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

КВАЛІФІКАЦІЙНА РОБОТА

Кіберфізична система "Цифровий двійник міста"
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.2302131.23.02.33 ПЗ

Виконав здобувач IV курсу, група КІ2с-23-2 _____ Роман ДУДА
Підпис Ініціали, прізвище

Керівник _____ Юрій ВОЙЧУР
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 КІС
Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дуді Роману Сергійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система "Цифровий двійник міста"

Керівник проекту (роботи) Войчур Юрій Олексійович, д.ф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2026 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 предметної області цифрових двійників міста та постановка задачі щодо розроблення кіберфізичної системи «Цифровий двійник міста»

Проєктування кіберфізичної системи «Цифровий двійник міста»

Програмно-апаратна реалізація та тестування кіберфізичної системи CityTwin

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

Архітектура ПЗ проєкту

Структурна схема кіберфізичної системи

Апаратне забезпечення та вузол збору даних

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 – Аналіз предметної області та існуючих рішень у сфері цифрових двійників міста	01.03.2026	виконано
4	Робота над розділом 2 – Проєктування кіберфізичної системи «Цифровий двійник міста»	01.04.2026	виконано
5	Робота над розділом 3 – Програмно-апаратна реалізація та тестування кіберфізичної системи CityTwin	29.04.2026	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконано
7	Попередній захист ВКР	26.05.2026	виконано
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач _____ Роман ДУДА
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Юрій ВОЙЧУР
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ.2302131.23.02.33 ПЗ	Пояснювальна записка	63		
			Графічні матеріали			
2		КвРКІ.2302131.23.02.33 Е8	Архітектура ПЗ проєкту	1		
3		КвРКІ.2302131.23.02.33 Е8	Структурна схема кіберфізичної системи	1		
4		КвРКІ.2302131.23.02.33 Е8	Апаратне забезпечення проєкту	1		

Зм	Арк	№ докум	Підпис	Дата	КвРКІ.2302131.23.02.33 ВП		
Розробив		Дуда			Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Войчур				У	1	1
Н. контр.		Лисенко				ХНУ, КІ2с-23-2		
Затв.		Павлова						

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Кіберфізична система «Цифровий двійник міста»».

Автор роботи: Дуда Роман.

Керівник роботи: Юрій Войчур.

Пояснювальна записка: 63 с., 8 рис., 8 табл., 3 дод., 50 джерел.

Графічна частина: 3 креслення.

АРХІТЕКТУРА, ГЕОДАНІ, КІБЕРФІЗИЧНА СИСТЕМА, МІСЬКЕ СЕРЕДОВИЩЕ, OPENSTREETMAP, ЦИФРОВИЙ ДВІЙНИК.

Кваліфікаційна робота бакалавра присвячена розробленню програмно-апаратної реалізації кіберфізичної системи «Цифровий двійник міста». Розроблена система CityTwin є інтерактивною браузерною платформою, що використовує відкриті геодані, погодні параметри та математичні моделі для оцінювання впливу міської забудови на вітровий комфорт, шумове навантаження та тепловий стрес. Актуальність теми зумовлена потребою у швидких інструментах попереднього аналізу міських сценаріїв, які можуть використовуватися до проведення складних інженерних розрахунків.

Метою роботи є проєктування, реалізація та тестування кіберфізичної системи «Цифровий двійник міста» для попереднього аналізу впливу забудови на вітровий комфорт, шумове навантаження та тепловий стрес. Для цього проаналізовано предметну область, обґрунтовано технологічний стек, спроєктовано структуру КФС, апаратний рівень і програмну архітектуру, реалізовано моделі аналізу та перевірено працездатність системи тестами.

Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

Коли на карті міста з'являється нова будівля, на перший погляд це просто ще один контур серед інших об'єктів. Але в реальному середовищі така зміна може вплинути на рух повітря між будинками, посилити шум у дворі або змінити нагрівання відкритих поверхонь у спекотний день. Саме тому для оцінювання міських рішень недостатньо дивитися тільки на план забудови. Потрібен інструмент, який хоча б на попередньому рівні показує, як просторові зміни можуть відбитися на умовах життя в кварталі.

У межах кваліфікаційної роботи цифровий двійник міста розглянуто як прикладну систему, у якій карта, погодні дані та розрахункові моделі працюють разом. Якщо користувач додає умовну будівлю або зелену зону, система повинна не просто відобразити новий об'єкт, а показати, як через це змінюються вітер, шумове поле або теплове навантаження.

Практична реалізація виконана у вебзастосунку CityTwin. Під час розроблення було використано MapLibre GL для роботи з картою, OpenFreeMap і OpenStreetMap для отримання просторових даних, Open-Meteo для погодної інформації та TypeScript для клієнтської логіки. Такий набір технологій обрано не випадково: застосунок можна запускати у браузері, без окремої складної серверної частини, а це спрощує перевірку і демонстрацію результату [1-7].

Актуальність теми пов'язана з тим, що багато рішень у міському середовищі спочатку оцінюються приблизно. Наприклад, ще до детальних CFD або акустичних розрахунків корисно побачити, у якій частині кварталу може прискорюватися вітер, де шум залишається найбільшим, чи дає озеленення помітний ефект для теплового фону. CityTwin не формує остаточний нормативний висновок, але допомагає швидко побачити напрям змін і зрозуміти, які сценарії варто перевіряти глибше.

Метою кваліфікаційної роботи є розроблення програмно-апаратної реалізації кіберфізичної системи «Цифровий двійник міста» для інтерактивного

оцінювання параметрів міського середовища за відкритими геоданими та погодною інформацією. У межах проєкту основну увагу приділено трьом показникам, які добре помітні для користувача і водночас потребують розрахункової логіки: вітровому комфорту, шумовому полю та тепловому стресу.

Об'єктом дослідження є процес проєктування програмно-апаратної кіберфізичної системи цифрового двійника міського середовища. Предметом дослідження є побудова цифрового двійника, у якому карта, витяг будівель і зелених зон, розрахункові модулі, сценарне редагування та автоматизоване тестування об'єднані в один робочий потік.

Щоб досягти поставленої мети, спочатку потрібно було розібратися, які можливості цифрового двійника реально реалізувати в межах бакалаврської роботи, а які краще залишити як напрям подальшого розвитку. Після цього було обрано джерела картографічних і погодних даних, спроєктовано структуру CityTwin, реалізовано модулі аналізу вітру, шуму й тепла, додано сценарний редактор міських втручань, багатомовний інтерфейс і перевірку основних функцій тестами.

Практична цінність роботи полягає в тому, що результатом став не лише теоретичний опис, а прототип, з яким можна працювати. Його можна використати для навчальної демонстрації цифрового двійника, для пояснення впливу простих міських втручань і для швидкого порівняння кількох варіантів на карті. При цьому важливо розуміти межі системи: вона не замінює професійні інженерні розрахунки, а дає зрозумілу попередню оцінку.

Пояснювальна записка має три розділи. У першому розділі розглянуто, що таке цифровий двійник міста, які геодані можуть бути його основою і як загалом підходять до аналізу вітру, шуму та теплового стресу. У другому розділі описано вимоги до CityTwin, архітектуру системи та логіку моделей. У третьому розділі показано програмно-апаратну реалізацію, приклади візуалізації, результати

тестування і висновки щодо отриманого продукту. Реалізацію та тестування подано у розділі 3.

У міському плануванні часто виникає ситуація, коли рішення ще перебуває на рівні попередньої ідеї, але його можливі наслідки вже потрібно оцінити. На цьому етапі не завжди доцільно запускати складні інженерні розрахунки для кожного варіанта, однак повністю ігнорувати вплив змін також не можна. Саме тут корисним стає інструмент, який дозволяє швидко порівняти сценарії та побачити, чи не створює певне втручання очевидних проблем.

Для міського середовища така попередня оцінка особливо важлива. Нова забудова, зміна відкритого простору або додавання зелених зон можуть по-різному впливати на комфорт території. Навіть якщо результат є наближеним, він допомагає краще зрозуміти, у якому напрямі змінюється ситуація і на що варто звернути увагу під час подальшого аналізу.

CityTwin у цій роботі розглядається саме як такий проміжний інструмент. Його задача полягає не в тому, щоб замінити професійні розрахункові комплекси, а в тому, щоб зробити перший аналіз більш наочним. Користувач бачить карту, змінює сценарій і отримує візуальний результат, який можна інтерпретувати без складної технічної підготовки.

Такий підхід також має навчальну цінність. Він показує, що цифровий двійник - це не лише технологічний термін, а практична модель, у якій карта, дані та розрахунки працюють разом. Завдяки цьому легше пояснити, чому міські рішення потрібно оцінювати не тільки за зовнішнім виглядом, а й за впливом на умови середовища.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ У СФЕРІ ЦИФРОВИХ ДВІЙНИКІВ МІСТА

1.1 Поняття цифрового двійника міського середовища

У межах цієї роботи цифровий двійник міського середовища розуміється передусім як робоча модель фрагмента міста. Йдеться не про карту, на якій лише красиво показані будівлі, а про програмне середовище, де геометрія кварталу, погода, розрахункові алгоритми і дії користувача зібрані в один процес. Для CityTwin це принциповий момент: користувач не просто дивиться на місто, а перевіряє, як воно може поводитися за різних сценаріїв [1-2, 22-23].

На звичайній карті квартал сприймається досить спокійно: є контури будинків, вулиці, зелені зони, водойми, окремі підписи. Для орієнтування цього достатньо. Але коли потрібно оцінити наслідки зміни забудови, така карта швидко стає обмеженою. Нова висотна будівля на плані може виглядати як невелике доповнення до існуючої геометрії, хоча для руху повітря, шумового фону або теплового режиму вона вже змінює умови в усьому локальному фрагменті.

Саме тому в CityTwin просторові дані не сприймаються як декоративна підкладка. Будівлі, дороги, зелені насадження і водойми розглядаються як об'єкти, з яких починається подальший аналіз. Якщо в моделі є тільки візуальна форма, але немає зв'язку між цією формою і розрахунками, цифровий двійник фактично перетворюється на звичайну інтерактивну карту. У такому випадку складно пояснити, чому певний сценарій отримав саме такий результат.

Кіберфізична логіка CityTwin побудована навколо послідовного циклу роботи з міським середовищем. Спочатку система отримує дані про квартал, потім виконує вибраний аналіз, після цього показує результат на карті і дає змогу змінити сценарій. Важливо, що цей цикл не завершується одним розрахунком. Користувач може повернутися до карти, додати інший об'єкт, змінити умови і знову побачити, як змінилася просторова картина.

Такий підхід робить зворотний зв'язок не другорядною деталлю, а основою роботи системи. Для користувача важливо не тільки отримати умовний показник, а й побачити, де саме на карті змінився вітер, шум або теплове навантаження. Порівняння базового стану з новим варіантом дозволяє оцінювати сценарій не абстрактно, а через конкретні ділянки міського простору.

У програмній реалізації це привело до чіткого розподілу ролей між частинами застосунку. Інтерфейс відповідає за вибір міста, режиму аналізу та сценарних дій. Карта показує просторовий контекст. Модулі витягу даних працюють з об'єктами з картографічних шарів. Аналітичні модулі виконують розрахунки і передають результат у форму, придатну для візуалізації. Такий поділ потрібен не для формальної архітектури, а для того, щоб система залишалась зрозумілою під час розвитку.

CityTwin у цій роботі не розглядається як завершена платформа міського управління. Це демонстраційний цифровий двійник, у якому навмисно використано моделі, придатні для роботи у браузері. Вони простіші за професійні CFD, акустичні або кліматичні комплекси, але саме завдяки цьому застосунок може швидко реагувати на дії користувача. Для бакалаврської роботи такий рівень є виправданим, оскільки він показує логіку системи і межі її використання.

Під час проєктування довелось постійно враховувати баланс між деталізацією і швидкістю. Якщо зробити модель надто важкою, користувач втратить можливість спокійно перевіряти кілька варіантів забудови або озеленення. Якщо ж надто спростити розрахунок, результат перестане бути корисним навіть для попереднього аналізу. Тому для CityTwin важливими стали розмір розрахункової сітки, кешування даних, спосіб оновлення оверлеїв і зрозуміле подання результату.

У межах CityTwin цифровий двійник також виконує пояснювальну функцію. Користувач бачить не тільки числовий результат, а просторову картину: ділянки з більшим вітровим впливом, зони підвищеного шуму або місця

з тепловим навантаженням. Такий формат зручний для навчання і для попереднього обговорення міських рішень, бо результат не захований у таблицях.

На рівні коду ці ідеї пов'язані з файлами README.md, src/App.tsx і src/components/MapScene.tsx. У них описано загальну логіку застосунку, сценарій взаємодії з картою та зв'язок між станом інтерфейсу і візуальним поданням результатів. Саме ці частини формують основу користувацького потоку: вибір міста, перегляд карти, запуск аналізу і порівняння сценаріїв.

Отже, у цій роботі цифровий двійник міського середовища розуміється як робоча програмна модель, яка поєднує дані, розрахунок і візуальний зворотний зв'язок. Для подальшого проєктування це означає потребу в єдиних типах даних, стабільній поведінці карти, прозорих межах моделі та автоматизованій перевірці основних функцій. Без цих елементів цифровий двійник швидко перетворюється на набір окремих демонстрацій, а не на цілісну систему.

1.2 Геоінформаційні дані як основа віртуальної моделі міста

У віртуальній моделі міста геодані фактично виконують роль початкового матеріалу. Саме з них CityTwin дізнається, де проходить дорога, де стоїть будівля, де є парк, а де починається водойма. Якщо ці об'єкти витягнуті неправильно, подальший розрахунок може виглядати переконливо на екрані, але він уже не буде добре прив'язаний до реального кварталу. Тому в цьому підрозділі геоінформаційні дані розглядаються не як фоновий шар карти, а як частина моделі, від якої залежить якість аналізу [2-5, 24-25].

Для CityTwin було важливо працювати з такими даними, які можна отримати без ручного створення окремої бази під кожне місто. Через це основним джерелом просторових об'єктів обрано OpenStreetMap. Його зручно використовувати у дипломному проєкті, бо одна й та сама логіка може працювати для різних локацій: користувач обирає місто, карта завантажує

потрібну область, а застосунок витягує з неї будівлі, дороги, зелені зони та інші об'єкти. Водночас OpenStreetMap не є ідеально рівномірним джерелом [2-5, 38-41]. У центральних районах дані часто детальніші, а на менш описаних територіях може бракувати висоти будівель, кількості поверхів або точного типу об'єкта.

У практичній роботі з відкритими геоданими важливо враховувати не тільки наявність об'єкта, а й повноту його опису. Для людини контур будівлі на карті вже виглядає достатнім, але для цифрового двійника цього мало. Система повинна розуміти, чи це житлова, промислова або допоміжна споруда, яку приблизну висоту вона має, як вона розташована відносно дороги, зелених зон та інших будівель. Саме ці характеристики потім впливають на розрахунок вітрових тіней, поширення шуму і теплового стану території.

Окрема складність полягає в тому, що геометрія в OpenStreetMap створюється різними учасниками спільноти. Через це навіть у межах одного міста можуть зустрічатися дуже детально описані квартали і ділянки, де об'єкти позначені спрощено. Для CityTwin це означає, що система має працювати не з ідеально підготовленою базою, а з реальними відкритими даними, у яких іноді відсутні висоти, класи доріг або точні типи територій. Такий підхід робить проєкт ближчим до реального використання, але одночасно вимагає обережного трактування результатів.

Через це робота з геоданими у CityTwin не зводиться до простого завантаження шару на карту. Система повинна враховувати, які саме властивості доступні, які можна використати напряму, а які потрібно оцінити наближено. Наприклад, висота будівлі може братися з тегів height, render_height або building:levels. Якщо таких даних немає, доводиться використовувати припущення, і це одразу впливає на точність подальшого моделювання.

Для браузерного застосунку важливим є формат векторних тайлів. Він дозволяє отримувати просторові об'єкти частинами, у межах поточної області карти, без власної серверної бази даних. Це добре підходить для дипломного

проєкту, бо користувач може переміщатися картою, вибирати місто і запускати аналіз без попереднього завантаження великого набору геоданих [3-4].

У структурі OpenMapTiles-сумісних шарів для CityTwin найбільше значення мають building, transportation, landuse і water. Перший шар дає контури забудови, другий допомагає визначити дорожню мережу для оцінювання шуму, третій використовується для зелених зон та типів територій, а четвертий - для водних об'єктів. Кожний із цих шарів потрібен не сам по собі, а як частина розрахункової картини міського середовища.

Одна з практичних проблем полягає в тому, що карта і модель мають різні вимоги до даних. Для відображення на екрані достатньо намалювати полігон, але для аналізу потрібно знати його тип, приблизну висоту, положення відносно інших об'єктів і межі розрахункової області. Тому після витягу з карти геометрія проходить нормалізацію і перетворюється у структури, з якими вже можуть працювати модулі вітру, шуму та тепла.

У CityTwin це реалізовано у файлах src/components/featureExtractor.ts, src/data/cityPresets.ts і src/data/osmBuildings.ts. Вони відповідають за отримання об'єктів із карти, опис початкових міст і роботу з будівлями, які використовуються під час аналізу. Завдяки такому поділу легше перевіряти, звідки взялися дані, як вони були перетворені і чому певний об'єкт потрапив або не потрапив у розрахунок.

Для зручної роботи застосунку також важливе кешування. Під час взаємодії з картою користувач може кілька разів запускати аналіз у тій самій області або змінювати сценарні об'єкти без повного оновлення всіх шарів. Тимчасове збереження результатів querySourceFeatures зменшує кількість повторних звернень до карти і робить реакцію інтерфейсу стабільнішою.

Водночас відкриті геодані мають природні обмеження. Їхня якість залежить від активності спільноти, деталізації конкретного міста і наявності тегів. Тому результати CityTwin потрібно трактувати як попередню оцінку, а не

як офіційний висновок. Якщо вхідна геометрія неповна, система може правильно виконати алгоритм, але показати менш точну картину реального середовища.

Для будівель найбільш критичними є контур і висота, бо саме вони визначають перешкоди для повітряного потоку та зони екранування шуму. Для доріг важливий не лише сам факт наявності лінії, а й її клас, оскільки магістральна вулиця і невеликий проїзд не повинні однаково впливати на шумову модель. Для зелених зон та водойм основне значення має правильне віднесення території до відповідного типу поверхні, адже ці об'єкти впливають на тепловий режим і сприйняття міського простору [39-40].

Тому підготовка геоданих у CityTwin фактично є окремим етапом моделювання. Перед тим як передати об'єкти в аналітичні модулі, система повинна привести їх до зрозумілого формату, відкинути зайве, зберегти корисні атрибути і не втратити просторовий зв'язок із картою. Без цього цифровий двійник залишився б лише візуальною оболонкою, а не інструментом для порівняння сценаріїв.

З цього випливає, що для цифрового двійника важливі не тільки формули аналізу, а й увесь шлях даних від карти до результату. Геометрія має бути отримана, очищена, приведена до єдиного формату і передана в моделі без втрати зв'язку з реальними об'єктами. Для подальшого проєктування CityTwin це означає потребу в стабільних типах даних, передбачуваних межах карти, зрозумілих припущеннях і тестах, які перевіряють не лише інтерфейс, а й коректність підготовки геоданих. Джерела даних наведено у табл. 1.1.

Таблиця 1.1 узагальнює джерела, з яких CityTwin бере просторову і погодну інформацію. Вона показує не лише назви сервісів, а й те, яку роль кожне джерело відіграє у моделі: OpenStreetMap та OpenFreeMap дають геометрію міста, Open-Meteo забезпечує погодні параметри, а локальні структури застосунку зберігають зміни, внесені користувачем у межах сценаріїв моделювання.

Таблиця 1.1 – Джерела даних системи CityTwin

Джерело	Дані	Спосіб використання	Обмеження
OpenFreeMap / OSM	Будівлі, дороги, вода, landuse	querySourceFeatures у MapLibre	Неповні висоти та різна деталізація
Open-Meteo	Вітер, температура, вологість, WMO	HTTP-запит з повторними спробами та тайм-аутом	Залежність від доступності API
ArcGIS Imagery	Супутникова підкладка	Опційний растровий шар	Не впливає на моделі
localStorage	Місто, тема, камера карти	safeStorage	Локальність на пристрої користувача

1.3 Методи моделювання вітрового комфорту в міській забудові

Вітровий комфорт у місті складно оцінити тільки за напрямком і швидкістю вітру з прогнозу погоди. У відкритому полі ці значення можуть бути достатніми для загального уявлення, але між будівлями потік поводиться інакше: десь він прискорюється, десь послаблюється, а за високими об'єктами виникають зони затінення. Саме тому в CityTwin вітровий режим [15-17] розглядається як просторове поле, яке залежить не тільки від погоди, а й від форми забудови.

Під час розроблення CityTwin важливо було не просто описати фізичні принципи руху повітря, а підібрати такий спосіб моделювання, який можна виконувати у браузері. Повноцінні CFD-розрахунки дають значно детальнішу картину, проте вони потребують складної сітки, значних обчислювальних ресурсів і часу. У CityTwin задача інша: швидко показати користувачу, як

приблизно зміниться картина вітру після додавання будівлі, зеленого об'єкта або іншого сценарного втручання.

Повноцінне CFD-моделювання зазвичай передбачає побудову детальної тривимірної області, задання граничних умов, вибір моделі турбулентності та виконання великої кількості ітерацій. Такі розрахунки мають високу цінність для інженерного проєктування, але вони погано відповідають логіці інтерактивного вебзастосунку. Користувач CityTwin очікує, що після зміни сценарію результат з'явиться майже одразу, а не після тривалого чисельного розрахунку.

Через це у дипломному проєкті важливо було не копіювати професійні CFD-комплекси, а підібрати модель, яка відповідає завданню роботи. CityTwin має показати напрям зміни: де після додавання будівлі може посилитися потік, де з'явиться зона затінення, які місця варто перевірити уважніше. Така попередня оцінка не претендує на нормативну точність, але вона корисна на ранньому етапі аналізу міського сценарію.

На практиці модель має враховувати кілька зрозумілих для міського середовища явищ. Перед будівлею може утворюватися зона підвищеного тиску, за нею - вітрова тінь, а між близько розташованими фасадами потік може прискорюватися. Такі ефекти не завжди видно на звичайній карті, але вони добре пояснюють, чому два сусідні місця в одному кварталі можуть відчуватися по-різному для пішохода.

У розробленій системі ці явища подані через поєднання швидких евристичних правил і чисельної моделі. Евристична частина потрібна для оперативного відгуку інтерфейсу: вона дає змогу швидко побачити приблизну зміну поля після зміни сценарію. Чисельний підхід, зокрема Lattice-Boltzmann D2Q9, використовується як більш детальний варіант моделювання, який краще показує розподіл потоку навколо перешкод.

Вибір такого поєднання пов'язаний із обмеженнями браузерного застосунку. Розрахунок не повинен блокувати інтерфейс, бо користувач працює з картою інтерактивно: змінює місто, напрямок вітру, додає об'єкти, порівнює

базовий і сценарний стани. Якщо кожна дія призводитиме до довгого очікування, практична цінність цифрового двійника різко зменшиться.

Тому під час проєктування довелось уважно ставитися до розміру розрахункової сітки, кількості ітерацій, способу кешування та формату виведення результату. Надто дрібна сітка підвищує навантаження на браузер, а надто груба може приховати локальні зміни біля будівель. У CityTwin обрано компромісний підхід: модель має залишатися достатньо швидкою для демонстрації, але при цьому не втрачати основну просторову логіку процесу.

Компроміс між швидкістю і точністю є одним із ключових рішень цієї роботи. Якщо зробити модель надто простою, вона швидко працюватиме, але не покаже важливих локальних ефектів біля забудови. Якщо зробити її надто складною, користувач втратить можливість швидко порівнювати сценарії. У CityTwin цей баланс вирішується через поєднання евристичного поля, яке дає швидку реакцію, та LBM-підходу, який демонструє більш фізично осмислену поведінку потоку.

Для бакалаврського проєкту такий вибір є виправданим, бо він показує не лише браузерну карту, а й інженерну логіку прийняття рішень. Система не приховує свої обмеження, але дає користувачу зрозумілий результат: можна побачити, як зміна форми забудови впливає на просторовий розподіл вітру, і вирішити, чи потребує цей варіант детальнішого аналізу.

Окреме значення має підготовка перешкод для розрахунку. Будівлі, отримані з карти, потрібно перетворити у форму, придатну для моделі: визначити їх положення у межах сітки, врахувати приблизну висоту і відокремити вільний простір від забудови. Після цього система може оцінювати, як напрямок вітру взаємодіє з геометрією кварталу.

Візуалізація результату у цій роботі не є другорядною частиною. Користувачеві недостатньо отримати масив чисел або середнє значення швидкості. Набагато корисніше побачити на карті, де саме з'являються ділянки підвищеної швидкості, де формується вітрова тінь і як змінюється ситуація після

сценарного втручання. Тому розрахункове поле подається у вигляді зрозумілого оверлею.

На рівні реалізації описані рішення пов'язані з файлами src/analysis/windField.ts, src/simulation/lbm.ts і src/simulation/lbmWorker.ts. Перший файл відповідає за формування швидкого поля вітру, другий містить чисельну LBM-логіку, а фоновий потік допомагає виконувати важчі обчислення окремо від основного потоку інтерфейсу. Такий поділ робить систему придатнішою для подальшого розширення і тестування.

Отже, моделювання вітрового комфорту в CityTwin побудоване як практичний компроміс між фізичною правдоподібністю, швидкістю роботи та зрозумілістю результату для користувача. Система не замінює професійний інженерний розрахунок, але дозволяє швидко оцінити, як забудова або сценарні зміни можуть вплинути на рух повітря у кварталі. Для цифрового двійника саме така попередня оцінка є корисною, бо вона допомагає вибрати варіанти, які варто аналізувати детальніше.

1.4 Підходи до оцінювання шумового навантаження

Шумове навантаження в CityTwin розглядається як просторовий вплив дорожньої мережі на вибраний фрагмент міста. Для дипломного прототипу важливо не відтворити повний акустичний розрахунок, а показати, як джерела шуму, забудова і сценарні зміни можуть впливати на загальну картину середовища [18-19].

Дороги різних класів у моделі мають різну відносну інтенсивність. Магістральна вулиця не повинна впливати на карту так само, як невеликий проїзд, тому тип дороги використовується як один із вхідних параметрів. Далі вплив поступово зменшується з відстанню, що відповідає логіці інженерних наближень для відкритого простору.

Будівлі враховуються як перешкоди, які можуть послаблювати поширення шуму. У спрощеній моделі вони не описують усю дифракцію, відбиття від фасадів і матеріали поверхонь, але дозволяють показати різницю між відкритою ділянкою та кварталом із щільною забудовою.

У реалізації ця логіка пов'язана з файлами src/analysis/noiseAnalysis.ts, src/analysis/profiles.ts і src/analysis/renderOverlay.ts. Перший файл формує розрахункове поле, другий задає профілі параметрів, а третій допомагає подати результат у вигляді шару на карті.

На практиці шумовий режим потрібний для порівняння сценаріїв. Якщо користувач додає парк або будівлю, система показує, де відносне навантаження може зменшитися, а де ситуація залишається проблемною. Такий результат зручніше читати як карту зон, а не як одну середню величину.

Обмеження моделі залишаються явними: вона не є сертифікованим акустичним інструментом і не замінює вимірювання на місцевості. Її завдання інше - швидко вказати ділянки, які потребують уважнішої перевірки під час подальшого аналізу.

Отже, шумова модель у роботі виконує роль попереднього індикатора. Вона поєднує класи доріг, просторові перешкоди та візуальне подання, щоб користувач міг побачити не тільки наявність джерела шуму, а й орієнтовний розподіл його впливу на території.

Шум у місті є одним із тих факторів, які люди відчувають щодня, навіть якщо не завжди формулюють це як технічну проблему. Постійний транспортний шум, відбиття звуку між фасадами, відкриті дороги біля житлових зон або щільна забудова можуть помітно впливати на комфорт середовища. Тому під час аналізу міського простору важливо дивитися не лише на геометрію забудови, а й на те, як у цьому просторі може поширюватися шумове навантаження [18, 21].

Повноцінна акустична модель потребує великої кількості даних. Потрібно враховувати інтенсивність транспортного потоку, тип покриття дороги, висоту й матеріали будівель, відбиття від фасадів, поглинання зеленими зонами, рельєф і

навіть час доби. Такий підхід дає точніший результат, але він складний для швидкого інтерактивного прототипу. Якщо вимагати від користувача занадто багато вхідних параметрів, система стане менш зручною і втратить свою демонстраційну цінність.

У CityTwin шумова модель використовується як спосіб попередньо оцінити просторову картину. Вона має показати, де потенційно може зберігатися більше шумове навантаження, а де вплив може бути меншим. Для користувача важливо не тільки побачити числове значення, а й зрозуміти розподіл на карті: які ділянки виглядають проблемними, як змінюється ситуація після додавання об’єкта, чи дає сценарій помітне покращення.

Такий підхід робить модель більш практичною для навчальної та попередньо-аналітичної задачі. Вона не замінює професійний акустичний розрахунок, але допомагає поставити правильні питання. Якщо візуалізація показує, що певна ділянка залишається шумонавантаженою, це означає, що її варто перевірити детальніше. Для цифрового двійника на рівні прототипу саме така швидка орієнтація є дуже важливою.

1.5 Моделювання теплового стресу та ефекту міського острова тепла

Тепловий стрес у міському середовищі залежить від того, які поверхні переважають у кварталі, наскільки щільною є забудова, де розташовані зелені зони й вода, а також яка поточна температура повітря. У CityTwin ці чинники [20, 28] подані у спрощеній формі, достатній для порівняння сценаріїв у браузерному прототипі.

Будівлі та дороги в моделі збільшують ризик локального перегрівання, бо вони відповідають твердим поверхням, що накопичують тепло. Зелені зони і вода, навпаки, зменшують оцінку теплового навантаження. Такий поділ не описує всю міську мікрокліматологію, але добре показує напрям зміни після сценарного втручання.

Окремо враховується щільність забудови. У щільних кварталах повітрообмін слабшає, а відкритих поверхонь для охолодження стає менше. Для швидкого обчислення локальної щільності використано Summed Area Tables, тому застосунок може перераховувати сценарій без помітної затримки для користувача.

Поточні погодні дані надходять через Open-Meteo і впливають на ампліфікацію теплового режиму. Якщо температура вища, одна й та сама структура поверхонь має сильніший ефект. Це робить модель чутливішою до реального контексту, хоча вона все одно залишається попередньою оцінкою [5,20].

На рівні коду теплова логіка зосереджена у src/analysis/heatAnalysis.ts, а зовнішні погодні параметри отримуються через src/api/weatherApi.ts і допоміжні структури з src/data/weather.ts. Такий поділ дозволяє окремо перевіряти джерело даних і саму модель розрахунку.

Результат теплового аналізу слід читати як порівняльну карту. Вона показує, де після додавання забудови або озеленення ситуація може стати гіршою чи кращою, але не підміняє повний кліматичний розрахунок із добовою інерцією матеріалів, радіаційним балансом і польовою валідацією.

Таким чином, теплова модель допомагає оцінити просторові наслідки сценарію на рівні прототипу. Її сильна сторона полягає у швидкому порівнянні варіантів, а не в претензії на абсолютну температуру кожної точки міського простору.

Тепловий стан міського середовища також не можна описати одним показником. На нього впливають матеріали поверхонь, щільність забудови, наявність зелених зон, затінення, температура повітря і рух вітру. У прототипі ці фактори подано у спрощеному вигляді, щоб система залишалася зрозумілою та швидкою. Для CityTwin важливішим є не абсолютне значення кожного параметра, а можливість побачити, як зміна сценарію впливає на загальну картину теплового навантаження.

Ефект міського острова тепла особливо помітний у щільній забудові, де багато твердих поверхонь накопичують тепло протягом дня і повільно віддають його ввечері. Асфальт, бетон, фасади будівель і недостатня кількість зелених насаджень можуть створювати умови, за яких температура відкритих просторів відчувається значно вищою. Для мешканців це проявляється не як абстрактний кліматичний показник, а як реальний дискомфорт під час перебування на вулиці.

У цифровому двійнику важливо показати хоча б загальний зв’язок між просторовою структурою і тепловим станом території. Якщо в сценарії збільшується частка забудови або зменшується кількість зелених зон, користувач повинен побачити, що це може вплинути на теплове навантаження. Навпаки, додавання озеленення або відкритих провітрюваних ділянок може зробити сценарій більш сприятливим.

Для CityTwin така модель має попередній характер. Вона не враховує всі параметри міського мікроклімату, але дозволяє порівнювати варіанти між собою. У межах дипломної роботи цього достатньо, оскільки головна мета полягає в демонстрації логіки цифрового двійника і взаємозв’язку між просторовими рішеннями та комфортом середовища.

1.6 Постановка задачі розроблення системи CityTwin

Задача розроблення CityTwin полягає у створенні браузерного прототипу цифрового двійника, який поєднує карту, відкриті геодані, погодні параметри, сценарне редагування і візуальні результати аналізу. Така постановка відповідає формату дипломної роботи: система має бути достатньо конкретною для демонстрації і водночас реалістичною за обсягом реалізації [1-14, 29-30, 41].

Основна вимога до прототипу - швидкий перехід від перегляду карти до порівняння сценарію. Користувач повинен обрати місто, запустити режим аналізу, додати умовний об'єкт і побачити, як змінюється просторовий шар.

Якщо для цього потрібна складна підготовка даних або окремий серверний модуль, застосунок втрачає демонстраційну простоту.

Система працює з джерелами, що формуються під час виконання: карта надає просторові об'єкти, Open-Meteo - погодні параметри, а локальний стан застосунку зберігає вибраний сценарій. Завдяки цьому CityTwin не потребує власної міської бази даних для кожної локації, хоча якість результату залежить від повноти відкритих джерел.

Функціонально прототип має підтримувати режими вітру, шуму і теплового стресу, сценарне додавання будівель або зелених зон, 2D/3D-перегляд, багатомовний інтерфейс і короткий числовий підсумок. Ці можливості разом утворюють робочий сценарій, а не набір окремих демонстраційних екранів.

Нефункціональні вимоги пов'язані з продуктивністю та надійністю. Розрахунки мають виконуватися без блокування інтерфейсу, зовнішні запити не повинні ламати сторінку, а ключова логіка має перевірятися тестами. Саме тому в проєкті використано обмежені розміри сіток, фоновий вебпрацівник для важчих обчислень і перевірку типових користувацьких сценаріїв.

У коді постановка задачі відображена у README.md, src/hooks/useAppReducer.ts і src/components/ScenarioPanel.tsx. README описує запуск і межі прототипу, reducer керує станом сценарію, а ScenarioPanel дає користувачу інструмент для створення міських втручань.

Отже, поставлена задача не зводиться до створення карти або окремої моделі. Йдеться про програмний ланцюг, у якому дані, сценарій, розрахунок і візуалізація працюють в одному середовищі та дозволяють отримати попередню оцінку змін у міському просторі.

Отже, CityTwin доцільно використовувати для попереднього оцінювання міських змін. Основою прототипу є відкриті геодані, погодні дані та спрощені моделі аналізу.

Таблиця 1.2 – Основні вимоги до кіберфізичної системи

Група вимог	Зміст вимоги	Реалізація у проєкті
Функціональні	Вибір міста, режиму аналізу, погоди, сценарних об'єктів	Header, NavRail, AnalysisPanel, ScenarioPanel
Аналітичні	Обчислення вітру, шуму, теплового стресу і KPI	analysis/*, simulation/lbm.ts, overlayManager
Картографічні	2.5D-карта, 3D-будівлі, супутниковий шар	MapScene, MapLibre GL
Надійність	Обробка помилок програмного інтерфейсу API, localStorage і компонентів інтерфейсу користувача	httpClient, safeStorage, ErrorBoundary
Тестованість	Модульні, інтеграційні та наскрізні перевірки	Vitest, Playwright, Testing Library

2 ПРОЄКТУВАННЯ КІБЕРФІЗИЧНОЇ СИСТЕМИ «ЦИФРОВИЙ ДВІЙНИК МІСТА»

2.1 Вимоги до системи та сценарії використання

Вимоги до CityTwin сформовано від очікуваного користувацького сценарію. Користувач відкриває міський фрагмент, обирає режим аналізу, змінює параметри або додає сценарний об'єкт і отримує оновлений результат на карті. Тому вимоги стосуються не тільки окремих кнопок, а всієї послідовності роботи з цифровим двійником [8-14].

До функціональних вимог належать вибір міста, перемикання режимів вітру, шуму і тепла, вибір мови, 2D/3D-перегляд, супутникова підкладка, запуск сценарного редактора та відображення числових показників. Кожна з цих можливостей потрібна для того, щоб користувач міг не просто подивитися карту, а перевірити певне припущення.

Сценарний редактор має підтримувати додавання будівель і парків із параметрами ширини, глибини, висоти, повороту та кольору. Для користувача це зрозуміла дія; для системи - зміна геометрії, яка повинна пройти перевірку колізій і бути врахована під час повторного розрахунку.

Окрема вимога стосується захисту від очевидно некоректних дій. Об'єкт не повинен розміщуватися на воді, поверх дороги або всередині наявної забудови без перевірки. Така перевірка не робить систему нормативним планувальним інструментом, але запобігає демонстраційно хибним сценаріям.

Нефункціональні вимоги охоплюють швидкодію, зрозумілий інтерфейс, стійкість до помилок програмного інтерфейсу API та повторюваність тестів. Якщо карта довго реагує на зміну режиму або помилка погодного сервісу зупиняє весь застосунок, цифровий двійник стає незручним навіть за правильної математичної логіки.

На рівні реалізації ці вимоги пов'язані з src/App.tsx, src/components/Header.tsx, src/components/NavRail.tsx і

src/components/ScenarioPanel.tsx. Header відповідає за загальні параметри, NavRail - за режими, ScenarioPanel - за сценарні дії, а App координує основний стан застосунку.

У підсумку вимоги до системи можна сформулювати як вимоги до безперервного робочого потоку: карта надає просторову основу, інтерфейс приймає дію користувача, моделі перераховують показники, а результат повертається у вигляді зрозумілого шару та короткого числового підсумку.

2.2 Структурна схема кіберфізичної системи CityTwin

Для усунення зміщення акценту лише на програмну частину CityTwin у роботі система розглядається як кіберфізична система. У такій постановці браузерний застосунок є не самодостатнім продуктом, а кібернетичним рівнем, який приймає дані про фізичне міське середовище, виконує обчислення, формує візуальний результат і підтримує прийняття рішень. Фізичний рівень у цій системі представлений міською забудовою, дорожньою мережею, зеленими зонами, локальним вітром, температурою, вологістю та шумовим навантаженням.

Ключова ознака КФС полягає у наявності контуру зв'язку між фізичним і цифровим середовищем. Для CityTwin цей контур має такий вигляд: фізичне міське середовище спостерігається через давачі або відкриті джерела даних; edge-вузол приводить вимірювання до єдиного формату; мережевий шар передає телеметрію; кібернетичний двійник виконує аналіз; людино-машинний інтерфейс показує результат; користувач приймає планувальне рішення, яке може змінити реальний простір. Така структура відповідає загальній логіці кіберфізичних систем, де обчислення, комунікація, спостереження, фізичний процес і людина розглядаються як частини одного циклу [42].

У межах бакалаврської роботи реалізовано програмну частину цього контуру та спроєктовано апаратний рівень збору польових даних. У прототипі

польові вимірювання замінено відкритими джерелами Open-Meteo та OpenStreetMap/OpenFreeMap, але архітектура передбачає підключення давачів через edge-вузол. Це добре відображає межі реалізації: система не імітує наявність зібраного стенда, однак має проєктну структуру, у яку такий стенд може бути включений без зміни загальної логіки застосунку.

На рисунку 2.1 показано, що Open-Meteo та OpenStreetMap у прототипі виконують роль джерел спостереження, але не скасовують потребу в апаратному шарі. OpenStreetMap/OpenFreeMap надають просторову основу: контури будівель, дороги, водні об'єкти та зелені зони.

Open-Meteo надає погодні параметри, які у реальній експлуатації можуть уточнюватися локальними метеовузлами. Таким чином, прототип використовує відкриті джерела як заміщення польових вимірювань, а повна КФС-версія може приймати телеметрію з давачів.

Рисунок 2.1 – Структурна схема кіберфізичної системи CityTwin

Опис архітектури подано через окреме представлення КФС, тому воно не дублює програмну архітектуру. Програмна схема відповідає на питання, які модулі є у застосунку, тоді як схема КФС показує, як дані з фізичного середовища потрапляють у цифрову модель і як результат моделі повертається до людини у вигляді рішення. Такий поділ також відповідає підходу ISO/IEC/IEEE 42010: архітектурний опис складної системи доцільно подавати через кілька узгоджених представлень, а не через одну універсальну схему [43].

2.3 Апаратний рівень та давачі системи

Апаратний рівень CityTwin призначений для збору локальних параметрів міського середовища.

Його доцільно будувати як набір польових вузлів, розміщених у контрольних точках кварталу: біля дороги, у внутрішньому дворі, поруч із зеленими зонами або на відкритій ділянці.

Кожен вузол вимірює частину параметрів, передає їх до шлюзу або безпосередньо у мережу, після чого кібернетичний двійник використовує ці дані для уточнення розрахункових моделей.

Мінімальний набір давачів повинен покривати саме ті показники, які аналізує CityTwin [44-47]. Для вітрового режиму потрібні анемометр і флюгер: перший дає швидкість вітру, другий - напрямок.

Для теплового режиму потрібні температура, вологість і бажано тиск, тому доцільним є BME280 або SHT31 у ролі метеодатчика.

Для додаткової температури поверхні або захищеної точки можна використати DS18B20, підключений по 1-Wire. Для шумового поля застосовується шумовий датчик, SPL-модуль або цифровий MEMS-мікрофон INMP441, який передає дані по I2S.

Таблиця 2.1 – Апаратні компоненти польового рівня CityTwin

Компонент	Параметр	Підключення	Роль у КФС
Анемометр	Швидкість вітру	GPIO interrupt, імпульсний вхід	Уточнення вітрового поля та перевірка прогнозних даних
Флюгер	Напрямок вітру	ADC або цифровий інтерфейс	Визначення вхідного напрямку для моделі вітру
BME280 / SHT31	Температура, вологість, тиск	I2C	Базові метеопараметри для теплового режиму
DS18B20	Локальна температура	1-Wire, резистор 4.7 кОм	Контроль температури поверхні або контрольної точки
INMP441 / SPL-модуль	Шумовий рівень	I2S або ADC	Калібрування та перевірка шумового шару
ESP32 / Raspberry Pi	Периферійна обробка	Wi-Fi, LTE або LoRaWAN-шлюз	Фільтрація, нормалізація і передача телеметрії

Для бакалаврського прототипу достатньо спроєктувати один типовий вузол збору даних. У реальному місті таких вузлів може бути декілька, і вони мають різні пріоритети. Біля магістралі важливішим буде шумовий канал, у щільній забудові - напрямок і швидкість вітру, у зеленій зоні - температура та вологість.

Однак структура вузла залишається однаковою: давачі формують первинні вимірювання, периферійний контролер перевіряє їх діапазони, додає часову мітку та передає у систему.

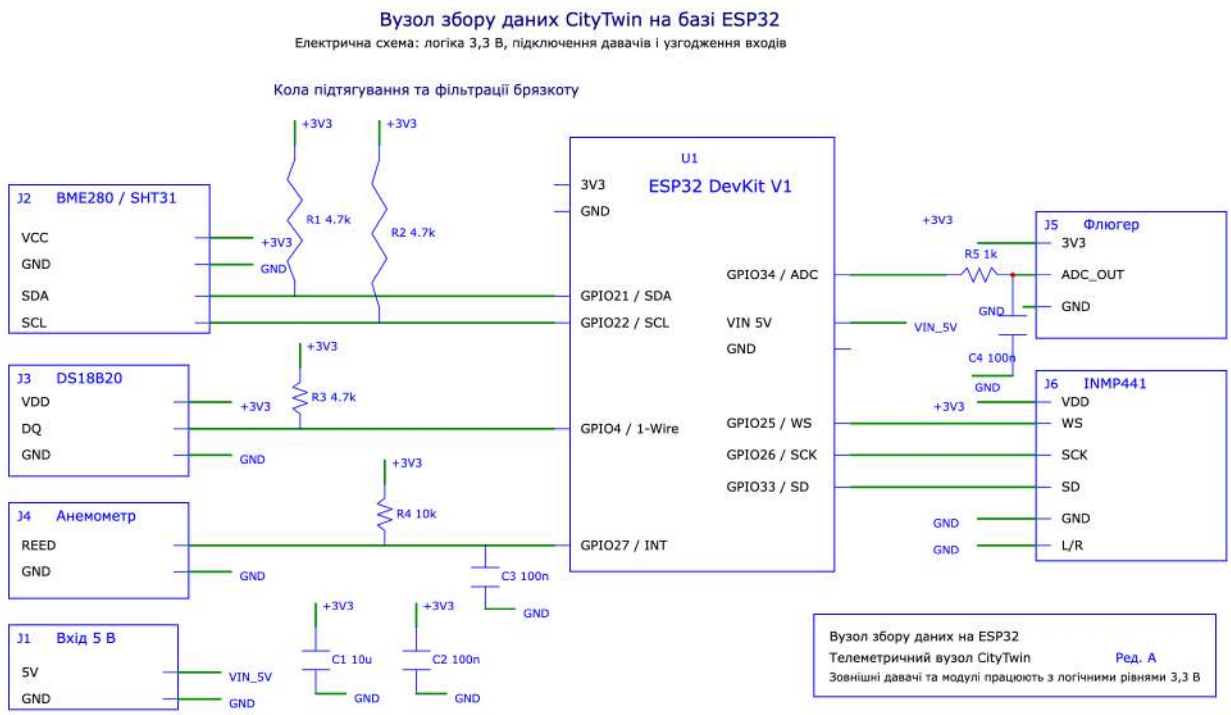


Рисунок 2.2 – Структурна електрична схема вузла збору міських параметрів

На рисунку 2.2 наведено не промислову друковану плату, а структурну електричну схему типового вузла. Плату розробника ESP32 DevKit вибрано як базовий периферійний контролер, оскільки він має вбудований Wi-Fi, достатню кількість GPIO, ADC, I2C та I2S, а також добре підходить для навчального прототипування [44]. BME280 або SHT31 підключається по I2C через SDA/SCL, DS18B20 - по 1-Wire з підтягувальним резистором 4.7 кОм, анемометр з reed switch - на переривання GPIO, флюгер - на ADC, а INMP441 - через I2S. Живлення вузла може надходити від 5 V USB або DC/DC-перетворювача, а логічні рівні залишаються 3.3 V.

Така схема важлива не стільки як інструкція з монтажу, скільки як доказ програмно-апаратної природи системи. Вона показує, звідки можуть надходити дані, які у поточному прототипі беруться з програмного інтерфейсу API. Якщо в майбутньому буде зібрано фізичний стенд, програмний рівень не потрібно переписувати повністю: достатньо реалізувати адаптер телеметрії, який перетворює покази давачів у той самий формат, що використовується в аналітичних модулях.

2.4 Телеметричні потоки та заміщення польових вимірювань у прототипі

Програмно-апаратна інтеграція потребує єдиного формату телеметрії. Без цього кожен давач або джерело даних довелося б підключати окремо до моделей, що швидко ускладнило б супровід системи. Тому для CityTwin доцільно виділити проміжний телеметричний пакет, у якому є ідентифікатор вузла, координати, час вимірювання, метеопараметри, шумовий рівень і ознака джерела даних.

Таблиця 2.2 – Формат телеметричного пакета CityTwin

Поле	Тип	Приклад	Призначення
nodeId	string	khm-node-01	Ідентифікація польового вузла або віртуального джерела
timestamp	ISO 8601	2026-05-26T12:00:00Z	Синхронізація вимірювань і кешу
lat, lon	number	49.422, 26.987	Прив'язка вимірювання до карти
windSpeed, windDirection	number	4.8; 230	Початкові дані для вітрової моделі
temperature, humidity, pressure	number	28.4; 51; 1009	Параметри теплового режиму
noiseDbA	number	62.5	Калібрування шумового поля
джерело	enum	сенсорне джерело | open-meteo | ручний режим	Відокремлення реальних вимірювань від заміщення у прототипі

Передача такого пакета може виконуватися через HTTP або MQTT [48-49]. HTTP простіший для браузерної та серверної інтеграції, оскільки телеметрію можна надсилати як POST-запит до адаптера програмного інтерфейсу API. MQTT краще підходить для великої кількості вузлів, бо використовує модель publish/subscribe і дозволяє розділяти канали за містом, типом давача або конкретним вузлом. Якщо вузли розміщені на великій території та не мають стабільного Wi-Fi, у мережевому шарі можна використати віддалений радіоканал або LTE-шлюз. Для стандартизованого опису сенсорних спостережень може застосовуватися OGC SensorThings API [50].

У поточному CityTwin роль телеметричного джерела виконує Open-Meteo. Це означає, що застосунок уже має логіку отримання погодних параметрів, обробку помилок, тайм-аут і повтори запиту. Для перетворення прототипу на повну КФС потрібно не замінити цю логіку, а додати альтернативний адаптер, який прийматиме ті самі поля від польового вузла. Якщо вимірювання доступні, вони мають пріоритет. Якщо ні, система може повернутися до Open-Meteo як резервного джерела.

OpenStreetMap і OpenFreeMap також слід трактувати як частину шару спостереження, але іншого типу. Вони не вимірюють температуру або шум, зате дають геометрію фізичного середовища. Для КФС це важливо, бо польові давачі без просторової моделі не пояснюють, чому певна зона має підвищений шум або змінений вітровий режим. Поєднання геометрії, метеоданих і сценарних дій користувача формує повний інформаційний контур CityTwin.

2.5 Загальна архітектура програмного комплексу

Архітектура CityTwin побудована як React SPA, у якій карта, стан застосунку, витяг геометрії, аналітичні модулі та візуалізація мають окремі ролі [6-12]. Такий поділ потрібний не для формальності, а для того, щоб зміна одного елемента не вимагала переписування всієї системи.

На верхньому рівні користувач працює з картою, навігаційною панеллю, заголовком, панеллю аналізу, сценарною панеллю та нижньою легендою. Ці елементи не виконують однакову роботу: одні задають параметри, інші показують простір, а ще інші пояснюють результат.

Центральний стан зберігає вибране місто, активний режим, погодні параметри, сценарні втручання і службові перемикачі. Це дозволяє передавати зміни між компонентами без випадкового дублювання логіки. Наприклад, перемикання режиму аналізу не повинно скидати місто або втрачати додані сценарні об'єкти.

MapScene виконує роль просторового ядра. Він показує карту, отримує з MapLibre геометрію під час виконання, передає підготовлені об'єкти до моделей і накладає результат назад на карту. Саме тут поєднуються візуальна частина і дані, потрібні для розрахунку.

Аналітичні модулі працюють із типовими структурами AnalysisGrid, BuildingFootprint та пов'язаними форматами. Вони не повинні знати, як виглядає кнопка або панель у React-компоненті; їхнє завдання - отримати підготовлені дані й повернути поле значень.

LBM-обчислення винесено у фоновий вебпрацівник, щоб чисельна симуляція не блокувала інтерфейс [31]. Це практичне архітектурне рішення: користувач може залишатися в тому самому сценарії, поки важча частина розрахунку виконується окремо від основного потоку інтерфейсу.

Файли src/App.tsx, src/components/MapScene.tsx і src/components/overlayManager.ts показують основний каркас цієї архітектури. App координує стан, MapScene працює з картою і просторовими даними, а overlayManager відповідає за перетворення результатів у шари, які користувач бачить на екрані. Схему наведено на рис. 2.3, компоненти - у табл. 2.3.

Рисунок 2.3 – Архітектура програмного комплексу CityTwin

Таблиця 2.3 – Компоненти програмної архітектури

Шар	Модулі	Призначення
Інтерфейс користувача	Header, NavRail, AnalysisPanel, ScenarioPanel, BottomBar	Керування режимами, параметрами, сценаріями та легендою
Стан	useAppReducer, useWeather, usePlacement, useToasts	Централізований стан і побічні ефекти
Дані	featureExtractor, weatherApi, cityPresets	Геометрія, погода, конфігурації міст
Аналіз	windField, lbm, noiseAnalysis, heatAnalysis	Розрахунок полів цифрового двійника

Кінець таблиці 2.3

Візуалізація	MapScene, renderOverlay, AnimatedOverlay, overlayManager	MapLibre, Canvas/WebGL і GeoJSON-шари

CityTwin побудовано так, щоб кожна частина системи виконувала свою зрозумілу роль. Цифровий двійник поєднує карту, дані, розрахункові модулі, сценарії користувача та інтерфейс.

Якщо вся ця логіка була б змішана в одному місці, застосунок було б важко підтримувати, тестувати й розвивати. Тому архітектура поділена на окремі функціональні частини.

Карта є основною точкою входу для користувача. Через неї він бачить міський фрагмент, орієнтується в забудові та взаємодіє зі сценарієм. Дані про будівлі, зелені зони, відкриті ділянки й погодні умови готуються окремо, щоб їх можна було передати до модулів аналізу. Такий поділ дозволяє не змішувати відображення карти з логікою розрахунків.

Аналітичні модулі відповідають за оцінювання вітру, шумового навантаження та теплового стресу. Кожен із цих напрямів має власну логіку, тому їх доцільно розглядати як окремі частини системи. У майбутньому це дасть змогу вдосконалити, наприклад, тільки модель вітру або тільки тепловий аналіз без повної перебудови застосунку.

Інтерфейс об’єднує результати в зрозумілу для користувача форму. Для CityTwin важливо не просто виконати розрахунок, а показати його так, щоб людина могла порівняти сценарії й зробити попередній висновок. Саме тому карта, дані, моделі та візуальні шари працюють як єдина система, а не як набір непов’язаних елементів.

2.6 Організація даних, стану та інтеграційних потоків

Дані в CityTwin не зберігаються у класичній базі даних. Вони формуються під час роботи застосунку: з карти береться геометрія, з Open-Meteo - погодні параметри, зі стану React - сценарні об'єкти та вибрані режими [2-5]. Через це важливо описати не тільки джерела даних, а й шлях їхнього перетворення.

Початкові координати і масштаби міст задаються у CITY_PRESETS. Це спрощує запуск роботи: користувач не вводить межі вручну, а обирає місто зі списку. Далі карта завантажує відповідну область, і з неї можна витягувати будівлі, дороги, зелені зони та водні об'єкти.

Погодні параметри отримуються через useWeather і weatherApi. Запити проходять через спільний httpClient із тайм-аутом та повторними спробами, тому тимчасова проблема програмного інтерфейсу API не повинна руйнувати весь інтерфейс. Якщо дані недоступні, система має поводитися передбачувано і зберігати зрозумілий стан [41].

featureExtractor перетворює об'єкти MapLibre на внутрішні контури, полігональні кільця і лінії доріг. Цей етап важливий, бо моделі не працюють із сирими об'єктами карти. Їм потрібна підготовлена геометрія, прив'язана до меж аналізу та формату розрахункової сітки [37].

Сценарні втручання зберігаються у React-стані. Для кожного об'єкта важливі тип, координати, footprint, висота, колір і пов'язані параметри. Коли користувач змінює сценарій, саме ці дані передаються до моделей разом із базовою геометрією міста.

safeStorage ізолює роботу з localStorage, щоб особливості браузерного середовища не ламали застосунок. Це невеликий, але практично важливий елемент: збереження параметрів має допомагати користувачу, а не створювати додаткове джерело помилок.

У підсумку потік даних у CityTwin проходить зрозумілу послідовність: джерело, нормалізація, стан, модель, візуалізація. Саме така послідовність дає

змогу пояснити, звідки взявся результат і чому після зміни сценарію оновлюється конкретний шар на карті.

У CityTwin робота з даними починається з карти, але карта не є лише фоном для відображення міста. Вона є точкою входу до всієї системи. Користувач бачить територію, вибирає сценарій, додає або змінює об’єкти, а система перетворює ці дії на дані, які можна використати для аналізу. Саме так звичайне зображення міського простору стає частиною цифрового двійника.

До моделі потрапляють ті об’єкти, які мають значення для оцінки середовища: будівлі, відкриті ділянки, зелені зони та інші просторові елементи. Окремо враховуються погодні дані, тому що вони впливають на результати розрахунків. Напрям і швидкість вітру важливі для вітрового аналізу, а температурні умови - для оцінювання теплового навантаження. Після підготовки ці дані передаються до аналітичних модулів.

Результат роботи модулів повертається користувачу у вигляді візуальних шарів. Це важливо, тому що людині легше сприймати не набір чисел, а картину на карті. Користувач бачить, де ситуація виглядає кращою, а де можуть бути проблемні ділянки. Завдяки цьому цифровий двійник стає не просто розрахунковою системою, а зручним засобом для інтерпретації міських змін.

Окрему роль відіграє стан застосунку. У ньому зберігається активний режим аналізу, поточний сценарій, параметри карти, додані користувачем об’єкти і результати розрахунків. Це дозволяє системі поводитися послідовно. Якщо користувач змінює сценарій, застосунок оновлює потрібні результати, але не втрачає загальний контекст роботи.

У підсумку користувач працює не з окремими ізольованими налаштуваннями, а з єдиною моделлю міського фрагмента.

2.7 Проєктування моделей вітру, шуму та теплового стресу

Проєктування моделей у CityTwin зосереджене на трьох показниках: вітровому режимі, шумовому полі та тепловому стресі. Вони мають різну фізичну природу, але в застосунку подаються через спільний принцип: підготовлена геометрія переноситься на сітку, модель обчислює значення, а карта показує результат користувачу [15-21, 26, 28].

Вітрова модель формує поля ux, uy, швидкості, турбулентності та перешкод. Евристичний режим потрібний для швидкої реакції, а LBM D2Q9 додає наочнішу симуляцію обтікання перешкод. Обидва підходи працюють із тією самою просторовою основою, тому їх можна порівнювати в межах одного сценарію.

Шумова модель будує roadGrid і obstacleGrid, після чого оцінює поширення впливу від дорожньої мережі. Тут важливі класи доріг, відстань до джерела та екранування будівлями. Модель не підміняє професійний акустичний розрахунок, але дає зрозумілий просторовий індикатор.

Теплова модель растеризує будівлі, дороги, зелені зони та воду, а потім оцінює щільність і охолоджувальний вплив. Використання Summed Area Tables допомагає швидко рахувати локальні характеристики навколо клітинки, що важливо для інтерактивного сценарію.

Файли src/analysis/windField.ts, src/analysis/noiseAnalysis.ts, src/analysis/heatAnalysis.ts і src/analysis/profiles.ts відповідають за ядро цих моделей. Профілі параметрів винесені окремо, тому їх можна коригувати без зміни компонентів інтерфейсу.

Результати моделей нормуються до діапазону, який зручно показувати на карті, або перетворюються на підписані значення для KPI. Через це користувач бачить не сирі масиви даних, а порівняльну картину: де сценарій погіршує умови, а де може давати позитивний ефект.

Отже, моделі аналізу спроєктовано як взаємопов'язані, але окремо перевірювані частини системи. Такий підхід дає змогу розвивати кожний режим незалежно і водночас зберігати спільний користувацький сценарій у CityTwin.

Таблиця 2.4 – Параметри моделей аналізу

Модель	Вхідні дані	Розрахункова сітка	Результат
Вітер	Будівлі, зелень, вітер, висота зрізу	до 200x200	ux, uy, швидкість, турбулентність
LBM D2Q9	Полігональні перешкоди, напрямок вітру	256x256 у реалізації	поле швидкості з перешкодами
Шум	Дороги, класи доріг, будівлі	200x200	відносне шумове поле
Тепло	Будівлі, дороги, зелень, вода, температура	200x200	тепловий стрес / UHI

Під час проєктування моделей важливо було зберегти однакову логіку для всіх трьох напрямів аналізу. Вітер, шум і тепловий стрес мають різну фізичну природу, але в межах CityTwin вони повинні працювати для користувача за зрозумілим принципом: система отримує просторові дані, враховує параметри сценарію, виконує розрахунок і повертає результат у вигляді шару на карті.

Такий підхід спрощує взаємодію із застосунком. Користувачу не потрібно окремо розбиратися з різними технічними форматами для кожної моделі. Він працює з єдиним сценарієм, а система вже сама передає потрібні дані до відповідного модуля аналізу. Це робить цифровий двійник більш цілісним і зручним для демонстрації.

Для вітрової моделі основним є вплив забудови на рух повітря. Для шумової моделі важливо показати просторовий розподіл навантаження. Для теплового аналізу головним є порівняння ділянок, які можуть бути більш або менш вразливими до перегрівання. Усі ці результати мають різний зміст, але вони подаються користувачу в одній логіці візуального порівняння.

Саме тому в проєкті не робиться акцент на складних числових таблицях. Для попереднього аналізу важливіше швидко побачити просторову картину і зрозуміти, як сценарій змінює середовище. Такий спосіб подання краще відповідає задачі браузерного цифрового двійника, орієнтованого на інтерактивну роботу з картою.

2.8 Проєктування сценарного редактора міських втручань

Сценарний редактор перетворює CityTwin із пасивної карти на інструмент порівняння міських втручань [6-10]. Користувач може додати умовну будівлю або парк, змінити параметри об'єкта і побачити, як це впливає на аналіз території.

Для кожного втручання задаються тип, геометрія, висота, поворот та інші службові параметри. Ці дані мають бути достатньо простими для користувача і водночас придатними для розрахункових модулів, які працюють із контурами будівель та сіткою аналізу.

placementEngine формує локальну геометрію перешкод із наявних будівель, води, доріг, залізниці та вже доданих сценарних об'єктів. Після цього він може перевірити, чи не потрапляє новий об'єкт у недопустиме місце.

Перевірка колізій виконується через полігональні перетини та буфери доріг у метрах. Це дає змогу відсіяти очевидно неправильні сценарії, наприклад розміщення будівлі на воді або поверх транспортної осі.

auto-rotate допомагає орієнтувати об'єкт за найближчою дорогою або довгим ребром сусідньої геометрії. auto-shift і auto-fit шукають допустиме

положення чи масштаб footprint без ручного підбору кожної координати. Такі функції роблять редактор практичнішим для демонстрації.

Після додавання втручання MapScene перераховує аналіз і формує порівняльні KPI. Для користувача це означає короткий цикл роботи: додати об'єкт, побачити зміну шару, порівняти базовий і сценарний стан.

Основні частини редактора розміщені у src/components/placementEngine.ts, src/hooks/usePlacement.ts і src/components/ScenarioPanel.tsx. Перший файл відповідає за геометричну логіку, другий - за поведінку розміщення, третій - за взаємодію користувача з панеллю сценарію.

2.9 Проєктування інтерфейсу користувача та багатомовності

Підрозділ присвячено практичній організації інтерфейсу CityTwin, а не загальному опису вікон застосунку. Екран побудовано навколо карти: користувач спочатку обирає місто й режим аналізу, після чого отримує доступ до параметрів сценарію, легенди та числового підсумку. Така структура зменшує кількість переходів між екранами і дозволяє порівнювати результати без втрати контексту [8-11].

Верхня панель Header об'єднує назву системи, вибір міста, перемикач 2D/3D та вибір мови. Це службовий рівень інтерфейсу: він задає початкові умови для всієї карти, але не змішується з налаштуваннями конкретної моделі. Завдяки цьому зміна локалізації або перспективи не потребує перезапуску розрахунку.

NavRail винесено в окрему вертикальну навігацію, бо режими вітру, шуму і тепла мають різні моделі, але однаковий спосіб запуску. Користувач бачить активний режим і швидко переходить між видами аналізу; MapScene при цьому передає до відповідного модуля ті самі межі карти та підготовлені просторові об'єкти.

AnalysisPanel виконує роль робочої панелі параметрів. У ній зібрано швидкість і напрям вітру, висоту зрізу, перемикачі шарів, температуру та

показники результату. У коді це пов'язано з компонентом src/components/AnalysisPanel.tsx, який не виконує сам розрахунок, а передає параметри до стану застосунку і відображає вже підготовлений підсумок.

ScenarioPanel відповідає за сценарні втручання: додавання будівлі або парку, зміну геометрії, висоти, повороту та перегляд списку створених об'єктів. Для користувача це окремий етап роботи: спочатку формується зміна на карті, потім система порівнює базовий і сценарний стан.

BottomBar використано для легенди та короткого пояснення активного шару. Це потрібно тому, що кольори для вітру, шуму й тепла мають різний зміст. Легенда допомагає читати карту без переходу до додаткових довідкових сторінок, а числовий блок показує, чи змінився середній або піковий показник.

Багатомовність реалізовано через i18next і файл src/i18n/resources.ts. Тексти інтерфейсу винесено з компонентів, тому українська, англійська та іспанська версії використовують одну й ту саму логіку екрана. Це зменшує дублювання коду і робить демонстрацію системи зрозумілішою для різної аудиторії.

Отже, інтерфейс CityTwin побудовано як робоче середовище для послідовного сценарію: вибір міста, вибір моделі, налаштування параметрів, додавання втручання і перегляд результату на карті. Саме така організація показує, що інтерфейс є частиною розрахункового процесу, а не лише оболонкою для візуалізації.

2.10 Обмеження моделі, надійність та вимоги до експлуатації

У цьому підрозділі визначено межі використання CityTwin та вимоги до стабільної роботи прототипу. Система не замінює сертифіковані CFD, акустичні або кліматичні комплекси; її призначення полягає у попередньому порівнянні сценаріїв на основі відкритих даних [16-21, 26-28].

Перше обмеження пов'язане з OpenStreetMap і OpenFreeMap. У відкритих геоданих можуть бути відсутні висоти будівель, точні контури зелених зон або класи окремих доріг. У таких випадках застосунок використовує евристичні припущення, тому результат потрібно читати як орієнтовний просторовий сигнал.

Друге обмеження стосується моделей. Вітровий режим поєднує евристичне поле і двовимірну LBM D2Q9-симуляцію, шумова модель працює з відносним дорожнім навантаженням, а теплова модель оцінює індекс перегрівання за морфологією поверхонь. Такі моделі достатні для навчального і демонстраційного прототипу, але не описують усі фізичні процеси реального міста.

Третє обмеження випливає з браузерного виконання. Розрахункова сітка, кількість об'єктів і частота оновлення шару мають залишатися такими, щоб інтерфейс не блокувався. Через це в окремих режимах використано обмеження 200x200 або 256x256 клітинок, кешування підготовлених даних і виконання важчих обчислень у фоновому вебпрацівнику.

Надійність реалізованого кібернетичного рівня підтримується кількома механізмами: ErrorBoundary перехоплює помилки React-компонентів, HTTP-клієнт використовує тайм-аут і повторні спроби, доступ до localStorage перевіряється перед використанням [34], а тести контролюють базові сценарії роботи. Це зменшує ризик, що збій програмного інтерфейсу API або некоректні дані повністю зупинять демонстрацію.

Вимоги до експлуатації також включають зрозуміле повідомлення про характер результату. Користувач повинен бачити, що шар на карті є попередньою аналітичною оцінкою. Якщо йдеться про реальне будівництво або нормативний висновок, результати CityTwin потрібно перевіряти польовими вимірюваннями чи спеціалізованими інженерними моделями.

На рівні коду ці обмеження відображено у README.md, src/components/ErrorBoundary.tsx та src/api/httpClient.ts. README пояснює

умови запуску і межі прототипу, ErrorBoundary відповідає за стійкість інтерфейсу, а httpClient обмежує час очікування зовнішніх запитів.

Таким чином, експлуатаційні вимоги в роботі сформульовано як поєднання прозорих припущень, контрольованої продуктивності та зрозумілої інтерпретації результатів. Це дозволяє використовувати CityTwin для демонстрації і попереднього аналізу, не створюючи хибного враження про нормативну точність розрахунків.

2.11 Висновки до другого розділу

У другому розділі виконано проєктування кіберфізичної системи CityTwin як інтерактивного прототипу цифрового двійника міського середовища. Визначено вимоги до системи, сценарії використання, структурну схему, архітектуру програмного комплексу та роль основних компонентів у загальній логіці роботи застосунку.

Обґрунтовано використання відкритих просторових і погодних даних. OpenStreetMap та OpenFreeMap формують геометричну основу міста, Open-Meteo забезпечує погодні параметри, а локальні структури застосунку зберігають сценарні зміни користувача. На цій основі спроєктовано модулі оцінювання вітрового комфорту, шумового навантаження та теплового стресу з поданням результатів у вигляді картографічних шарів.

Окремо визначено логіку сценарного редактора, інтерфейсу користувача, багатомовності, обмеження моделі та вимоги до надійності. CityTwin не замінює професійні інженерні комплекси, а призначений для попереднього порівняння міських сценаріїв, навчальної демонстрації та підготовки рішень до детальнішого аналізу. Отже, у другому розділі сформовано проєктну основу для подальшої програмно-апаратної реалізації та тестування прототипу.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ КІБЕРФІЗИЧНОЇ СИСТЕМИ CITYTWIN

3.1 Організація коду, інструменти розробки та місце апаратного контуру

Код CityTwin організовано як Vite-проєкт із TypeScript, модульною структурою src та окремими наборами тестів [9-14]. Така організація дає змогу показати систему не лише як набір сторінок, а як клієнтський застосунок із власною аналітичною логікою.

package.json містить основні команди для локального запуску, збірки, попереднього перегляду, перевірки якості коду і тестування поведінки застосунку.

TypeScript і конфігурація Vite забезпечують типізацію, JSX-компіляцію, HMR під час розроблення та оптимізовану збірку. Для реалізації CityTwin це важливо, бо помилки імпорту або типів бажано виявляти до демонстрації системи.

Каталог analysis містить моделі й побудову розрахункових полів, components - карту та елементи інтерфейсу, hooks - стан і побічні ефекти, api - роботу із зовнішніми сервісами, а data - підготовлені довідкові структури. Такий поділ робить код читабельнішим і полегшує пошук потрібного модуля.

Тести розділено за analysis, api, components, data, hooks, i18n, integration, pages, simulation та utils. Це дозволяє перевіряти не тільки інтерфейс, а й окремі розрахункові функції, роботу з програмним інтерфейсом API, симуляцію та допоміжні перетворення.

Динамічні imports використано для MapScene і частини панелей, щоб зменшити початкове навантаження інтерфейсу. Це практичне рішення для картографічного застосунку, де важкі модулі не обов'язково повинні завантажуватися одночасно з усім іншим кодом.

Отже, структура коду підтримує подальший розвиток CityTwin. Модель шуму, програмний інтерфейс API погоди, сценарний редактор або панель

інтерфейсу користувача можна змінювати окремо, не перетворюючи кожну правку на ризик для всієї системи.

Структура програмного коду CityTwin побудована за функціональним принципом. У components розміщені основні елементи інтерфейсу: карта, панелі керування, навігація, повідомлення та візуальні компоненти. У analysis зосереджена логіка розрахунку вітру, шуму й теплового стресу. Каталог data містить підготовлені просторові дані та міські пресети, а api відповідає за взаємодію із зовнішніми джерелами, зокрема погодною інформацією. Окремо виділені hooks, які допомагають керувати станом і поведінкою застосунку.

Такий поділ спрощує підтримку проєкту. Наприклад, зміни в компоненті карти не повинні ламати модель шуму, а зміни в програмному інтерфейсі API погоди не повинні впливати на сценарний редактор. Це робить програму більш зрозумілою і дає можливість поступово розвивати її після завершення базової реалізації.

3.2 Реалізація програмно-апаратної інтеграції та формату телеметрії

У реалізованій версії CityTwin програмно-апаратний контур подано через адаптерний підхід. Поточний застосунок отримує погодні параметри з Open-Meteo, але ці дані розглядаються як віртуальна телеметрія, сумісна з майбутнім польовим вузлом. Тому на рівні реалізації важливо не те, що джерелом зараз є програмний інтерфейс API, а те, що дані приводяться до внутрішньої структури, яку надалі може заповнювати ESP32 або Raspberry Pi.

Логічно такий адаптер розміщується між api/weatherApi.ts, hooks/useWeather.ts і аналітичними модулями. weatherApi.ts відповідає за HTTP-запит, перевірку відповіді та перетворення її у набір погодних параметрів. useWeather координує завантаження цих параметрів у стан застосунку. Далі дані використовуються моделями windField, heatAnalysis і візуальними компонентами. Для сенсорного вузла цей шлях залишився б таким самим:

змінюється тільки перше джерело, а внутрішній контракт залишається стабільним.

Таблиця 3.1 – Відповідність телеметрії апаратного вузла модулям CityTwin

Дані вузла	Джерело у прототипі	Модуль використання	Перевірка
Швидкість і напрямок вітру	Open-Meteo	windField, lbmWorker, WindCompass	Діапазони, резервний сценарій, оновлення режиму вітру
Температура і вологість	Open-Meteo	heatAnalysis, AnalysisPanel	Коректність одиниць і наявність значень
Шумовий рівень	Розрахунок від дорожньої мережі	noiseAnalysis, renderOverlay	Порівняння з очікуваними dBA-діапазонами
Координати вузла	Місто та межі карти	useMapProjection, featureExtractor	Потрапляння у межі області і проєкція на сітку
Ознака джерела	open-meteo або ручний режим	майбутній telemetryAdapter	Вибір пріоритету сенсорних і відкритих даних

Формат телеметричного пакета може бути поданий як JSON-об'єкт з полями nodeId, timestamp, координатами, windSpeed, windDirection, temperature, humidity, pressure, noiseDbA та джерелом даних. У браузерному прототипі цей об'єкт не зберігається як окремий серверний документ, але така структура природно узгоджується з існуючими типами у src/analysis/types.ts і з погодним сервісом. Це дає можливість розвивати систему без зміни моделей: новий адаптер лише перетворює сирі покази давачів у вже очікувані параметри.

Якщо використовується HTTP, edge-вузол може надсилати телеметрію періодично, наприклад раз на 30 або 60 секунд. Якщо використовується MQTT,

вузол публікує повідомлення у topic на зразок citytwin/khmelnytskyi/node-01/weather. У будь-якому випадку CityTwin не повинен напряму залежати від конкретного протоколу. Протокол належить до мережевого шару, а аналітичне ядро має отримувати нормалізовані дані.

Для практичної реалізації важливо передбачити пріоритет джерел. Найвищий пріоритет має локальний сенсорний вузол, тому що він вимірює саме ту точку, яка аналізується. Другий рівень - Open-Meteo як резервне джерело погодних параметрів. Третій рівень - ручне введення користувача для демонстраційного режиму. Така логіка не приховує обмеження прототипу, але робить архітектуру придатною до переходу від відкритих даних до польових вимірювань.

3.3 Тестування API, сенсорного контуру та відмов джерел даних

Тестування програмно-апаратної частини у межах цієї роботи має дві різні задачі. Перша задача - підтвердити, що реалізований браузерний прототип стабільно працює з відкритими джерелами даних і не руйнує інтерфейс при помилках API. Друга задача - показати, як має перевірятися майбутній сенсорний контур, навіть якщо фізичний стенд не зібрано. Саме така постановка дозволяє чесно відділити реалізовану програмну частину від спроєктованого апаратного рівня.

Для API-рівня вже є тести, які перевіряють weatherApi та httpClient. Вони важливі для КФС, бо погодні дані виступають заміщенням сенсорних вимірювань. Якщо API повертає неповну відповідь, тайм-аут або помилку, система повинна перейти у контрольований стан, а не показати некоректний аналітичний шар. У цьому сенсі тести API виконують роль перевірки каналу спостереження.

Для майбутнього сенсорного вузла потрібно перевіряти кілька рівнів. На рівні електричного підключення перевіряється живлення 3.3 V, спільна земля,

наявність I2C-пристрою, імпульси анемометра на GPIO та коректне зчитування ADC для флюгера. На рівні прошивки перевіряються часові мітки, діапазони значень, усереднення шуму та повторна передача після втрати зв'язку. На рівні CityTwin перевіряється, що пакет телеметрії приймається, нормалізується і змінює відповідний режим аналізу.

Окремо необхідно тестувати відмови. Для КФС це принципово, бо фізичні давачі можуть давати шум, пропуски або недопустимі значення. Наприклад, швидкість вітру не може бути від'ємною, вологість не повинна виходити за межі 0-100 %, а шумовий рівень має контролюватися за реалістичним діапазоном. Якщо дані не проходять перевірку, система повинна або взяти резервне джерело, або позначити результат як недостатньо надійний.

Таким чином, тестування CityTwin не обмежується командою npm run test. Команда підтверджує стабільність реалізованого коду, але програмно-апаратний характер системи вимагає також проєктної методики перевірки сенсорного контуру. У записці це важливо показати явно: прототип працює з відкритими даними, але його архітектура та формат телеметрії підготовлені до переходу на реальні вимірювання.

3.4 Реалізація карти та шару витягу просторових об'єктів

Підрозділ описує реалізацію карти як центрального компонента середовища виконання CityTwin. Саме MapScene поєднує MapLibre GL, межі поточної області, завантажені векторні тайли, сценарні об'єкти і шари результатів. Без цього компонента аналітичні модулі мали б дані, але не мали б єдиного просторового контексту [2-7].

Базова карта формується з темного стилю MapLibre: окремо описано шари води, зелених зон, доріг, залізниць, будівель і підписів. Такий стиль обрано тому, що поверх нього добре читаються кольорові оверлеї вітру, шуму і тепла.

Супутниковий шар ArcGIS підключено як допоміжну підкладку для візуальної перевірки, але він не бере участі в обчисленнях.

Витяг просторових об'єктів виконується через querySourceFeatures. Компонент читає об'єкти з джерела openmaptiles у межах поточної карти, після чого featureExtractor перетворює їх на структури, придатні для розрахунків. Будівлі, дороги, водні об'єкти і землекористування не зберігаються як окрема база, а готуються під час роботи застосунку.

Функції extractBuildingFootprints, extractRoads і extractPolygonRings нормалізують геометрію. Для будівель важливо отримати полігональні контури й орієнтовну висоту, для доріг - класифікацію і лінійну геометрію, для зелених зон і води - замкнені полігони. Після цього дані можна переносити на розрахункову сітку.

Межі карти перетворюються на GeoBounds, а фізичний розмір клітинки визначає масштаб сітки. Завдяки цьому одна й та сама логіка може працювати для різних міст: змінюється не алгоритм, а набір об'єктів у поточному фрагменті карти.

На рівні файлів основна логіка зосереджена у src/components/MapScene.tsx, src/components/featureExtractor.ts і src/components/mapSceneUtils.ts. MapScene координує події карти та стан застосунку, featureExtractor готує геометрію, а mapSceneUtils містить допоміжні перетворення для меж, координат і сітки.

Результатом роботи цього шару є не лише відображення карти, а підготовлений набір даних для моделей. Коли користувач змінює місто, масштаб або сценарний об'єкт, MapScene має оновити просторову основу, передати її до аналізу і показати актуальний шар на тому самому полотні.

Отже, реалізація карти в CityTwin виконує роль точки збирання даних і візуалізації. Вона переводить відкриті геодані у внутрішні структури застосунку, підтримує однаковий просторовий масштаб для моделей і дає користувачу зрозумілу основу для порівняння сценаріїв. Приклад наведено на рис. 3.1.

Рисунок 3.1 – Візуалізація режиму вітрового комфорту

3.5 Реалізація чисельної та евристичної моделей вітру

Підрозділ розкриває, як у CityTwin реалізовано два рівні моделювання вітру: швидке евристичне поле для інтерактивної роботи та LBM D2Q9-симуляцію для наочнішого показу обтікання перешкод [15-17, 26]. Такий поділ потрібний, бо користувач очікує швидкої реакції інтерфейсу, але дипломна робота також має показати інженерну логіку чисельного підходу.

Евристична модель починається з computeCompositeWindVectorField. Функція отримує межі карти, напрямок і швидкість вітру, висоту зрізу, контури будівель та зелені зони. На виході формується векторне поле, у якому враховано вітрові тіні, локальне прискорення між перешкодами, вплив шорсткості та нормування швидкості.

Для підготовки перешкод buildObstacleGrid переносить контури будівель на регулярну сітку. Це дозволяє швидко визначити, які клітинки зайняті забудовою, а які залишаються відкритими для потоку. Такий крок важливий і для

евристики, і для LBM, бо обидва підходи працюють з однаково підготовленим простором.

Параметр computeDynamicAlpha використовує доступні погодні дані: швидкість вітру на 10, 80 і 120 м або, якщо їх недостатньо, клас стабільності Пасквілла-Гіффорда. Завдяки цьому профіль вітру не задається вручну одним числом, а пов'язується з даними Open-Meteo.

Чисельний режим створюється через createWindSimulationState. У ньому ініціалізуються D2Q9-розподіли, маска перешкод, поля швидкості та граничні умови. Подальші кроки LBM виконують поширення і зіткнення розподілів, після чого поле швидкості можна передати у візуалізацію.

Щоб не блокувати інтерфейс, LBM винесено в src/simulation/lbmWorker.ts. Фоновий вебпрацівник приймає підготовлений стан, виконує ітерації окремо від потоку інтерфейсу і повертає результат у форматі, який може бути показаний на карті. Це особливо важливо для демонстрації, коли користувач змінює сценарій і очікує видимого оновлення.

Основні файли цього підрозділу - src/analysis/windField.ts, src/simulation/lbm.ts і src/simulation/lbmWorker.ts. Перший відповідає за інтерактивну оцінку, другий містить чисельну логіку, третій організовує виконання важчих обчислень.

Отже, модель вітру в CityTwin побудовано як компроміс між швидкістю, наочністю і фізичною правдоподібністю. Вона не претендує на повну CFD-точність, але дозволяє побачити, як забудова або сценарне втручання можуть змінити напрямок і відносну швидкість потоку у вибраному фрагменті міста.

3.6 Реалізація моделей шумового поля та теплового стресу

У цьому підрозділі описано дві моделі, які працюють за спільною схемою растеризації простору, але оцінюють різні явища: шумове навантаження і тепловий стрес [18-21, 28]. Обидві моделі використовують підготовлені

геометрії карти, тому результати можна показувати однаковим механізмом оверлеїв.

Шумова модель реалізована у src/analysis/noiseAnalysis.ts. Roads класифікуються за типами, після чого для кожної групи формується відносна інтенсивність. Далі модель оцінює поширення шуму від дорожньої мережі по сітці та враховує, що будівлі можуть екранувати частину впливу.

Для екранування використано полігональні контури будівель і висотний коефіцієнт. Якщо промінь від дороги до клітинки перетинає забудову, рівень впливу зменшується. Це спрощення не замінює акустичний розрахунок із відбиттями фасадів, але дає зрозумілу картину для попереднього порівняння сценаріїв.

Теплова модель реалізована у src/analysis/heatAnalysis.ts. Вона враховує buildingGrid, roadGrid, greenGrid, waterGrid, heightDensityGrid і shadowGrid. Забудова та дороги підсилюють ризик перегрівання, зелені зони й вода зменшують його, а щільність та тіні уточнюють локальну оцінку.

Для прискорення теплового аналізу використано Summed Area Tables. Вони дозволяють швидко обчислювати локальну щільність забудови або поверхонь у вікні навколо клітинки. Без такого прийому кожне оновлення сценарію потребувало б значно більшої кількості операцій.

Обидві моделі повертають AnalysisGrid. Цей спільний формат містить розміри сітки, межі карти і нормовані значення впливу. Завдяки однаковому результату renderOverlay може будувати шар незалежно від того, чи йдеться про шум, тепло або інший режим аналізу.

Обмеження моделей залишаються явними: шум не враховує повну хвильову картину, а теплова оцінка не моделює добову інерцію матеріалів і нічне охолодження. У межах бакалаврської роботи це прийнятно, бо завдання полягає у демонстрації зв'язку між просторовими даними, сценарієм і наочним результатом.

Отже, моделі шуму й теплового стресу доповнюють вітровий аналіз і показують, що один сценарій міського втручання може мати кілька наслідків. У CityTwin ці наслідки подано в однаковій логіці: підготовка геометрії, розрахунок сітки, нормування значень і відображення шару на карті. Приклади наведено на рис. 3.2 і рис. 3.3.

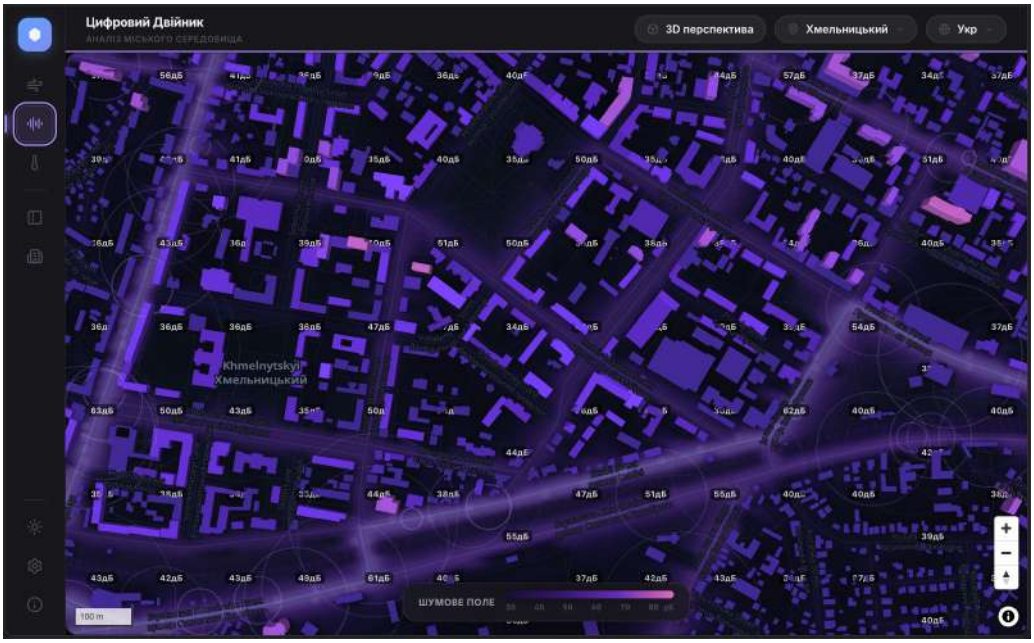


Рисунок 3.2 – Візуалізація шумового поля у 2D-режимі

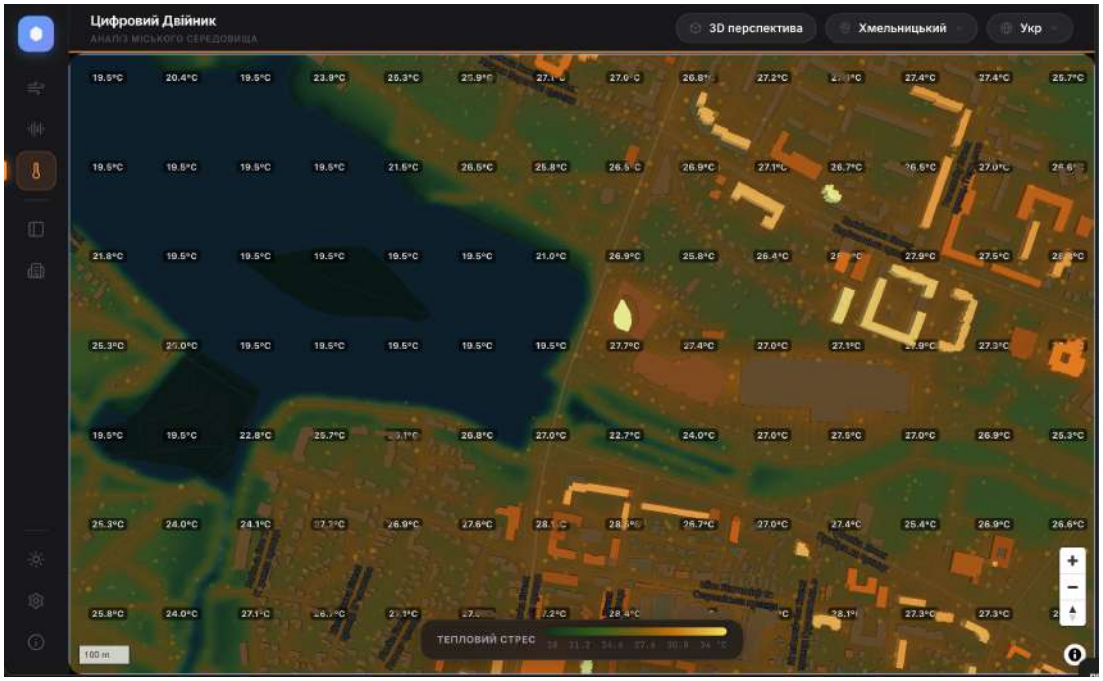


Рисунок 3.3 – Візуалізація теплового стресу

3.7 Реалізація візуалізації результатів і сценарного порівняння

Підрозділ присвячено тому, як розрахункові поля перетворюються на зрозумілий для користувача результат. У CityTwin числові значення не залишаються тільки масивом у пам'яті: вони подаються як кольорові шари, підписи, KPI та порівняння базового і сценарного станів [6-7, 32-33].

Функція renderGridToDataURL будує растровий шар для режимів вітру, шуму і тепла. Для кожного режиму використано окрему кольорову шкалу, бо однаковий колір не може означати те саме для швидкості вітру, шумового навантаження і теплового стресу. На виході формується URL-даних, який можна накласти на карту.

AnimatedOverlay відповідає за плавну появу шару і підписів. Це важливо для сприйняття результату: користувач бачить, що зміна сценарію не просто перемикає статичну картинку, а оновлює конкретний аналітичний шар поверх міського фрагмента.

Для вітрового режиму підготовлено WindParticleLayer. WebGL-шар частинок показує напрямок руху поля і робить результат зрозумілішим, ніж лише заливка кольором. Така візуалізація допомагає пояснити, де потік прискорюється, де сповільнюється і як забудова впливає на картину.

overlayManager збирає кілька типів шарів: результат аналізу, сценарні будівлі, парки та попередній контур майбутнього об'єкта. Завдяки цьому користувач бачить не тільки підсумковий вплив, а й саму причину зміни - доданий або запланований сценарний елемент.

buildAnalysisSummary формує короткі числові показники: середнє значення, пікову зону і частку проблемних клітинок для базового та сценарного стану. Ці KPI потрібні для порівняння: карта показує де відбулася зміна, а числовий блок показує наскільки вона помітна.

Основні файли цього шару - src/analysis/renderOverlay.ts, src/analysis/animatedOverlay.ts і src/components/overlayManager.ts. Вони

відокремлюють побудову зображення від логіки моделей, тому новий режим аналізу можна додати без повного переписування карти.

Отже, візуалізація в CityTwin виконує аналітичну функцію. Вона допомагає користувачу прочитати результат сценарію, порівняти його з базовим станом і зрозуміти, які ділянки міського середовища потребують детальнішої перевірки. 3D-режим і образ системи наведено на рис. 3.4 і рис. 3.5.

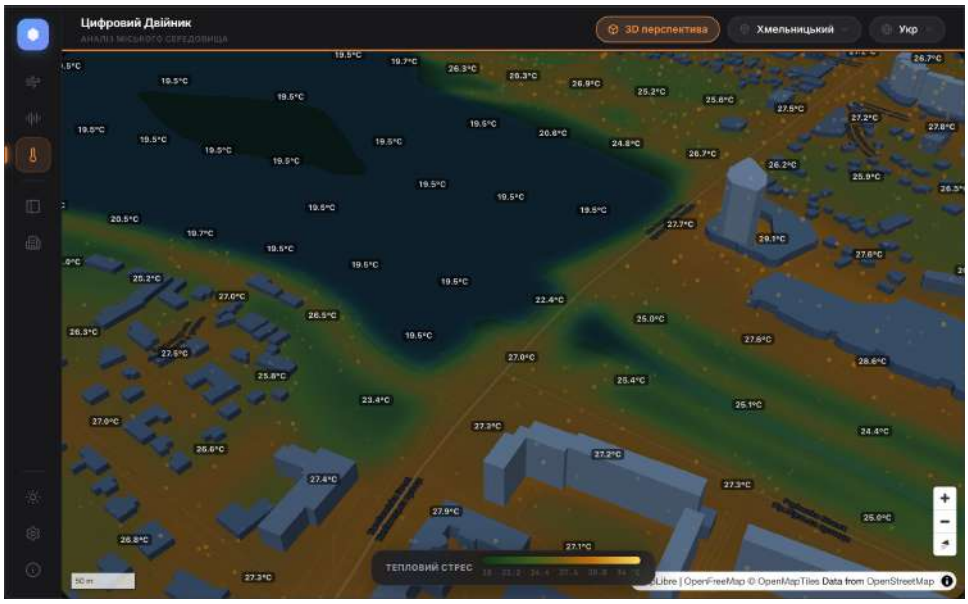


Рисунок 3.4 – Відображення теплового режиму в 3D-перспективі

Рисунок 3.5 – Графічний образ системи CityTwin

3.8 Тестування, збірка та аналіз отриманих результатів

Тестування в роботі виконувало практичну роль: потрібно було перевірити, чи CityTwin справді працює як єдиний застосунок, а не лише як набір окремих модулів. Перевірка охопила розрахункову логіку, компоненти інтерфейсу, API, hooks, інтеграційні сценарії та наскрізну взаємодію з картою [13-14, 35-36].

Локальний запуск Vitest підтвердив проходження 69 тестових файлів і 1356 тестів. Це важливо для аналітичних модулів, бо навіть невелика зміна у підготовці геометрії або нормуванні результату може вплинути на карту, яку бачить користувач.

Модульні тести перевіряють функції, які можна оцінити без повного інтерфейсу: нормалізацію даних, розрахунок показників, допоміжні перетворення і стабільність окремих модулів. Інтеграційні тести показують, чи правильно взаємодіють компоненти, hooks і стан застосунку.

Наскрізні перевірки через Playwright відтворюють ближчий до реальної роботи сценарій: користувач відкриває застосунок, переходить між режимами, працює з картою і перевіряє появу результатів. Такий рівень тестування потрібний саме для системи з картою, де помилка може виникнути на межі між даними, моделлю та візуальним шаром.

Виробнича збірка npm run build завершилась успішно і сформувала dist-артефакти. Vite попередив про великий фрагмент MapLibre, що очікувано для картографічної бібліотеки. Це не блокує демонстрацію, але може бути напрямом подальшої оптимізації [6-7].

Знімки екрана з результатів тестування та локального перегляду підтвердили, що режими вітру, шуму, тепла, шар накладання і 3D-перспектива відображаються. Тому перевірка охопила не тільки факт проходження команд, а й видимий результат у користувацькому інтерфейсі.

Загалом тестування підтверджує програмно-апаратну працездатність прототипу на рівні реалізованого кібернетичного контуру. Воно не доводить фізичну точність моделей, але показує, що застосунок стабільно виконує закладений сценарій: отримує дані, будує шари, реагує на дії користувача і дозволяє порівнювати базовий та змінений стан. Результати перевірки наведено у табл. 3.2.

Таблиця 3.2 – Результати локальної перевірки проєкту

Перевірка	Команда	Результат	Коментар
Модульні / інтеграційні	npm run test	69 файлів, 1356 тестів - пройдено	Локальний запуск виконано під час підготовки документа
Виробнича збірка	npm run build	успішно	Vite попередив про великий фрагмент MapLibre
Наскрізна структура	npm run test:e2e	підготовлено у проєкті	Навігація, панелі, шумовий шар, 3D-перспектива
Візуальна перевірка	Знімки екрана Playwright	створено рисунки режимів	Вітер, шум, тепло, 3D-перспектива

Під час реалізації моделі вітру важливо було зробити результат не лише технічно обчисленим, а й зрозумілим для користувача. У реальному застосуванні людину цікавить не сама формула, а те, що відбувається з простором після зміни сценарію. Тому система показує просторову картину: де потік може послаблюватися через забудову, де він проходить вільніше, а де можуть

виникати ділянки з помітнішою зміною швидкості. Такий спосіб подання дозволяє швидко порівнювати сценарії між собою.

Для шумового поля логіка подібна. Користувачу важливо побачити, які частини території можуть залишатися більш навантаженими, а які виглядають спокійнішими з погляду шуму. У прототипі це подано через візуальний шар, який допомагає не загубитися в числах і одразу побачити проблемні ділянки на карті. Такий підхід особливо корисний тоді, коли потрібно швидко оцінити не один точковий показник, а загальну картину середовища.

Тепловий аналіз також реалізовано з орієнтацією на порівняння сценаріїв. У місті нагрівання території залежить від багатьох чинників, але для користувача важливо побачити, як змінюється ситуація після додавання забудови або зелених зон. Якщо один сценарій створює більше умов для перегрівання, а інший виглядає комфортнішим, це вже дає корисний попередній висновок. Саме тому в CityTwin візуалізація є не додатковим елементом, а ключовою частиною роботи системи.

У результаті модулі аналізу й візуалізації працюють разом. Розрахунок формує дані, а карта перетворює їх на зрозумілу для користувача картину. Це важливо для всієї ідеї цифрового двійника: система повинна не просто виконувати обчислення, а допомагати людині інтерпретувати зміни в міському середовищі.

Тестування системи було потрібне не тільки для формальної перевірки працездатності. CityTwin складається з кількох взаємопов’язаних частин, тому помилка в одному модулі може вплинути на результат усього сценарію. Наприклад, некоректна обробка просторового об’єкта може спотворити візуалізацію, а помилка в керуванні станом може призвести до неправильного перемикання режимів аналізу.

У межах роботи перевірялися модулі, які відповідають за дані, аналітичні розрахунки, компоненти інтерфейсу та сценарну взаємодію. Такий підхід дозволяє переконатися, що система не просто відкривається у браузері, а виконує

основні дії користувача: показує карту, перемикає режими, обробляє сценарії та формує результати аналізу.

Окрему роль відіграє перевірка аналітичних модулів. Для вітру, шуму й теплового стресу важливо, щоб результат залишався узгодженим із вхідними даними та не змінювався випадково після оновлення інших частин застосунку. Тому тести допомагають контролювати базову стабільність розрахункової логіки.

Компоненти інтерфейсу також потребують перевірки, оскільки саме через них користувач взаємодіє з цифровим двійником. Якщо панелі керування, режими аналізу або повідомлення працюють некоректно, навіть правильна розрахункова модель стає менш корисною. Тому тестування інтерфейсу є важливою частиною загальної якості системи.

Під час збірки проєкту перевіряється, чи коректно поєднуються всі частини застосунку. Це дає змогу виявити помилки імпорту, проблеми типізації або несумісність між модулями ще до демонстрації результату. Така перевірка важлива, тому що підтверджує не лише наявність коду, а й його працездатність як єдиного прототипу КФС.

Отримані результати показують, що CityTwin може виконувати основні задачі, поставлені в роботі. Система відображає міський фрагмент, підтримує сценарне редагування, формує візуальні шари для аналізу та дозволяє порівнювати зміни. Це підтверджує практичну реалізованість обраної концепції цифрового двійника.

Водночас результати потрібно інтерпретувати з урахуванням наближеного характеру моделей. Система добре підходить для попереднього аналізу, але не повинна використовуватися як єдине джерело для остаточного інженерного рішення. Така межа застосування є нормальною для прототипу і відповідає задачам, поставленим у роботі.

Завдяки тестуванню, збірці та аналізу результатів можна зробити висновок, що розроблений застосунок є цілісним програмним рішенням. Він

поєднує теоретичну ідею цифрового двійника з практичною реалізацією у вигляді браузерної системи, придатної для демонстрації та подальшого розвитку.

Під час перевірки користувацьких сценаріїв окремо враховувалася послідовність дій у застосунку. Важливо, щоб користувач міг перейти від перегляду карти до вибору режиму аналізу, додати умовний об’єкт і побачити оновлений результат без втрати контексту. Саме така послідовність дій відповідає практичній ідеї цифрового двійника.

Також перевірялося, чи не виникає розриву між даними та візуалізацією. Якщо модель формує результат, але він подається на карті незрозуміло, користувач не зможе правильно інтерпретувати сценарій. Тому в роботі важливим було поєднання розрахункової частини з інтерфейсом, а не лише окрема перевірка формул або компонентів.

Результати тестування показали, що обрана структура проєкту є придатною для подальшого розвитку. Окремі модулі можна змінювати або доповнювати без повного переписування застосунку. Це особливо важливо для цифрового двійника, оскільки в майбутньому до нього можуть додаватися нові джерела даних, точніші моделі або додаткові режими аналізу.

Загалом тестування підтвердило, що CityTwin працює як єдина система, а не як набір окремих демонстраційних елементів. Карта, сценарії, модулі аналізу, стан застосунку та візуальні шари пов’язані між собою і підтримують основний робочий процес користувача.

Це дозволяє розглядати отриманий результат як завершений прототип у межах поставленої задачі. Він має обмеження, але ці обмеження відповідають обраному формату роботи: швидкій попередній оцінці міського середовища за відкритими даними та спрощеними моделями.

Під час аналізу отриманих результатів основну увагу було приділено не окремим числовим значенням, а тому, наскільки система дозволяє порівнювати сценарії між собою. Для цифрового двійника це важливо, оскільки користувач

зазвичай працює не з одним станом території, а з кількома можливими варіантами її зміни.

У режимі вітрового аналізу результат допомагає оцінити, як забудова може впливати на рух повітря. Якщо після додавання об’єкта частина території показує гірше провітрювання, це не означає автоматичної заборони такого рішення, але дає підставу звернути на нього увагу. Саме така попередня інтерпретація є корисною на ранньому етапі аналізу.

У шумовому режимі важливим є просторовий розподіл навантаження. Користувач може побачити, які ділянки залишаються більш проблемними, а де сценарій виглядає спокійнішим. Такий результат особливо корисний тоді, коли потрібно швидко порівняти кілька варіантів забудови або розміщення зелених зон. Тепловий режим показує, як зміни у просторі можуть впливати на загальну картину перегрівання. У щільній міській забудові це має практичне значення, тому що навіть невеликі зміни у співвідношенні забудови, відкритих поверхонь та озеленення можуть впливати на комфорт перебування людей.

Окремою перевагою реалізації є те, що всі режими аналізу подані в одному інтерфейсі. Користувачу не потрібно переходити між різними програмами або окремими розрахунковими модулями. Він працює з одним міським фрагментом і може послідовно оцінювати його за кількома показниками.

Такий підхід робить систему зручною для демонстрації. На прикладі одного сценарію можна показати, що просторова зміна має не один, а кілька можливих наслідків. Будівля може впливати на вітер, шум і тепловий стан одночасно, тому цифровий двійник має сенс саме як комплексний інструмент попередньої оцінки.

Під час роботи з результатами важливо також враховувати масштаб аналізу. CityTwin орієнтований на фрагмент міста, а не на повноцінну модель усього населеного пункту. Такий масштаб є зручним для дипломного проєкту, оскільки дозволяє зосередитися на конкретній території та показати роботу моделей без надмірного ускладнення системи.

Ще одним важливим моментом є зрозумілість візуалізації. Якщо результат подано занадто технічно, користувач може не побачити практичного змісту сценарію. Тому в CityTwin розрахункові дані перетворюються на шари, які можна сприймати як просторову картину: де ситуація краща, де гірша, а де потрібна додаткова перевірка.

У підсумку реалізовані режими аналізу підтверджують, що навіть спрощений цифровий двійник може бути корисним для першого ознайомлення з наслідками міських змін. Він не дає остаточного інженерного рішення, але допомагає сформувати більш обґрунтоване уявлення про сценарій.

Це особливо важливо для навчального контексту. Користувач бачить не лише результат, а й саму логіку роботи системи: карта надає просторову основу, дані формують модель, аналітичні модулі виконують оцінювання, а інтерфейс допомагає зрозуміти результат.

3.9 Висновки до розділу 3

Таким чином, аналіз отриманих результатів показує, що CityTwin виконує поставлену задачу як прототип цифрового двійника. Система забезпечує зв’язок між просторовими даними, сценарним редагуванням і візуальною оцінкою параметрів міського середовища.

Найважливішим результатом є не окрема модель вітру, шуму або тепла, а їхнє поєднання в одному робочому процесі. Саме це дозволяє розглядати CityTwin як цілісний застосунок, а не як набір окремих демонстраційних функцій.

Таким чином, описано програмно-апаратну реалізацію CityTwin і перевірено працездатність основних частин системи. Реалізовані карта, витяг просторових об’єктів, моделі аналізу, сценарне редагування, візуалізація та тестування підтверджують придатність прототипу для попереднього порівняння міських сценаріїв.

ВИСНОВКИ

Підсумком роботи став CityTwin - браузерний прототип цифрового двійника міського середовища. У ньому карта не існує окремо від розрахунків: просторові об'єкти, погодні умови, модулі аналізу та сценарні дії користувача працюють в одному потоці. Такий підхід дав змогу перевіряти не лише розміщення об'єктів на карті, а й можливі зміни вітрового режиму, шумового навантаження та теплового фону після додавання забудови або озеленення.

Аналіз предметної області показав, що головна цінність цифрового двійника полягає не в самому відображенні міста. Статична карта допомагає побачити, де розташовані будівлі й дороги, але вона не пояснює, як ці об'єкти впливають на середовище. Тому в роботі акцент зроблено на зв'язку між реальною геометрією кварталу, розрахунковими моделями та результатом, який одразу відображається на карті після зміни сценарію.

Для просторової основи CityTwin використано OpenStreetMap та OpenFreeMap. Ці джерела дозволили отримувати контури будівель, дороги, зелені зони й водні об'єкти без ручного створення окремого набору даних для кожної локації. Погодні дані надходять через Open-Meteo API: з нього застосунок бере швидкість і напрямок вітру, температуру, вологість та інші параметри, які потрібні для роботи моделей аналізу.

Архітектуру застосунку побудовано як React/TypeScript SPA. Під час проєктування було важливо розвести відповідальність між частинами системи: інтерфейсом, центральним станом, витягом даних із карти, аналітичними моделями та шарами візуалізації. Така структура зробила проєкт зрозумілішим для супроводу: окремі модулі можна перевіряти, змінювати або розширювати без повного переписування всієї програми.

Реалізовано евристичну модель вітрового поля з урахуванням вітрових тіней, зон тиску, ефекту Вентурі, локального кутового прискорення, турбулентності й висотного профілю. Додатково реалізовано LBM D2Q9-

симуляцію у фоновому вебпрацівнику, що демонструє чисельний підхід до моделювання обтікання перешкод.

Розроблено модель шумового поля, яка використовує класи дорожньої мережі, логарифмічне затухання, екранування будівлями та фасадний вплив. Реалізовано модель теплового стресу, що враховує забудову, дороги, зелені зони, воду, щільність об'єктів, тіні та реальну температуру повітря.

Створено сценарний редактор міських втручань. Користувач може додавати будівлі або парки, змінювати їхні параметри, а система перевіряє колізії з наявними об'єктами, дорогами, водою та сценарними елементами. Це забезпечує практичний механізм порівняння базового й зміненого стану.

Реалізовано візуалізацію результатів через MapLibre GL, Canvas/WebGL-оверлеї, кольорові шкали, підписи значень і KPI. Інтерфейс підтримує режими вітру, шуму й тепла, 2D/3D-представлення, супутникову підкладку, вибір міста та три мови інтерфейсу.

Працездатність проєкту перевірено локальними командами `npm run test` і `npm run build`. Модульні та інтеграційні тести пройшли успішно: 69 тестових файлів і 1356 тестів. Виробнича збірка також завершилася успішно, що підтверджує готовність реалізованого кібернетичного рівня КФС до демонстраційного використання.

Практичне значення отриманого результату полягає в тому, що CityTwin показує повний шлях створення цифрового двійника на прикладному рівні. У роботі поєднано відкриті геодані, погодну інформацію, браузерну карту, сценарне редагування, аналітичні моделі та візуалізацію результатів. Це робить систему корисною не тільки як браузерний прототип КФС, а й як навчальний приклад того, як міські дані можуть перетворюватися на зрозумілий інструмент аналізу.

Водночас результати роботи потрібно трактувати в межах обраної постановки. CityTwin не замінює професійні CFD, акустичні або кліматичні комплекси, але дозволяє швидко оцінити напрям змін і побачити потенційно

проблемні зони. Саме така роль є доречною для бакалаврської кваліфікаційної роботи: система демонструє інженерну логіку, прозорість припущень і можливість подальшого розвитку.

Подальший розвиток системи доцільно спрямувати на збереження сценаріїв, підключення серверного кешу геоданих, калібрування моделей за польовими вимірюваннями, розширення LBM або CFD-модуля до тривимірних розрахунків і формування експортованих звітів для містобудівних порівнянь.

Подальше калібрування доцільно виконувати поетапно. Спочатку можна зберігати користувацькі сценарії та додати експорт звіту, після цього - підключити серверний кеш геоданих і порівняння з польовими вимірюваннями, а вже потім розширювати фізичні моделі. Такий шлях розвитку є реалістичним, бо не руйнує поточну архітектуру і дозволяє поступово підвищувати точність без втрати інтерактивності.

Для практичного використання важливо, що прототип не приховує своїх обмежень. Якщо потрібен остаточний висновок для реального будівництва, результати потрібно порівнювати з вимірюваннями, спеціалізованими CFD, акустичними або кліматичними моделями. Однак для першого відбору сценаріїв CityTwin дає корисну перевагу: він швидко показує напрям зміни і допомагає зрозуміти, який варіант варто досліджувати детальніше.

Окремо слід підкреслити, що цінність CityTwin полягає у прозорому поєднанні кількох рівнів роботи з даними. Користувач бачить карту, але за нею послідовно працюють витяг просторових об'єктів, підготовка розрахункової сітки, вибір моделі, побудова шару результатів і коротке числове узагальнення. Саме така послідовність робить систему придатною для пояснення логіки цифрового двійника: результат не з'являється як окрема картинка, а формується з конкретних вхідних даних і зрозумілих припущень.

З огляду на це отриманий результат можна вважати завершеним для поставленої мети. У роботі показано не лише окремі алгоритми, а повний шлях їх включення у користувацький сценарій: від вибору міста до порівняння

базового і зміненого станів. Такий результат відповідає рівню бакалаврської кваліфікаційної роботи, оскільки демонструє проєктування, реалізацію, тестування і критичне пояснення меж застосування системи.

Після уточнення архітектури роботу доцільно розглядати не як ізольований вебзастосунок, а як програмно-апаратну кіберфізичну систему з реалізованим кібернетичним рівнем і спроєктованим апаратним шаром. У такій постановці CityTwin отримує чіткий фізичний контекст: міське середовище є об'єктом спостереження, давачі або відкриті джерела формують вхідні дані, edge-вузол виконує попередню обробку, а цифровий двійник перетворює ці дані на просторовий результат для користувача.

Важливим результатом роботи є структурна схема КФС, яка показує повний контур: фізичне міське середовище, давачі, периферійна обробка/шлюз, мережевий шар, кібернетичний двійник, інтерфейс, рішення користувача і зворотний вплив на середовище. Ця схема відокремлює кіберфізичну архітектуру від архітектури програмного комплексу і пояснює, чому CityTwin не слід трактувати лише як програмну систему.

У роботі також спроєктовано апаратний рівень збору даних. Для нього визначено мінімальний набір компонентів: анемометр для швидкості вітру, флюгер для напрямку, BME280 або SHT31 для температури, вологості та тиску, DS18B20 для локальної температури, шумовий датчик або MEMS-мікрофон для акустичного каналу, а також ESP32 або Raspberry Pi як edge-вузол. Запропонована електрична структура не стверджує, що стенд фізично зібрано, але дає реалістичну основу для його подальшого створення.

Окремо визначено формат телеметрії, який дозволяє поєднати відкриті джерела даних і майбутні польові вимірювання. У поточному прототипі Open-Meteo та OpenStreetMap/OpenFreeMap виконують роль заміщення фізичних спостережень: перше джерело надає погодні параметри, друге - геометрію міського середовища. При підключенні реального вузла ті самі поля можуть

заповнюватися сенсорними даними, а програмні моделі залишаються сумісними з уже реалізованою логікою.

Таким чином, поставлена мета досягнута у межах чесної бакалаврської постановки: реалізовано програмну частину кіберфізичної системи та спроєктовано апаратний рівень збору польових даних. Прототип не підміняє професійні CFD, акустичні або кліматичні комплекси і не видає результати за нормативний розрахунок. Його призначення полягає у швидкому попередньому аналізі, навчальній демонстрації та підготовці сценаріїв, які в подальшому можуть перевірятися точнішими інженерними методами або польовими вимірюваннями.

Подальший розвиток системи доцільно виконувати у два взаємопов'язані напрями. Перший напрям - програмний: додати серверний телеметричний адаптер, кеш сценаріїв, експорт звітів і механізм пріоритету між сенсорним джерелом, open-meteo та ручним джерелом. Другий напрям - апаратний: зібрати один ESP32-вузол, перевірити I2C, 1-Wire, ADC та I2S-канали, виконати калібрування шумового і температурного каналу, після чого порівняти відкриті погодні дані з локальними вимірюваннями.

У підсумку CityTwin має завершений вигляд як кіберфізична система рівня прототипу. Сильна сторона роботи полягає не тільки у вебінтерфейсі, а в поєднанні фізичного міського середовища, джерел спостереження, апаратного проєкту, моделей аналізу, сценарного редагування, візуалізації та тестування. Саме таке поєднання відповідає темі роботи і переводить акцент із суто програмної реалізації на програмно-апаратну архітектуру.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Grieves M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication : white paper. [S. l.] : Michael W. Grieves, LLC, 2014. 7 p.
2. OpenStreetMap contributors. Copyright and License. OpenStreetMap. URL: https://www.openstreetmap.org/copyright/en (дата звернення: 18.05.2026).
3. OpenMapTiles. OpenMapTiles vector tile schema. URL: https://openmaptiles.org/schema/ (дата звернення: 18.05.2026).
4. OpenFreeMap. Free and open-source map hosting and vector tiles. URL: https://openfreemap.org/ (дата звернення: 18.05.2026).
5. Open-Meteo. Weather Forecast API. URL: https://open-meteo.com/en/docs (дата звернення: 18.05.2026).
6. MapLibre. MapLibre GL JS: JavaScript library for interactive maps. URL: https://maplibre.org/maplibre-gl-js/docs/ (дата звернення: 18.05.2026).
7. MapLibre. MapLibre GL JS. URL: https://maplibre.org/projects/gl-js/ (дата звернення: 18.05.2026).
8. React. The library for web and native user interfaces. URL: https://react.dev/ (дата звернення: 18.05.2026).
9. TypeScript. The TypeScript Handbook. URL: https://www.typescriptlang.org/docs/handbook/ (дата звернення: 18.05.2026).
10. Vite. Vite Guide. URL: https://vite.dev/guide/ (дата звернення: 18.05.2026).
11. React Router. BrowserRouter. URL: https://reactrouter.com/api/declarative-routers/BrowserRouter (дата звернення: 10.06.2026).
12. i18next. i18next documentation. URL: https://www.i18next.com/ (дата звернення: 18.05.2026).
13. Vitest. Vitest Guide. URL: https://vitest.dev/guide/ (дата звернення: 18.05.2026).
14. Playwright. Running and debugging tests. URL: https://playwright.dev/docs/running-tests (дата звернення: 18.05.2026).

15. Kliemank M. L., Wilde D., Bedrunka M. C., Krämer A., Foysi H., Reith D. Assessment of lattice Boltzmann method for low-rise building wind flow simulation with limited resources. Discrete and Continuous Dynamical Systems - S. 2024. Vol. 17(11). P. 3205-3223. DOI: 10.3934/dcdss.2023046.

16. Kruger T., Kusumaatmaja H., Kuzmin A., Shardt O., Silva G., Viggen E. M. The Lattice Boltzmann Method: Principles and Practice. Cham : Springer, 2017. XXIV, 694 p. DOI: 10.1007/978-3-319-44649-3.

17. Zhong J., Liu J., Zhao Y., Niu J., Carmeliet J. Recent advances in modeling turbulent wind flow at pedestrian-level in the built environment. Architectural Intelligence. 2022. Vol. 1. Article 5. DOI: 10.1007/s44223-022-00008-7.

18. World Health Organization. Environmental noise guidelines for the European Region. Copenhagen : WHO Regional Office for Europe, 2018. 160 p. ISBN 978-92-890-5356-3. URL: https://www.who.int/europe/publications/i/item/9789289053563 (дата звернення: 10.06.2026).

19. ISO 9613-2:2024. Acoustics - Attenuation of sound during propagation outdoors - Part 2: Engineering method for the prediction of sound pressure levels outdoors. 2nd ed. Geneva : International Organization for Standardization, 2024. 46 p. URL: https://www.iso.org/standard/74047.html (дата звернення: 10.06.2026).

20. U.S. Environmental Protection Agency. What Are Heat Islands? URL: https://www.epa.gov/heatislands/what-are-heat-islands (дата звернення: 10.06.2026).

21. European Environment Agency. Environmental noise in Europe 2025. EEA Report No. 05/2025. Luxembourg : Publications Office of the European Union, 2025. ISBN 978-92-9480-765-6. ISSN 1977-8449. DOI: 10.2800/5134480. URL: https://www.eea.europa.eu/en/analysis/publications/environmental-noise-in-europe-2025 (дата звернення: 10.06.2026).

22. Batty M. The New Science of Cities. Cambridge, MA : MIT Press, 2013. XXI, 496 p.

23. Tao F., Zhang H., Liu A., Nee A. Y. C. Digital Twin in Industry: State-of-the-Art. IEEE Transactions on Industrial Informatics. 2019. Vol. 15(4). P. 2405-2415. DOI: 10.1109/TII.2018.2873186.

24. Goodchild M. F. Citizens as sensors: the world of volunteered geography. *GeoJournal*. 2007. Vol. 69(4). P. 211-221. DOI: 10.1007/s10708-007-9111-y.

25. Boeing G. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems.* 2017. Vol. 65. P. 126-139. DOI: 10.1016/j.compenvurbsys.2017.05.004.

26. Jacob J., Merlier L., Marlow F., Sagaut P. Lattice Boltzmann method-based simulations of pollutant dispersion and urban physics. *Atmosphere*. 2021. Vol. 12(7). Article 833. DOI: 10.3390/atmos12070833.

27. Seinfeld J. H., Pandis S. N. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. 3rd ed. Hoboken : Wiley, 2016. XXVI, 1120 p.

28. Cárdenas-León I., Morales-Ortega R., Koeva M., Atún F., Pfeffer K. Digital twin-based framework for heat stress calculation. I*SPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences.* 2024. Vol. X-4-2024(4). P. 67-74. DOI: 10.5194/isprs-annals-X-4-2024-67-2024.

29. ISO 37120:2018. Sustainable cities and communities - Indicators for city services and quality of life. 2nd ed. Geneva : International Organization for Standardization, 2018. 121 p. URL: https://www.iso.org/standard/68498.html (дата звернення: 10.06.2026).

30. ISO 37122:2019. Sustainable cities and communities - Indicators for smart cities. 1st ed. Geneva : International Organization for Standardization, 2019. 95 p. URL: https://www.iso.org/standard/69050.html (дата звернення: 10.06.2026).

31. WHATWG. HTML Living Standard: Web workers. Living Standard. Last updated 10 June 2026. URL: https://html.spec.whatwg.org/multipage/workers.html (дата звернення: 10.06.2026).

32. MDN Web Docs. Canvas API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API (дата звернення: 18.05.2026).

33. MDN Web Docs. WebGL API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API (дата звернення: 18.05.2026).

34. MDN Web Docs. Web Storage API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API (дата звернення: 18.05.2026).

35. Testing Library. React Testing Library: introduction. URL: https://testing-library.com/docs/react-testing-library/intro/ (дата звернення: 18.05.2026).

36. ESLint. ESLint documentation. URL: https://eslint.org/docs/latest/ (дата звернення: 18.05.2026).

37. Butler H., Daly M., Doyle A., Gillies S., Hagen S., Schaub T. The GeoJSON Format. RFC 7946. RFC Editor, 2016. 28 p. DOI: 10.17487/RFC7946. URL: https://www.rfc-editor.org/info/rfc7946/ (дата звернення: 10.06.2026).

38. OpenStreetMap Wiki. Map features. URL: https://wiki.openstreetmap.org/wiki/Map_features (дата звернення: 18.05.2026).

39. OpenStreetMap Wiki. Key:building. URL: https://wiki.openstreetmap.org/wiki/Key:building (дата звернення: 18.05.2026).

40. OpenStreetMap Wiki. Key:highway. URL: https://wiki.openstreetmap.org/wiki/Key:highway (дата звернення: 18.05.2026).

41. WHATWG. Fetch Standard : Living Standard. Last updated 5 June 2026. URL: https://fetch.spec.whatwg.org/ (дата звернення: 10.06.2026).

42. Greer C., Burns M., Wollman D., Griffor E. Cyber-Physical Systems and Internet of Things. NIST Special Publication 1900-202. *Gaithersburg, MD : National Institute of Standards and Technology*, 2019. 61 p. DOI: 10.6028/NIST.SP.1900-202. URL: https://www.nist.gov/publications/cyber-physical-systems-and-internet-things (дата звернення: 10.06.2026).

43. ISO/IEC/IEEE 42010:2022. Software, systems and enterprise - Architecture description. 2nd ed. Geneva : International Organization for Standardization, 2022. 62 p. URL: https://www.iso.org/standard/74393.html (дата звернення: 10.06.2026).

44. Espressif Systems. ESP32 Series Datasheet. Version 5.2. Shanghai : Espressif Systems, 2025. 78 p. URL:

https://documentation.espressif.com/esp32_datasheet_en.pdf (дата звернення: 10.06.2026).

45. Bosch Sensortec. BME280 Combined humidity and pressure sensor : data sheet. Document revision 1.24. Document number BST-BME280-DS001-24. Reutlingen : Bosch Sensortec GmbH, 2024. 60 p. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (дата звернення: 10.06.2026).

46. Analog Devices. DS18B20 Programmable Resolution 1-Wire Digital Thermometer : data sheet. Rev. 6. Wilmington : Analog Devices, 2019. 20 p. URL: https://www.analog.com/media/en/technical-documentation/data-sheets/ds18b20.pdf (дата звернення: 10.06.2026).

47. TDK InvenSense. INMP441 Omnidirectional Microphone with Bottom Port and I2S Digital Output : data sheet. Document number DS-INMP441-00. Revision 1.1. Rev. date 21.05.2014. San Jose : InvenSense Inc., 2014. 21 p. URL: https://product.tdk.com/system/files/dam/doc/product/sw_piezo/mic/mems-mic/data_sheet/inmp441.pdf (дата звернення: 10.06.2026).

48. OASIS. MQTT Version 5.0. Edited by A. Banks, E. Briggs, K. Borgendale, R. Gupta. OASIS Standard, 07 March 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (дата звернення: 10.06.2026).

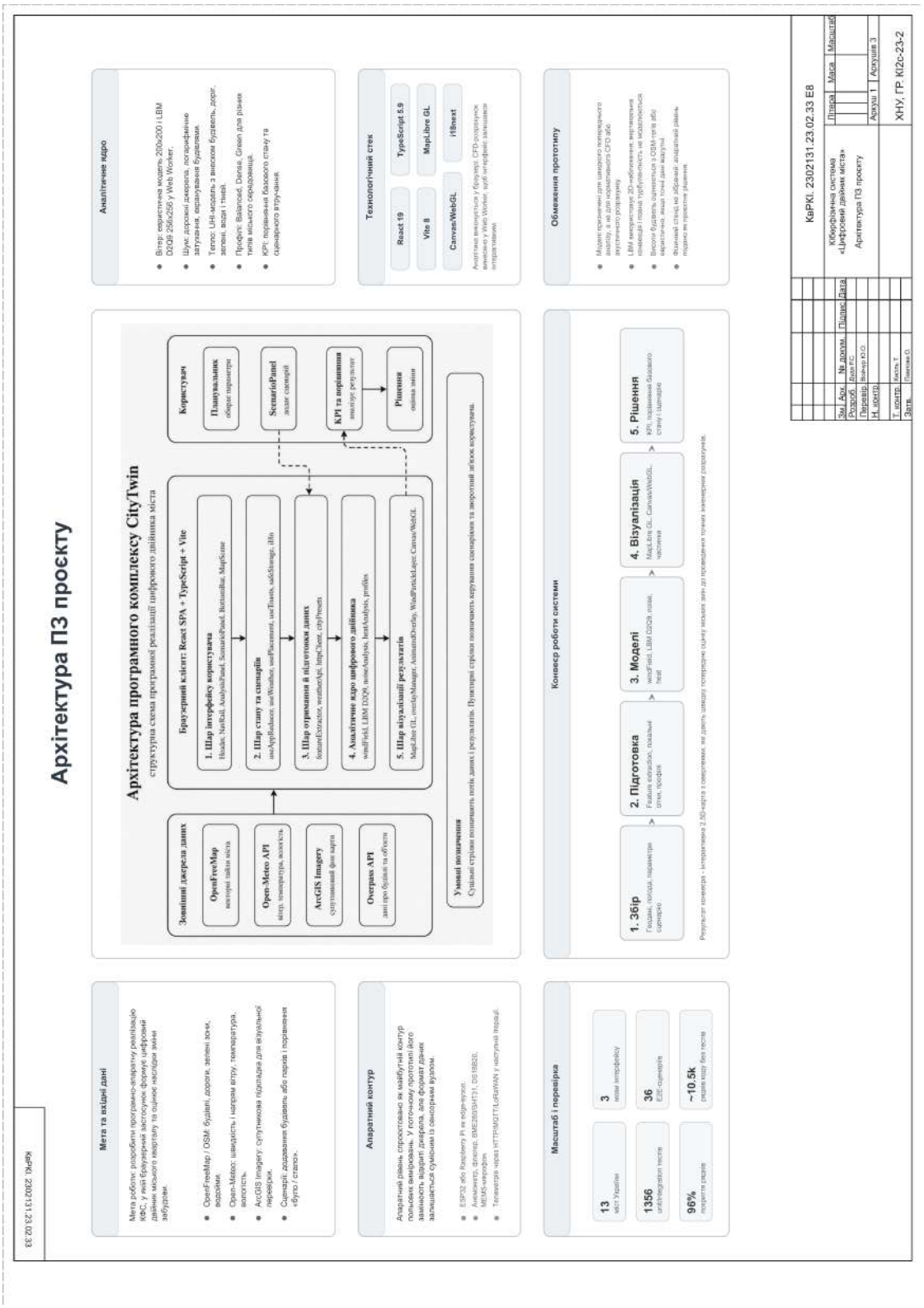
49. Fielding R., Nottingham M., Reschke J. RFC 9110: HTTP Semantics. RFC Editor, 2022. 194 p. DOI: 10.17487/RFC9110. URL: https://www.rfc-editor.org/info/rfc9110/ (дата звернення: 10.06.2026).

50. Liang S., Khalafbeigi T., van der Schaaf H. OGC SensorThings API Part 1: Sensing. Version 1.1. OGC 18-088. Open Geospatial Consortium, 2021. URL: https://docs.ogc.org/is/18-088/18-088.html (дата звернення: 10.06.2026).

ДОДАТОК А

(обов’язковий)

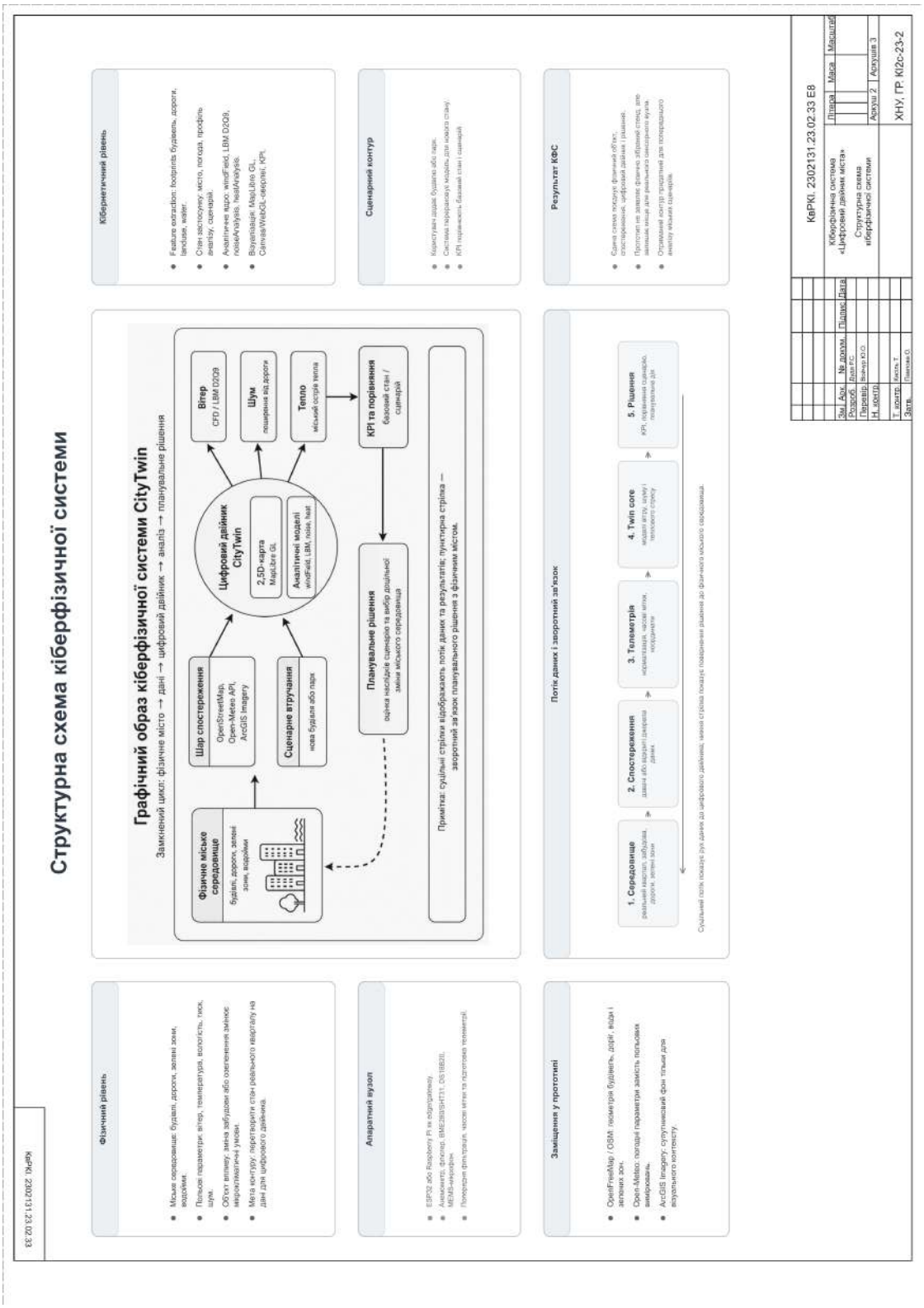
Копія креслення «Архітектура ПЗ проєкту»



ДОДАТОК Б

(обов’язковий)

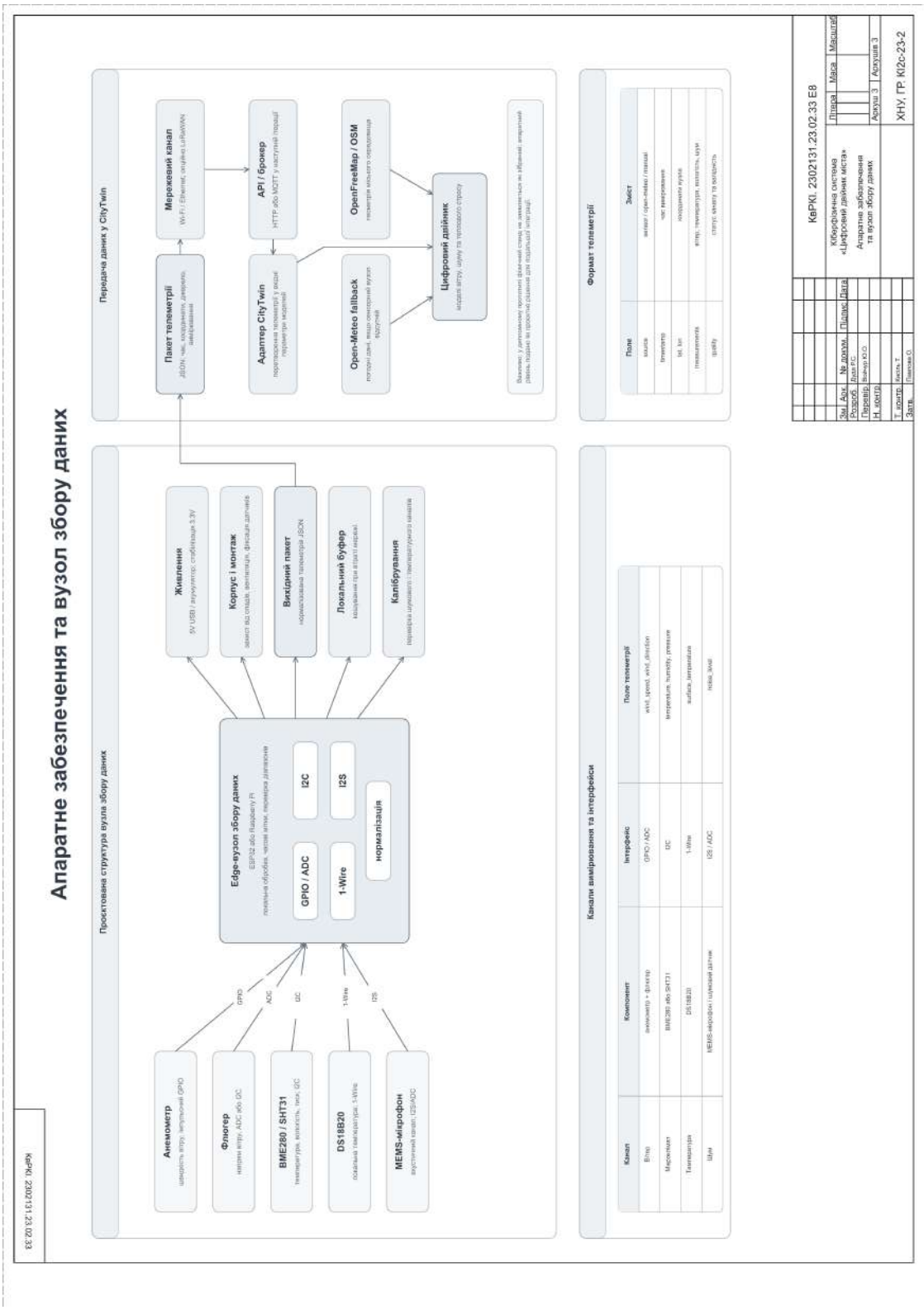
Копія креслення «Структурна схема кіберфізичної системи»



ДОДАТОК В

(обов’язковий)

Копія креслення «Апаратне забезпечення та вузол збору даних»



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: Роман ДУДА

Співавтор:

Назва: 7Дипломна Дуда Роман Сергійович+

Експерт: Юрій ВОЙЧУР

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:5.25%

Коефіцієнт подібності 2:1.15%

Мікропробіли: 3

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-06-13 06:05:03.0

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:

2026-06-13 Доцент Андрій Нічепорук

Дата експерт

Fri Jun 12 21:23:04 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

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

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 275040 Назва: БКР Кіберфізична система “Цифровий двійник міста” Додано в БД: 2026-06-12 Автора: Роман ДУДА Керівники: Юрій ВОЙЧУР Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	112339	971	4781 (4%)	63 (6%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Дуда Роман Сергійович

Тема: Кіберфізична система “Цифровий двійник міста”

Спеціальність: 123 «Комп’ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___63___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розроблення програмно-апаратної реалізації кіберфізичної системи «Цифровий двійник міста» для інтерактивного оцінювання параметрів міського середовища за відкритими геоданими та погодною інформацією. У роботі реалізовано прототип CityTwin, що поєднує карту міста, погодні дані, сценарне редагування забудови і зелених зон, а також моделі вітрового комфорту, шумового навантаження та теплового стресу.

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: В першому розділі кваліфікаційної роботи проведено аналіз предметної області цифрових двійників міського середовища, розглянуто геоінформаційні дані як основу віртуальної моделі міста, методи моделювання вітрового комфорту, шумового навантаження і теплового стресу, а також виконано постановку задачі розроблення системи CityTwin. В другому розділі спроєктовано кіберфізичну систему «Цифровий двійник міста»: визначено вимоги та сценарії використання, побудовано структурну схему системи, описано апаратний рівень і давачі, телеметричні потоки, архітектуру програмного комплексу, організацію даних, моделі аналізу, сценарний редактор міських втручань, інтерфейс користувача та багатомовність. В третьому розділі виконано програмно-апаратну реалізацію і тестування системи CityTwin: реалізовано вебзастосунок на основі React, TypeScript, Vite, MapLibre GL, OpenFreeMap, OpenStreetMap та Open-Meteo;

розроблено модулі карти, витягу просторових об'єктів, чисельної та евристичної моделей вітру, моделей шумового поля і теплового стресу, візуалізації результатів та сценарного порівняння. Результати роботи перевірено unit-, integration- та E2E-тестами, production-збірка застосунку виконана успішно.

4. Позитивні сторони роботи: висока практична цінність роботи, актуальність теми, використання сучасного технологічного стеку, поєднання програмної реалізації з проєктуванням апаратного рівня збору даних, наявність інтерактивного прототипу та комплексного тестування основних функцій системи.

5. Негативні сторони роботи: у межах прототипу не виконано повне калібрування моделей за польовими вимірюваннями, а апаратний вузол збору даних подано переважно як проєктне рішення для подальшої реалізації.

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 належному технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре / C / 73 балів

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

Клюєв Юрій Павлович, к.т.н, доцент

завідувач кафедри кібербезпеки ХНУ

"18" 06 2026 р. _____ (підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Дуда Роман Сергійович

ПІБ здобувача вищої освіти

ФІТ, IV 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.

28 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

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

Назва кваліфікаційної роботи Кіберфізична система «Цифровий двійник міста»
Автор Роман ДУДА
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д.ф. Юрій ВОЙЧУР

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 5,25%; та системою Anti-Plagiarism складає 4%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

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

Керівник кваліфікаційної роботи _____ Юрій ВОЙЧУР
Підпис Ім'я, ПРІЗВИЩЕ