*. URL: https://www.who.int/news/item/17-03-2025-nearly-50-million-people-sign-up-call-for-clean-air-action-for-better-health?utm_source=chatgpt.com (дата звернення: 24.05.2025).

20. PMS5003 Luftkvalitetssensor PM2.5 Module. *Ardustore.dk.* URL: https://ardustore.dk/produkt/pms5003-luftkvalitetssensor-pm2-5-module (дата звернення: 13.05.2025).

21. PMS7003 Laser Dust Sensor. *วัดฝุ่น PM2.5 ตรวจคุณภาพอากาศ ใช้กับ MCU Arduino, ESP32, Pi. อิมิคอนซิสเท็ม (iMicon System).* URL: https://www.imiconsystem.com/product/pms7003/ (дата звернення: 24.05.2025).

22. PostgreSQL. URL: https://www.postgresql.org/ (дата звернення: 14.05.2025).

23. posthresql functions in-depth. Online Course. *TutorialsPoint.* URL: https://surl.lu/pglotg (дата звернення: 14.05.2025).

24. Postman: The World’s Leading API Platform. Sign Up for Free. *Postman API Platform*. URL: https://www.postman.com/ (дата звернення: 16.05.2025).

25. Railway. URL: https://railway.com/ (дата звернення: 16.05.2025).

26. Real-time Air Quality Monitoring by PurpleAir. *PurpleAir*. URL: https://www2.purpleair.com/ (дата звернення: 01.05.2025).

27. Sensor.Community. Build your own sensor and join the worldwide civic tech network. URL: https://sensor.community/en/ (дата звернення: 01.05.2025).

28. The AI PaaS for Deploying, Managing, and Scaling Apps. *Heroku*. URL: https://www.heroku.com/ (дата звернення: 16.05.2025).

29. Welcome to Flask. Flask Documentation (3.1.x). URL: https://flask.palletsprojects.com/en/stable/ (дата звернення: 14.05.2025).

30. Wokwi ESP32, STM32, Arduino Simulator. Wokwi World’s most advanced ESP32 Simulator. URL: https://wokwi.com/projects/430783390549202945 (дата звернення: 14.05.2025).

31. Міжнародна науково-практична конференція молодих вчених та студентів. «*Програмне та апаратне забезпечення в інформаційних технологіях*». URL: https://conference.net.ua/ (дата звернення: 19.05.2025).

32. Пась В., Лавренчук С. Система моніторингу забруднення повітря з прогнозуванням впливу на здоров’я людей. *Програмне та апаратне забезпечення в інформаційних технологіях: матеріали Міжнар. науково- практ. конф., м. Луцьк, 6 травня 2025 р. м. Луцьк, 2025*. С. 155-158.

ДОДАТКИ

Додаток А

Тези в збірнику конференції

Міністерство освіти і науки України
Луцький національний технічний університет (м. Луцьк)
Наукове товариство студентів, аспірантів, докторантів та молодих вчених ЛНТУ (м. Луцьк)
Люблінська Політехніка (Польща, Люблін)
Університету «Дюнаре де Жос» (Румунія, Галац)
Університет Коменського (Словаччина, Братислава)
Університет Оснабрюк (Німеччина, Оснабрюк)
Університет Трансмонтани і Верхнього Дору (Португалія, Віла-Реал)
Чеський університет природничих наук (Чехія, Прага)
Національний університет «Чернігівська Політехніка» (м. Чернігів)
Тернопільський національний технічний університет імені Івана Пулюя (м. Тернопіль)
Чернівецький національний університет імені Юрія Федьковича (м. Чернівці)

6 травня 2025 року, м. Луцьк

ПРОГРАМНЕ ТА АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ

Тези доповідей Міжнародної науково-практичної конференції молодих вчених та студентів

SOFTWARE AND HARDWARE IN INFORMATION TECHNOLOGIES

Abstracts of the International Scientific Conference for Young Scientists and Students

Випуск 1

Луцький національний технічний університет

Луцьк – 2025

2.Шкварок О. В. Розробка та дослідження автоматизованої системи управління розумним будинком на базі бездротових мереж. URL: https://surl.li/vgmolo (дата звернення: 02.04.2025 р.)

3.Квітень Д. О. Веб-інтерфейс для управління ресурсами Інтернету речей в корпоративних мережах. URL: https://surl.li/hnkkqu (дата звернення: 02.04.2025 р.)

4.Кравець М. В. Web-додаток управління побутовими системами розумного будинку. URL: https://surl.li/gzootn (дата звернення: 02.04.2025 р.)

Вадим ПАСЬ,
здобувач вищої освіти,
Світлана ЛАВРЕНЧУК,
канд. техн. наук, доцент
Луцький національний технічний університет

СИСТЕМА МОНІТОРИНГУ ЗАБРУДНЕННЯ ПОВІТРЯ З ПРОГНОЗУВАННЯМ ВПЛИВУ НА ЗДОРОВ'Я ЛЮДЕЙ

Забруднення повітря є серйозною загрозою для здоров'я населення, особливо в урбанізованих регіонах. Існуючі методи моніторингу обмежені високою вартістю обладнання або неповним територіальним охопленням, що ускладнює оцінку реального впливу забруднення на здоров'я людей. Тому розробка ефективної системи моніторингу з можливістю прогнозування впливу на здоров'я людей є актуальною задачею.

Про актуальність цієї теми також свідчить велика кількість досліджень та публікацій. Наприклад, в статті [1] розглядається використання моделей машинного навчання для класифікації забруднюючих речовин у реальному часі за даними оптичних мікродатчиків, оцінюються три ML-підходи – XGBoost, LSTM і приховані ланцюги Маркова. Результати підтверджують ефективність мікродатчиків і ML для моніторингу якості повітря.

В дослідженні [2] описано погодинний прогноз концентрацію PM2,5 у приміщеннях 24 будівель в Австралії. Дані зібрані з 91 датчика у 8 містах (2019-2022). Автори використовують алгоритм DEML із трьома базовими моделями (SVM, RF, XGBoost) і двома метамоделями (RF, GLM). В роботі відмічено, що вплив зовнішнього PM2,5 був особливо помітним під час лісових пожеж.

Глобальний ринок систем моніторингу якості повітря стрімко зростає: з $5,82 млрд у 2024 році до прогнозованих $12,06 млрд у

2034-му (CAGR 7,56%). У Північній Америці ринок оцінюється в $2,21 млрд у 2024 році, із прогнозованим зростанням 7,66% щороку [3]. Це підкреслює зростаючу потребу в ефективному контролі якості повітря.

Метою запропонованої системи є моделювання та аналіз даних датчиків якості повітря за допомогою бібліотек Python (TensorFlow, Scikit-learn). Система генерує симульовані набори даних, що імітують показники датчиків MQ-135 (аміак, CO_2, оксиди азоту), PMS5003 (PM2.5, PM10) і DHT22 (температура, вологість), які використовуються для тестування алгоритму прогнозування та оцінки якості повітря.

Для прогнозування впливу якості повітря на здоров'я людей застосовується алгоритм машинного навчання Random Forest.

Система моніторингу складається з таких основних компонентів:

– сенсори: MQ-135 для визначення рівня газів-забрудників; PMS5003 для аналізу твердих частинок у повітрі; DHT22 для вимірювання температури та вологості;

– передача даних відбувається через протокол MQTT, що дає змогу збирати дані у режимі реального часу;

– дані зберігаються в базі даних PostgreSQL (рис. 1);

	id [PK] integer	timestamp character varying (50)	sensor_id integer	sensor_type character varying (50)	latitude double precision	longitude double precision	temperature double precision	humidity double precision	co2 double precision	pm2_5 double precision	pm10 double precision
75	75	2025-03-26 21:54:14	3	PMS5003	50.702935	25.286108	[null]	[null]	[null]	65.77	56.24
76	76	2025-03-26 21:54:14	3	PMS5003	50.702935	25.286108	[null]	[null]	[null]	65.77	56.24
77	77	2025-03-26 21:54:14	1	DHT22	50.743967	25.430634	22.16	57.77	[null]	[null]	[null]
78	78	2025-03-26 21:54:14	1	DHT22	50.743967	25.430634	22.16	57.77	[null]	[null]	[null]
79	79	2025-03-26 21:54:24	3	PMS5003	50.735224	25.443542	[null]	[null]	[null]	17.76	123.41
80	82	2025-03-26 21:54:24	2	MQ-135	50.736315	25.402162	[null]	[null]	471.39	[null]	[null]
81	84	2025-03-26 21:54:24	1	DHT22	50.724787	25.337534	22.54	49.9	[null]	[null]	[null]
82	85	2025-03-26 21:54:34	3	PMS5003	50.741106	25.448143	[null]	[null]	[null]	38.55	67.23

Рисунок 1 – Дані з сенсорів в базі даних PostgreSQL

– моделювання забруднення на основі великих обсягів симульованих даних, що дозволяє отримувати прогнозні показники навіть за умов обмеженої кількості фізичних датчиків;

– аналіз та прогнозування за допомогою моделі Random Forest;

– візуалізація результатів у веб-інтерфейсі на базі React.js та Leaflet, що дозволяє користувачам оцінювати прогнозні показники забруднення та його вплив на здоров'я (рис. 2).

Рисунок 2 – Веб-інтерфейс на базі React.js та Leaflet

Розроблена система моніторингу якості повітря інтегрує сучасні сенсори, алгоритми машинного навчання та веб-візуалізацію, забезпечуючи точний аналіз і прогнозування рівня забруднення. Використання протоколу MQTT гарантує безперервний збір даних у режимі реального часу, а PostgreSQL ефективно обробляє великі масиви інформації. Алгоритм Random Forest дозволяє передбачати вплив забруднення на здоров'я. Завдяки інтерактивному веб-інтерфейсу на базі React.js та Leaflet користувачі отримують доступ до актуальних даних про стан забруднення повітря. Використання симуляції даних забезпечує значні переваги, такі як зменшення витрат на обладнання та можливість отримання детальних прогнозів за умов обмеженої кількості реальних датчиків. Подальший розвиток системи можливий в напрямку інтеграції глибокого навчання, розширення мережі сенсорів та підключення IoT-рішень для ще більшої точності й масштабованості прогнозування. Також можна розвивати проєкт в напрямку вдосконалення алгоритмів аналізу та інтеграції системи з мобільними додатками для оперативного оповіщення користувачів про критичні рівні забруднення. Це сприятиме ефективному контролю якості повітря та зниженню екологічних ризиків.

Перелік використаних джерел

1. Azeraf Elie et al. Real-time Pollutant Identification through Optical PM Micro-Sensor. *arXiv*. 2025. URL: https://www.arxiv.org/abs/2503.10724

2. Indoor Pm2.5 Forecasting and the Association with Outdoor Air Pollution: A Modelling Study Based on Sensor Data in Australia. URL:

https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4952937 (дата звернення: 28.03.2025 р.)

3. Zoting S. Air Quality Monitoring System Market Size, Share and Trends 2024 to 2034. *Precedence Research*. URL: https://www.precedenceresearch.com/air-quality-monitoring-system-market (дата звернення: 24.03.2025 р.)

Павло ПИРОГОВ,
аспірант
Харківський національний університет
імені В. Н. Каразіна

МЕТРИКИ ТА КРИТЕРІЇ ДЛЯ АНАЛІЗУ РЕСУРСНИХ ВИТРАТ КВАНТОВИХ АЛГОРИТМІВ

Інтенсивний розвиток квантових технологій ставить перед дослідниками нові завдання щодо аналізу та оптимізації ресурсних витрат при реалізації квантових алгоритмів. Однією з ключових проблем є ефективне використання обмежених ресурсів необхідних для виконання алгоритмів на реальному обладнанні, таких як кількість кубітів, глибина квантових схем, а також складність і кількість квантових вентилів тощо.

У цьому контексті набуває особливої важливості систематизація метрик і критеріїв, що дозволяють комплексно оцінити ресурсоємність алгоритмічних рішень. Метою даної тези є визначення найважливіших метрик і критеріїв для об'єктивної оцінки ресурсних витрат квантових алгоритмів, що забезпечить їх ефективну реалізацію та подальше застосування у практичних задачах.

Одним із ключових параметрів є кількість кубітів, де розрізняють логічні та фізичні кубіти. Логічні кубіти відображають мінімальні ідеалізовані вимоги алгоритму, тоді як фізичні кубіти враховують додаткові витрати на кореляцію помилок [1]. Велика кількість фізичних кубітів суттєво ускладнює апаратну реалізацію квантового обчислення.

Другим важливим параметром є глибина квантової схеми, що визначає кількість послідовних шарів вентилів. Менша глибина сприяє зниженню впливу декогеренції. Кількість квантових вентилів, особливо багатокубітних, істотно впливає на складність

Додаток Б
Програмування серверної частини

```
from flask import Flask, jsonify, Response, request
from flask_sqlalchemy import SQLAlchemy
from flask_cors import CORS
from apscheduler.schedulers.background import BackgroundScheduler
import requests
import logging
from datetime import datetime
import pytz
from io import StringIO
import csv
import joblib
import numpy as np
from notify import send_alert

app = Flask(__name__)
CORS(app)

app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql://postgres:555@localhost/air_quality'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

class AirQuality(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    timestamp = db.Column(db.DateTime, nullable=False)
    address = db.Column(db.String(120), nullable=False)
    sensor_type = db.Column(db.String(50), nullable=False)
    value = db.Column(db.Float, nullable=True)
    unit = db.Column(db.String(20), nullable=True)

    def to_dict(self):
        return {
            'timestamp': self.timestamp.strftime("%Y-%m-%d %H:%M:%S"),
            'address': self.address,
            'sensor_type': self.sensor_type,
            'value': self.value,
            'unit': self.unit
        }

with app.app_context():
    db.create_all()

# Завантаження моделі
model = joblib.load("backend/health_risk_model_all.pkl")
scaler = joblib.load("backend/health_risk_scaler_all.pkl")

def fetch_live_data():
    try:
        aq_response = requests.get(
            "https://air-quality-api.open-meteo.com/v1/air-quality",
            params={
                "latitude": 50.7472,
```

```
                "longitude": 25.3254,
                "current": "pm10,pm2_5"
            }
        )
        aq_response.raise_for_status()
        aq_data = aq_response.json().get("current", {})

        weather_response = requests.get(
            "https://api.open-meteo.com/v1/forecast",
            params={
                "latitude": 50.7472,
                "longitude": 25.3254,
                "current": "temperature_2m,relative_humidity_2m"
            }
        )
        weather_response.raise_for_status()
        weather_data = weather_response.json().get("current", {})

        full_data = {
            **aq_data,
            "temperature": weather_data.get("temperature_2m"),
            "humidity": weather_data.get("relative_humidity_2m")
        }

        with app.app_context():
            now = datetime.now(pytz.timezone("Europe/Kyiv"))
            address = "м. Луцьк"

            measurements = {
                "pm10": (full_data.get("pm10"), "µg/m³"),
                "pm2_5": (full_data.get("pm2_5"), "µg/m³"),
                "temperature": (full_data.get("temperature"), "°C"),
                "humidity": (full_data.get("humidity"), "%")
            }

            for sensor, (value, unit) in measurements.items():
                if value is not None:
                    record = AirQuality(
                        timestamp=now,
                        address=address,
                        sensor_type=sensor,
                        value=value,
                        unit=unit
                    )
                    db.session.add(record)
            db.session.commit()
            logging.info("✅ Дані успішно збережено у БД")

    except Exception as e:
        logging.error(f"❌ Open-Meteo fetch error: {e}")

@app.route("/api/data")
def get_data():
    data = AirQuality.query.order_by(AirQuality.timestamp.desc()).limit(500).all()
    return jsonify([d.to_dict() for d in data])
```

```
@app.route("/api/export_csv")
def export_csv():
    data = AirQuality.query.order_by(AirQuality.timestamp.desc()).all()

    si = StringIO()
    cw = csv.writer(si)
    cw.writerow(['timestamp', 'address', 'sensor_type', 'value', 'unit'])

    for d in data:
        cw.writerow([d.timestamp, d.address, d.sensor_type, d.value, d.unit])

    output = si.getvalue()
    return Response(
        output,
        mimetype="text/csv",
        headers={"Content-Disposition": "attachment;
filename=air_quality_data.csv"}
    )

@app.route("/predict_health", methods=["POST"])
def predict_health():
    try:
        data = request.get_json()

        pm2_5 = data.get("pm2_5", 0)
        pm10 = data.get("pm10", 0)
        temperature = data.get("temperature", 0)
        humidity = data.get("humidity", 0)

        features = np.array([[pm2_5, pm10, humidity, temperature]])
        features_scaled = scaler.transform(features)
        prediction = model.predict(features_scaled)[0]

        # 🔔 Сповіщення при високому ризику
        if prediction >= 4:
            msg = (
                f"🚨 <b>Попередження про високий ризик</b>\n"
                f"PM2.5: {pm2_5} µg/m³\n"
                f"PM10: {pm10} µg/m³\n"
                f"Температура: {temperature}°C\n"
                f"Вологість: {humidity}%\n"
                f"<b>Рівень ризику: {prediction}</b>"
            )
            send_alert(msg)

        return jsonify({"health_risk": int(prediction)})

    except Exception as e:
        logging.error(f"❌ Prediction error: {e}")
        return jsonify({"error": str(e)}), 500

@app.route("/")
def home():
    return "Сервер працює. Дані доступні на /api/data"

if __name__ == "__main__":
```

```
scheduler = BackgroundScheduler()
scheduler.add_job(fetch_live_data, 'interval', seconds=30)
scheduler.start()
app.run(debug=True)
```