МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут новітніх освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління

Шинкарчук Микола Богданович

Програмний модуль інтеграції даних з сенсорних мереж для контекстної моделі цифрового двійника/Data integration software module from sensor networks for the digital twin context model

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КНзм-21
М. Б.Шинкарчук
_____

Науковий керівник:
к.т.н., доцент Осолінський О.Р.
_____

Кваліфікаційну роботу
допущено до захисту:
«___» _____ 20___ р.
В.о. завідувача кафедри
_____ Н.В. Дзюбановська

ТЕРНОПІЛЬ - 2025

Навчально-науковий інститут новітніх освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____20__р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Шинкарчуку Миколі Богдановичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Програмний модуль інтеграції даних з сенсорних мереж для контекстної моделі цифрового двійника/Data integration software module from sensor networks for the digital twin context model
керівник роботи к.т.н., доцент Осолінський О.Р.
затверджені наказами по університету від 20 грудня 2024 року № 938 та від 28 жовтня 2025 року №1356
2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.

4. Основні питання, які потрібно розробити

– огляд цифрових двійників кіберфізичних систем: поняття, класифікація та архітектури ;
– дослідити IoT-інфраструктури для цифрових двійників та сенсорних мереж;
– проаналізувати підходи до інтеграції даних сенсорних мереж у системи цифрових двійників;
– вибір перспективного шляху і постановка задачі дослідження;
– розробити формальну модель контекстної моделі цифрового двійника;
– розробити метод попередньої обробки та нормалізації даних сенсорних мереж;
– розробити метод побудови та оновлення контекстної моделі цифрового двійника;
– розробити архітектуру програмного модуля інтеграції даних;
– розробити програмний модуль інтеграції;
– розробити інтерфейс взаємодії з контекстною моделлю цифрового двійника;
– провести аналіз результатів експериментальних досліджень;

5. Перелік графічного матеріалу у роботі

– Функціональна схема програмного модуля інтеграції даних сенсорних мереж у цифровий двійник.

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи.	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 20.05.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 28.10. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 11.11.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 25.11.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту.	до 1.12.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 04.12.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії.	до 14.12. 2025 р.	

Студент _____М. Б. Шинкарчук
підпис

Керівник роботи _____ к.т.н., доцент Осолінський О.Р.
підпис

РЕЗЮМЕ

Кваліфікаційна робота на тему «Програмний модуль інтеграції даних з сенсорних мереж для контекстної моделі цифрового двійника» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 76 сторінки і містить 20 ілюстрацій, 4 додатки та 32 використаних джерел.

Метою даної кваліфікаційної роботи є розроблення програмного модуля інтеграції даних з сенсорних мереж для контекстної моделі цифрового двійника та імплементація в нього методу інтеграції, який забезпечує формування контекстної моделі, придатної для моніторингу, діагностики та підтримки прийняття рішень.

Методи досліджень: включають аналіз наукових досліджень існуючих джерел з побудови цифрових двійників та IoT-інфраструктур; методи формального моделювання для побудови контекстної моделі; методи цифрової обробки сигналів і часових рядів для фільтрації, виявлення пропусків і аномалій.

Результати дослідження: розроблено та реалізовано метод попередньої обробки і нормалізації потоків сенсорних даних, що включає фільтрацію шуму, виявлення та компенсацію пропусків, узгодження часових масштабів, видалення аномалій та перетворення вимірювань з ознаками якості.

Результати роботи можуть бути використані як базовий інтеграційний модуль у цифрових двійниках для моніторингу мікроклімату в лабораторних та навчальних стендах, демонстраційних установках і прикладних IoT-системах.

Ключові слова: ЦИФРОВИЙ ДВІЙНИК, СЕНСОРНІ МЕРЕЖІ, ІНТЕРНЕТ РЕЧЕЙ (IoT), КОНТЕКСТНА МОДЕЛЬ, ПОПЕРЕДНЯ ОБРОБКА ТА НОРМАЛІЗАЦІЯ ДАНИХ, EDGE-ОБЧИСЛЕННЯ, RASPBERRY PI, ESP32, МОНІТОРИНГ МІКРОКЛІМАТУ, ІНТЕГРАЦІЯ ДАНИХ СЕНСОРІВ.

ABSTRACT

Qualification thesis titled “ Data integration software module from sensor networks for the digital twin context model” for obtaining the educational degree “Master” in speciality 122 “Computer Science” within the educational program “Computer Science” consists of 76 pages and includes 20 figures, 4 appendices, and 32 references.

The purpose of this qualification paper is to develop a software module for integrating data from sensor networks for a contextual digital twin model and to implement within it an integration method that ensures the formation of a contextual model suitable for monitoring, diagnostics, and decision support.

Research methods include analysis of scientific research and existing sources on building digital twins and IoT infrastructures; methods of formal modeling for constructing a contextual model; methods of digital signal and time series processing for filtering, gap detection, and anomaly detection.

Research results: a method for preprocessing and normalizing streams of sensor data has been developed and implemented. This method includes noise filtering, gap detection and compensation, time scale synchronization, anomaly removal, and transformation of measurements with quality indicators.

The results of the work can be used as a basic integration module in digital twins for monitoring the microclimate in laboratory and educational setups, demonstration systems, and applied IoT systems.

Keywords: DIGITAL TWIN, SENSOR NETWORKS, INTERNET OF THINGS (IoT), CONTEXTUAL MODEL, DATA PREPROCESSING AND NORMALIZATION, EDGE COMPUTING, RASPBERRY PI, ESP32, MICROCLIMATE MONITORING, SENSOR DATA INTEGRATION.

ЗМІСТ

ВСТУП

В даний час стрімко зростає роль цифрових двійників, які забезпечують безперервний моніторинг, аналіз та оптимізацію роботи складних об’єктів і інфраструктур протягом усього життєвого циклу [1-3]. Цифрові двійники впроваджуються в промисловості, енергетиці, транспорті, розумних містах, аграрному секторі та інших галузях, де необхідно в режимі, наближеному до реального часу, відстежувати стан обладнання, прогнозувати відмови та підтримувати прийняття рішень на основі даних[1, 2].

Ключовою передумовою побудови цифрового двійника є наявність розвиненої інфраструктури Інтернету речей (IoT) та сенсорних мереж, які забезпечують безперервний потік телеметрії від фізичних об’єктів до віртуальної моделі [4-6, 13].

Разом з тим практичний досвід розгортання цифрових двійників показує, що на шляху від сирих сенсорних вимірювань до контекстної моделі виникає низка проблем. Потоки даних характеризуються гетерогенністю форматів і одиниць вимірювання, наявністю шуму, пропусків, аномальних значень [11, 12]. Без проміжного шару попередньої обробки та нормалізації ці недоліки призводять до спотворення станів цифрового двійника, помилкового виявлення аномалій, неякісних рішень керування.

Окремо постає проблема переходу від низькорівневих вимірювань до осмислених контекстних змінних, інтегральних станів і подій, які використовуються для моніторингу та підтримки прийняття рішень [7,8].

Отже, актуальною науково-практичною задачею є розроблення способу інтеграції даних сенсорних мереж у цифровий двійник.

Метою роботи є розроблення програмного модуля інтеграції даних з сенсорних мереж для контекстної моделі цифрового двійника та імплементація в нього методу інтеграції, який забезпечує формування контекстної моделі, придатної для моніторингу, діагностики та підтримки прийняття рішень.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

1. Провести аналіз концепції цифрових двійників, архітектур IoT-інфраструктур та існуючих підходів до інтеграції даних сенсорних мереж у цифрові двійники, виявити їх переваги та обмеження.

2. Описати формальну контекстну модель для цифрового двійника.

3. Розробити підхід оновлення контекстної моделі цифрового двійника,.

4. Реалізувати архітектуру програмного модуля взаємодії з цифровим двійником.

5. Створити програмний модуль для підключення сенсорних даних до цифрового двійника та інструменти для їх візуального моніторингу майже в реальному часі.

6. Виконати серію експериментальних досліджень на лабораторному стенді, що включає мережу вузлів ESP32 та Raspberry Pi, з метою порівняльного аналізу якості та структурованості даних у цифровому двійнику, а також оцінки впливу запропонованої методики на точність моніторингу та ефективність детектування аномалій.

Об’єктом дослідження є процес інтеграції даних сенсорних мереж у цифровий двійник на рівні edge-інфраструктури.

Предметом дослідження є методи та програмні засоби попередньої обробки, нормалізації й контекстної інтерпретації потоків телеметричних даних.

Методи дослідження включають аналіз наукових досліджень існуючих джерел з побудови цифрових двійників та IoT-інфраструктур; методи формального моделювання для побудови контекстної моделі; методи цифрової обробки сигналів і часових рядів для фільтрації, виявлення пропусків і аномалій;

Практичне значення одержаних результатів полягає в тому, що розроблений програмний модуль інтеграції даних сенсорних мереж може бути використаний як базовий компонент цифрових двійників для лабораторних стендів, демонстраційних установок і прикладних IoT-систем.

Структура та обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел.

Апробація результатів дослідження. Основні теоретичні положення роботи й практичні результати дослідження доповідалися й обговорювалися на IX Всеукраїнській студентській конференції «Науковий простір: аналіз, сучасний стан, тренди та перспективи», м. Вінниця, Україна та VI Міжнародній науковій конференції, «Теорія модернізації в контексті сучасної світової науки» м. Івано-Франківськ, Україна.

1 АНАЛІЗ ЦИФРОВИХ ДВІЙНИКІВ ТА ІНТЕГРАЦІЇ ДАНИХ СЕНСОРНИХ МЕРЕЖ

1.1 Цифрові двійники кіберфізичних систем: поняття, класифікація та архітектури

В сучасних кіберфізичних системах все більшу роль відіграють цифрові представлення реальних об’єктів, процесів та інфраструктур [1,6]. Поступове ускладнення виробничих, транспортних, енергетичних та міських систем, а також масове впровадження інтернету речей призводить до зростання обсягів даних і вимог до їх аналізу в режимі, наближеному до реального часу. Тому концепція цифрового двійника розглядається, як ключова технологія для моніторингу, прогнозування та оптимізації поведінки складних об’єктів протягом усього життєвого циклу [1, 3].

Перші формулювання поняття цифрового двійника пов’язують з роботами М. Грівса та Д. Віккерса, які описували його як поєднання трьох складових: фізичного об’єкта, відповідного віртуального подання та безперервного потоку даних між цими двома сутностями [1].

В такому підході цифровий двійник не є статичною моделлю, а виступає динамічною структурою, яка постійно оновлюється на основі вимірювань з сенсорів, журналів подій та інших джерел інформації [2, 3]. Це дозволяє виконувати аналіз поточного стану, моделювати можливі сценарії розвитку подій, оцінювати ризики та приймати керуючі рішення до того, як відбудуться зміни у фізичній системі.

Щоб краще показати місце цифрового двійника серед інших цифрових подань, в літературі пропонується тріада «цифрова модель – цифрова тінь – цифровий двійник». В літературі описуються, що цифрова модель описує об’єкт або процес, але не пов’язана з ним потоком даних у реальному часі. Це можуть бути, наприклад, тривимірні геометричні моделі, математичні чи імітаційні моделі, які застосовують на етапі проектування або аналізу.

Цифрова тінь (digital shadow) отримує дані від фізичної системи, але цей зв’язок має односторонній напрямок, тобто інформація надходить від об’єкта до цифрового представлення, тоді як зворотній вплив або контроль відсутній [3]. Така тінь дозволяє виконувати моніторинг, звітність, візуалізацію та базову аналітику стану, але не передбачає активного втручання в роботу фізичного об’єкту.

Цифровий двійник, на відміну від цифрової тіні, характеризується двосторонньою взаємодією між фізичною та віртуальною частинами [1-3]. Дані з сенсорів в реальному часі оновлюють стан віртуальної моделі, а результати аналізу, оптимізації або прогнозування, виконані у цифровому середовищі, можуть впливати на керування фізичною системою. Такий двосторонній зв’язок дозволяє не лише «спостерігати», а і активно змінювати поведінку об’єкта, досягаючи кращої якості керування, підвищення надійності, економії ресурсів чи енергетичної ефективності.

Цифрові двійники часто розглядають у контексті кіберфізичних систем [6]. Кіберфізична система об’єднує фізичні компоненти, такі як машини, пристрої, сенсори, виконавчі механізми з програмним забезпеченням, мережевою інфраструктурою та сервісами обробки даних. В оглядах підкреслюється, що як кіберфізичні системи, так і цифрові двійники базуються на тісній взаємодії фізичного та цифрового світів, реальному часі та інтеграції різнорідних джерел даних [4, 6]. Водночас цифровий двійник акцентує увагу саме на створенні узгодженого цифрового «образу» об’єкта, який супроводжує його на всіх етапах життєвого циклу і використовується як інструмент аналізу та прийняття рішень [2, 3].

В різних прикладних доменах виділяють декілька основних типів цифрових двійників: двійники компонентів; активів; систем та процесів [2 ,3]. Таке розділення відображає масштаб об’єкта моделювання та глибину відображення його поведінки. Для окремих галузей також пропонують класифікацію за рівнем складності: описові двійники, що відображають

поточний стан, прогностичні двійники, які дозволяють оцінювати майбутні стани, і ті що формують рекомендації або управляючі впливи [2].

Крім поділу за масштабом, розглядають також багатовимірні представлення цифрових моделей. Ідеї n-вимірних моделей виникли у будівельній галузі у контексті інформаційного моделювання будівель, де до тривимірної геометрії послідовно додають часовий вимір, витрати, показники експлуатації, стійкості тощо [2]. В сучасних роботах цифровий двійник розглядають, як подальший етап розвитку таких багатовимірних моделей, де до геометрії та параметрів додаються потоки даних в режимі реального часу, знання про стани системи та алгоритми керування [1-3, 6].

З архітектурної точки зору цифровий двійник описують як багатошарову систему (рисунок 1.1) [2, 6, 7].

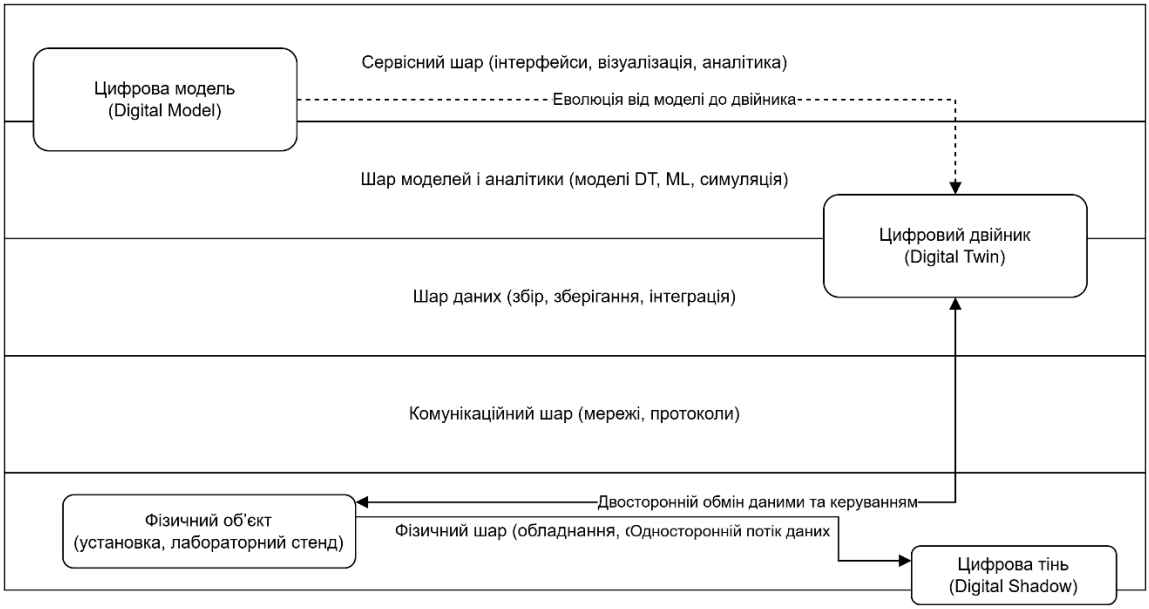


Рисунок 1.1 – Взаємозв'язок між цифровою моделлю, цифровою тінню та цифровим двійником у складі кіберфізичної системи

Типова структура включає фізичний, комунікаційний шар, шар даних, модельно-аналітичний шар та сервісний шар. Також виділяють додаткові шари, такі як спеціальний «шар даних» або «шар симуляції», які забезпечують гнучке підключення різних аналітичних сервісів до ядра цифрового двійника.

Ключовим елементом, який об’єднує всі ці шари, є надійний потік даних між фізичною системою та віртуальною моделлю. Саме якість, повнота, актуальність і контекстна інтерпретація даних визначають, наскільки точно цифровий двійник відображає реальний об’єкт і наскільки корисними будуть результати аналізу [7, 8]. В сучасних кіберфізичних системах основним джерелом даних є сенсорні мережі на базі мікроконтролерів, одноплатних комп’ютерів та різноманітних датчиків [4, 5, 13]. Потоки таких даних часто характеризуються гетерогенністю форматів, нестабільною затримкою та можливими пропусками, що ускладнює їх безпосереднє використання в контекстних моделях цифрових двійників [11, 12].

1.2 IoT-інфраструктура для цифрових двійників та сенсорні мережі

Цифровий двійник неможливо реалізувати без надійного потоку даних. Цей потік забезпечують сенсорні мережі та інфраструктура Інтернету речей, які виступають [4-6, 13]. Саме через них цифровий двійник отримує вимірювання про стан об’єкта, довкілля та технологічних процесів, і у відповідь може генерувати сигнали управління або рекомендації.

Сенсорна мережа — це сукупність просторово розподілених сенсорних вузлів, які вимірюють фізичні параметри, такі як температура, вологість, тиск, освітленість або стан обладнання та передають дані до центрального вузла або сервера для подальшого аналізу [5]. В сучасних визначеннях підкреслюється, що такі мережі складаються з малопотужних пристроїв із вбудованими сенсорами, обчислювальними ресурсами та бездротовим модулем зв’язку, які разом формують розподілену систему реального часу [4, 5].

З розвитком IoT сенсорні мережі перестали бути ізольованими системами. Вони інтегруються з глобальними або приватними мережами Інтернету, хмарними сервісами та цифровими платформами, фактично стаючи складовою частиною ширшої IoT-екосистеми [4, 6, 13]. У такому контексті WSN розглядається як підсистема IoT, що відповідає за первинне збори даних

на рівні «периферії» (edge), тоді як інші компоненти IoT-архітектури забезпечують транспортування, зберігання та обробку цих даних [4].

Для побудови сенсорних мереж на практиці все ширше застосовуються доступні апаратні платформи, наприклад, мікроконтролери ESP32 та одноплатні комп'ютери Raspberry Pi [13].

ESP32 поєднує у собі обчислювальні ядра, модулі Wi-Fi/Bluetooth та набір інтерфейсів для підключення датчиків, що робить його типовим «сенсорним вузлом» IoT-системи [13].

Raspberry Pi, навпаки, має значно більші обчислювальні можливості, підтримує повноцінну операційну систему та мережеві сервіси, тому часто використовується як IoT-шлюз або edge-вузол, який об'єднує декілька сенсорних вузлів, проводить локальну обробку даних і взаємодіє з хмарою чи цифровим двійником [4, 6]. Сучасні дослідження показують, що Raspberry Pi як IoT-шлюзу та вузла для крайових обчислень, зокрема для сценаріїв цифрових двійників у промисловості.

У такій конфігурації сенсорна мережа може бути організована як набір вузлів ESP32, до яких під'єднані різні датчики, що вимірюють цільові параметри. Кожен із цих вузлів збирає дані та по бездротовому каналу передає їх до Raspberry Pi, який виконує роль локального концентратора даних. Далі Raspberry Pi може або безпосередньо оновлювати контекстну модель цифрового двійника, або передавати агреговані дані до хмарної платформи, де розташована більш комплексна модель.

Ключовим питанням для IoT-інфраструктури цифрового двійника є вибір протоколів і механізмів обміну даними між вузлами сенсорної мережі, шлюзом та сервісами верхніх рівнів. У цьому контексті широкого поширення набув протокол MQTT [9]. Він є полегшеним протоколом обміну повідомленнями у моделі «publish/subscribe», спеціально розробленим для пристроїв. У цій моделі сенсорні вузли публікують повідомлення в певні топіки, брокер MQTT розподіляє їх, а зацікавлені сервіси (передплатники) отримують лише ті дані, які їм потрібні.

Крім MQTT, в сенсорних мережах і IoT-інфраструктурі застосовуються й інші протоколи — CoAP, HTTP/HTTPS, різні варіанти промислових протоколів поверх IP-мереж, а також спеціалізовані технології зв’язку, такі як LoRaWAN, NB-IoT, ZigBee та Bluetooth Low Energy. Вибір конкретного рішення залежить від радіуса дії, вимог до енергоспоживання, пропускної здатності, надійності та безпеки. Для лабораторних стендів та малих цифрових двійників, які будуються на базі ESP32 та Raspberry Pi (рисунок 2.1), достатньо стандартної Wi-Fi-мережі та MQTT-шлюзу [9, 13].

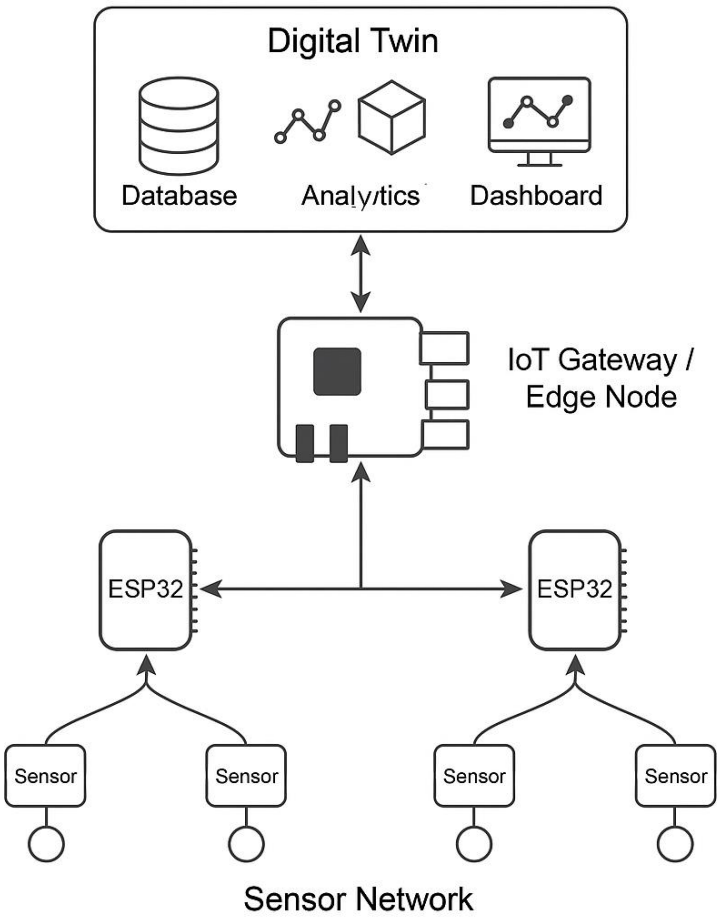


Рисунок 1.2 – Узагальнена архітектура сенсорної мережі та IoT-інфраструктури цифрового двійника на базі ESP32 і Raspberry Pi.

В контексті цифрових двійників сенсорні мережі й IoT-інфраструктура виконують декілька взаємопов’язаних функцій. По-перше, вони забезпечують безперервний збір даних про стан фізичної системи з достатньою просторово-

часовою роздільною здатністю. По-друге, за рахунок використання шлюзів та edge-вузлів частина обробки переноситься ближче до джерела даних. Це зменшує затримки, знижує навантаження на мережу та дозволяє швидше виявляти аномалії або критичні стани. По-третє, стандартизовані протоколи обміну спрощують інтеграцію різнорідних сенсорних вузлів у єдину контекстну модель цифрового двійника, де вимірювання перетворюються на логічні стани, події та індикатори. Така інфраструктура створює основу для зворотного зв’язку: цифровий двійник може надсилати керувальні команди або рекомендації назад на периферію, де вони реалізуються через виконавчі механізми.

1.3 Підходи до інтеграції даних сенсорних мереж у системи цифрових двійників

Ефективність цифрового двійника безпосередньо залежить від того, яким чином дані із сенсорних мереж потрапляють до його внутрішніх моделей [4, 6, 8]. Інтеграція даних визначає не лише технічні аспекти сумісності протоколів та форматів, але і можливості подальшої аналітики, масштабування системи та реалізації контекстної інтерпретації. На практиці побудови цифрових двійників можна виокремити кілька базових підходів до інтеграції даних сенсорних мереж: пряме підключення, використання брокерів повідомлень та інтеграція через спеціалізовані IoT-платформи. Кожен із них має свої переваги та обмеження і крім того по-різному впливає на вирішення проблем гетерогенності, затримок, якості даних і контекстної інтерпретації.

Під прямою інтеграцією (рисунок 1.3) розуміють такий підхід, за якого сенсорні вузли або шлюзи безпосередньо взаємодіють з компонентами цифрового двійника через стандартні мережеві протоколи – найчастіше HTTP/HTTPS або власні TCP/UDP-з’єднання. У найпростішому варіанті кожен вузол періодично надсилає вимірювання до REST-інтерфейсу сервера,

де розгорнуто модулі цифрового двійника, або відкриває постійне з’єднання для передачі потокових даних.

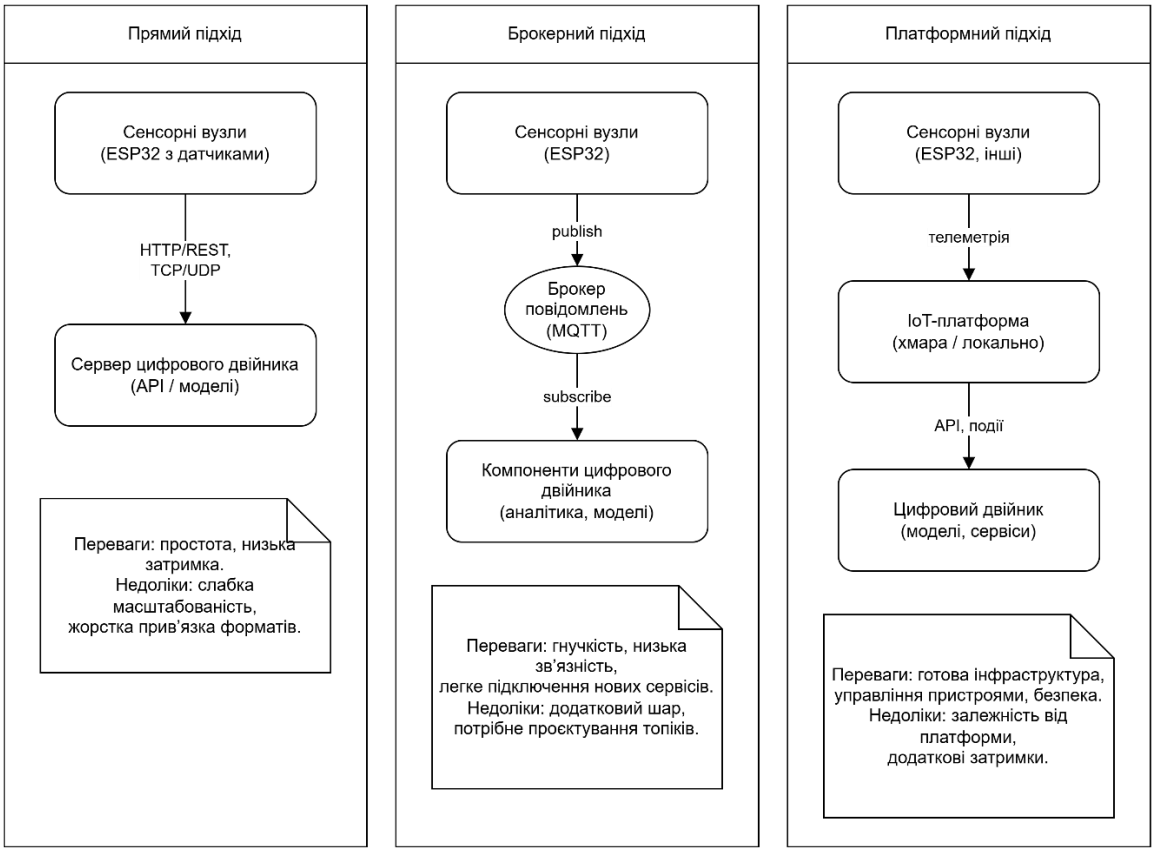


Рисунок 1.3 – Підходи до інтеграції даних сенсорних мереж у цифрові двійники

Перевага цього підходу полягає в його простоті, томущо структура даних та логіка обміну можуть бути максимально прозорими, а затримки – мінімальними, оскільки немає проміжних посередників.

Але з ростом кількості сенсорних вузлів пряме підключення створює низку проблем. По-перше, сервер цифрового двійника змушений обробляти велику кількість однотипних з’єднань, що ускладнює масштабування та балансування навантаження. По-друге, різні сенсорні вузли можуть використовувати різні формати повідомлень, що призводить до ускладнення логіки обробки даних великою кількістю перетворень. По-третє, будь-які зміни в протоколі або форматі на стороні сенсора потребують модифікації

серверної частини. В результаті прямий підхід доцільний переважно в невеликих стендах або експериментальних системах з обмеженою кількістю пристроїв, де гнучкість і простота реалізації важливіші за масштабованість.

Другий підхід – інтеграція на основі брокера повідомлень. У цьому випадку між сенсорними вузлами та модулем цифрового двійника додається проміжна ланка – брокер, який реалізує модель обміну «publish/subscribe». Сенсорні вузли публікують повідомлення у визначені топіки, а цифровий двійник або окремі його компоненти підписуються на потрібні теми і отримують лише ті дані, які їм необхідні.

Такий підхід дозволяє зменшити зв’язність між сенсорними вузлами та цифровим двійником. Вузли не мають інформації про структуру серверних компонентів, а лише про адресу брокера і ім’я теми. Це спрощує додавання нових сенсорів, зміну конфігурації, підключення додаткових споживачів. До того ж брокер може забезпечувати буферизацію повідомлень, базову політику якості доставки та прості механізми безпеки.

Разом із тим брокерний підхід вимагає більш складнішого проектування структури тем, формату повідомлень та політик підписки. Якщо ці аспекти не продумані заздалегідь, система може стати важкокерованою при масштабуванні. Окрім того, брокер створює додатковий рівень, на якому також потрібно контролювати продуктивність, затримки й надійність. Цей варіант часто розглядають, як базова інфраструктура для інтеграції сенсорних мереж, оскільки він поєднує достатню гнучкість з можливістю побудови модульної архітектури, де різні сервіси цифрового двійника можуть незалежно споживати дані.

Третій підхід – використання повноцінних IoT-платформ, як проміжної ланки між сенсорними мережами та цифровим двійником [4, 6, 13]. У цьому випадку дані від сенсорних вузлів потрапляють спочатку до хмарної або локальної IoT-платформи, а вже звідти надходять до модулів цифрового двійника через стандартизовані API. Така платформа надає вбудовані засоби для управління пристроями, аутентифікації, шифрування, зберігання

історичних даних, а також може містити попередньо реалізовані механізми аналітики та візуалізації.

Перевага платформного підходу полягає у тому, що значна частина складних інфраструктурних задач розв’язується за рахунок готових компонентів платформи. Розробнику цифрового двійника не потрібно самостійно будувати весь стек, вся робота проводиться на рівні логіки предметної області. Але це призводить до залежності від конкретного постачальника платформи, необхідності враховувати її обмеження і потенційні затримки, пов’язані з передаванням даних через хмарну інфраструктуру.

Окрему групу питань становлять проблеми гетерогенності, затримок, якості та контекстної інтерпретації даних, які виникають незалежно від обраного підходу інтеграції [7, 8, 11]. Гетерогенність проявляється як на рівні апаратних засобів, так і на рівні даних, наприклад, різні одиниці вимірювання, формати повідомлень, структури пакетів. Щоб обійти ці обмеження необхідні уніфіковані схеми даних, механізми нормалізації та опису метаданих, а також чітко визначені інтерфейси між сенсорним рівнем та ядром цифрового двійника.

Затримки та нестабільність каналів зв’язку безпосередньо впливають на здатність цифрового двійника відображати реальний стан системи. Якщо дані надходять із суттєвою затримкою, виникає розсинхронізація між фізичним і віртуальним представленням, що знижує корисність моделі для задач прогнозування та керування. Тому в архітектурі інтеграції завжди необхідно враховувати часові вимоги, механізми буферизації, можливість обробки пропусків та реконструкції часових рядів.

Якість даних включає не лише точність окремих вимірювань, а і повноту, достовірність, відсутність дублювання, наявність коректних міток часу та контекстної інформації. Без цієї інформації цифровий двійник не може коректно інтерпретувати значення параметрів, діагностувати відмови чи виявляти аномалії. Саме тому значна частина логіки інтеграційних модулів

присвячена попередній обробці даних – фільтрації шуму, перевірці діапазонів, виявленню збоїв.

Контекстна інтерпретація даних є ключовим кроком, який перетворює сирі значення сенсорів на осмислені стани, події та показники, що використовуються цифровим двійником. Вона може базуватися як на правилах, так і на алгоритмах машинного навчання, які навчаються за історичними даними [11, 12]. Вибір підходу до інтеграції даних визначає, де саме реалізуються ці механізми: безпосередньо в цифровому двійнику, на рівні брокера чи всередині IoT-платформи.

1.4 Вибір перспективного шляху і постановка задачі дослідження

Аналіз сучасних підходів до побудови цифрових двійників показав, що ключовою передумовою їх ефективного функціонування є надійна інтеграція даних сенсорних мереж у контекстну модель двійника.

Пряме підключення сенсорних вузлів забезпечує мінімальні затримки, проте швидко втрачає керованість при зростанні кількості пристроїв і вимагає рознесення складної логіки обробки на серверну частину [4]. Брокерний підхід на основі MQTT полегшує масштабування, але потребує продуманої структури тем, формату повідомлень і додаткових сервісів для забезпечення якості та цілісності потоків даних. Використання хмарних IoT-платформ дозволяє перекласти частину інфраструктурних задач на постачальника сервісів, однак призводить до залежності від конкретної платформи та додаткових мережевих затримок, що є критичним для компактних лабораторних цифрових двійників та edge-сценаріїв.

Крім того незалежно від обраної схеми інтеграції сенсорні потоки характеризуються шумом, пропусками, аномаліями, неоднорідністю одиниць вимірювання та часовою розсинхронізацією. Без проміжного шару попередньої обробки такі дані не можуть безпосередньо використовуватися контекстною моделлю цифрового двійника, оскільки призводять до

помилкових висновків, некоректного виявлення аномалій та низької довіри до прогнозів. Це вимагає розроблення формалізованого методу нормалізації, фільтрації та позначення якості даних, а також чітко визначеної контекстної моделі, що перетворює числові вимірювання на стани, події та високорівневі індикатори.

Отже перспективним шляхом розвитку є побудова легковагового інтеграційного модуля на рівні edge-вузла (Raspberry Pi), який:

– приймає телеметрію від сенсорних вузлів ESP32;

– виконує формалізовану попередню обробку та нормалізацію потоків даних;

– реалізує контекстну модель цифрового двійника з виділенням станів і подій;

– надає стандартизований програмний інтерфейс для подальшого використання цих даних у системах моніторингу та керування.

Виходячи з проведеного аналізу, виявлено, що є необхідність підвищення якості та інформативності даних цифрового двійника за рахунок розроблення методу інтеграції потоків сенсорних вимірювань, який поєднує попередню обробку, нормалізацію, контекстну інтерпретацію та програмну реалізацію на рівні edge-шлюзу.

Метою даної роботи є розроблення програмного модуля інтеграції даних з сенсорних мереж для контекстної моделі цифрового двійника та імплементації в нього методу інтеграції даних сенсорних мереж, який забезпечує формування контекстної моделі, придатної для моніторингу, діагностики та підтримки прийняття рішень.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Сформувати формальну контекстну модель цифрового двійника.

2. Удосконалити метод попередньої обробки та нормалізації даних сенсорних мереж та підхід до побудови та оновлення контекстної моделі цифрового двійника, який перетворює нормалізовані вектори вимірювань на контекстні змінні, інтегральні стани та події.

3. Розробити архітектуру програмного модуля інтеграції даних на базі Raspberry Pi та ESP32, сервісів прийому, обробки та інтерфейсів взаємодії з цифровим двійником.

4. Реалізувати програмний модуль інтеграції даних сенсорних мереж у цифровий двійник та розробити засоби візуалізації для моніторингу контекстних змінних і станів у режимі, близькому до реального часу.

5. Провести експериментальні дослідження на лабораторному стенді з кількома вузлами ESP32 та Raspberry Pi, порівняти якість і структурованість даних цифрового двійника, оцінити вплив запропонованого підходу на можливості моніторингу та виявлення аномалій.

Висновки до розділу 1

1. Проаналізовано сучасні концепції про цифрові двійники. Показано еволюцію від звичайних цифрових моделей до цифрової тіні та цифрового двійника.

2. Проаналізовано багатошарову архітектуру цифрового двійника, яка включає фізичний, комунікаційний, шар даних, модельно-аналітичний та сервісний шари.

3. Проаналізовано роль сенсорних мереж та IoT-інфраструктури при реалізації цифрових двійників.

4. Проаналізовано три базові підходи до інтеграції даних сенсорних мереж у системи цифрових двійників

5. Визначено проблеми гетерогенності, затримок та якості даних, що виникають незалежно від обраного варіанта інтеграції.

6. На основі проведеного аналізу обґрунтовано доцільність розроблення легковагового інтеграційного модуля на рівні edge-вузла (Raspberry Pi), який би приймав дані від вузлів ESP32, виконував їх формалізовану попередню обробку, нормалізацію та контекстну

інтерпретацію, а також надавав стандартизовані інтерфейси доступу до сформованої контекстної моделі.

2 МЕТОД ІНТЕГРАЦІЇ ДАНИХ СЕНСОРНИХ МЕРЕЖ У ЦИФРОВИЙ ДВІЙНИК

2.1 Формальна контекстна модель цифрового двійника

Цифровий двійник зберігає не тільки сирі телеметричні дані, які отримані від сенсорної мережі. Для прийняття рішень, виявлення аномалій та підтримки керування необхідний перехід від числових значень параметрів до осмислених станів, подій та ситуацій. Цей перехід забезпечується контекстною моделлю цифрового двійника, яка описує, як інтерпретувати вимірювання в термінах предметної області [7, 8].

Під контекстною моделлю цифрового двійника розуміється формальна структура, яка пов’язує сенсорні дані, часові мітки, метадані про джерела вимірювань, знання про об’єкт моніторингу та правила інтерпретації в єдину систему. Така модель дозволяє визначити, в якому стані перебуває об’єкт, які події відбуваються, які ризики виникають і які керуючі команди можуть бути доцільними.

Нехай T–множина моментів часу, у які здійснюються вимірювання. Тоді через S можна позначити множину сенсорних вузлів $S=\{s_1,s_2,\ldots,s_n\}$, а через P позначити множину фізичних параметрів, що вимірюються $P=\{p_1,p_2,\ldots,p_m\}$, наприклад, температура, вологість повітря, освітленість та вологість ґрунту.

Тоді сирі дані сенсорної мережі можна формально розглядати, як відображення де значення NaN використовується для позначення пропусків або некоректних вимірювань.

$$D_{raw}: S\times P\times T\rightarrow R\cup\{NaN\}, \tag{2.1}$$

Після попередньої обробки і нормалізації отримується множина нормалізованих вимірювань.

$$X(t)=(x_1(t),x_2(t),\ldots,x_m(t)),t\in T, \tag{2.2}$$

де $xj(t)$- значення параметра pj в момент часу t, приведене до формату, узгодженого масштабу та пов'язане з відповідними метаданими, які можуть являти собою ідентифікатор сенсора, якість вимірювання та місце встановлення.

Для опису контексту можна ввести множину контекстних змінних

$$C=\{c_1,c_2,\ldots,c_k\}, \tag{2.3}$$

кожна з яких відображає певний аспект ситуації. В задачах моніторингу середовища це можуть бути змінні:

c_1 – рівень теплового комфорту,

c_2 – ризик перегріву,

c_3 – ризик конденсації,

c_4 – якість вимірювань тощо.

Кожна контекстна змінна c_i має власну область значень Dom(c_i), яка може бути числовою, логічною або категорією. Наприклад, Dom(c_2)={низький, середній, високий}

Формально контекстний вектор в момент часу t можна подати як

$$C(t)=(c_1(t),c_2(t),\ldots,c_k(t)), \tag{2.4}$$

де кожен компонент $c_i(t)\in$Dom(c_i) обчислюється на основі нормалізованих вимірювань $X(t)$ та додаткових знань про систему.

Також в модель можна ввести відображення

$$f_{ctx}:X(t)\times M\rightarrow C(t), \tag{2.5}$$

де, M– множина метаданих до яких входять порогові значення, допустимі діапазони, конфігурація об'єкта, інформація про розташування сенсорів.

Тобто, f_{ctx} задає правила інтерпретації, які перетворюють сирі числові дані на контекстні оцінки.

Множину можливих станів цифрового двійника можна позначити як

$$\Sigma=\{\sigma_1,\sigma_2,\ldots,\sigma_r\}, \tag{2.6}$$

де, кожен стан σ_i відображає певну конфігурацію контекстних змінних і може відповідати, нормальному режиму, попереджувальному режиму або критичному стану. Для формального зв'язку між контекстним вектором і станами можна ввести

$$g_{state}: C(t)\rightarrow\Sigma, \tag{2.7}$$

яке на основі значень контекстних змінних визначає поточний стан цифрового двійника. В найпростішому випадку g_{state} може бути набором правил типу «якщо–то», але в загальному вигляді це може бути класифікаційна модель, побудована за даними спостережень.

Окремо розглядається множина подій

$$\text{E}=\{e_1,e_2,\ldots,e_q\}, \tag{2.8}$$

які описують важливі зміни в поведінці системи: вихід параметра за допустимий діапазон, перехід із нормального стану до попереджувального, відновлення після аварійної ситуації. Формально подію можна подати як пару ($\sigma_{\text{до}}$, $\sigma_{\text{після}}$) разом з часовим інтервалом $[t_1,t_2]$, у який відбувся перехід, та умовами його ініціювання. Тоді контекстна модель включає функцію переходів

$$h_{\text{event}}:\Sigma\times C(t)\rightarrow\Sigma\times\text{E}^*, \tag{2.9}$$

де E^* – підмножина подій, які генеруються при зміні стану.

Схематичну структуру формальної контекстної моделі цифрового двійника наведено на рисунку 2.1.

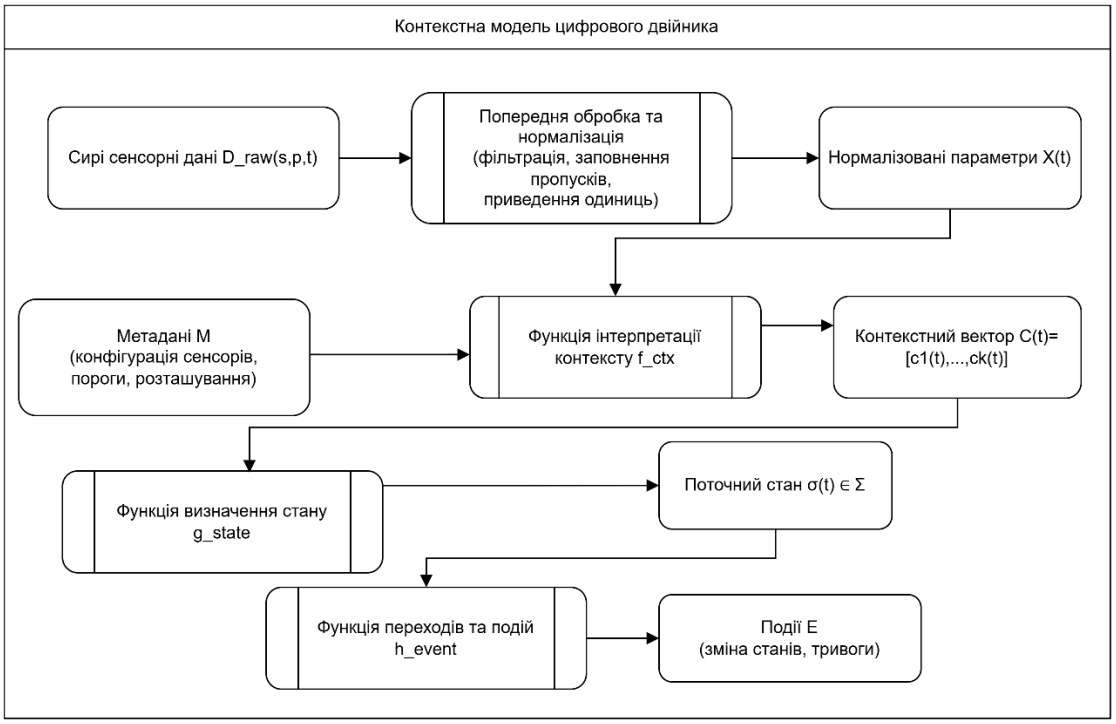


Рисунок 2.1 – Формальна контекстна модель цифрового двійника

Відносно структури зберігання даних контекстну модель можна реалізувати в декількох формах. Один з поширених варіантів – таблична модель, де для кожного моменту часу або для кожної сутності фіксується вектор параметрів, контекстних змінних, поточний стан і пов’язані події. Такий підхід легко реалізується у реляційних базах даних або сховищах часових рядів.

Інший варіант – онтологічна модель, в якій сутності, такі як сенсори, параметри, контекстні змінні, стани подаються як класи й екземпляри, а зв’язки між ними описуються у вигляді відношень «вимірює», «належить зоні», «характеризує стан», «ініціює подію». Цей підхід спрощує семантичну інтеграцію даних з різних джерел і дозволяє використовувати запити високого рівня.

Ще одна форма – граф станів, де вершинами є стани Σ, а дугами – можливі переходи між ними, позначені умовами активації, пов’язаними з контекстними змінними. В такій моделі контекстні дані інтерпретуються, як послідовність переходів по графу, що відображає еволюцію об’єкта в часі.

Тобто потрібно так побудувати контекстну модель цифрового двійника щоб вона могла спиратися на формальні визначення множин параметрів, контекстних змінних, станів і подій та бути придатною до реалізації у вигляді таблиць і структур, що зберігаються на шлюзі та можуть оновлюватися в режимі, наближеному до реального часу.

Виходячи з вище сказаного, формальна модель контексту задає основу, на якій будуть визначені конкретні алгоритми попередньої обробки, нормалізації та інтерпретації даних сенсорних мереж у контексті цифрового двійника.

2.2 Метод попередньої обробки та нормалізації даних сенсорних мереж

Ефективність контекстної моделі цифрового двійника залежить від якості вхідних даних, які надходять з сенсорних мереж. В реальних умовах вимірювані сигнали містять шум, пропуски, аномальні значення, часові зсуви, різні одиниці вимірювання та формати представлення. Тому між шаром збору телеметрії і моделлю контексту необхідно розміщувати модуль попередньої обробки та нормалізації, який приводить потік даних до єдиного канонічного вигляду [11-13].

В загальному випадку вхід до методу попередньої обробки утворює потік сирих вимірювань

$$D_{\text{raw}}(s,p,t),$$

де s– ідентифікатор сенсорного вузла;

p– параметр температури, вологості або освітлення;

t– момент часу.

Дані можуть надходити у вигляді MQTT-повідомлень, HTTP-запитів або двійкових пакетів. Першим кроком алгоритму є перевірка структури: контроль

формату повідомлення, наявності обов’язкових полів, коректності часової мітки та ідентифікаторів сенсорів. Пакети, що не відповідають очікуваному формату, відкидаються з реєстрацією помилки в журналі, щоб не потрапити до подальших етапів обробки (рисунок2.2) [31].

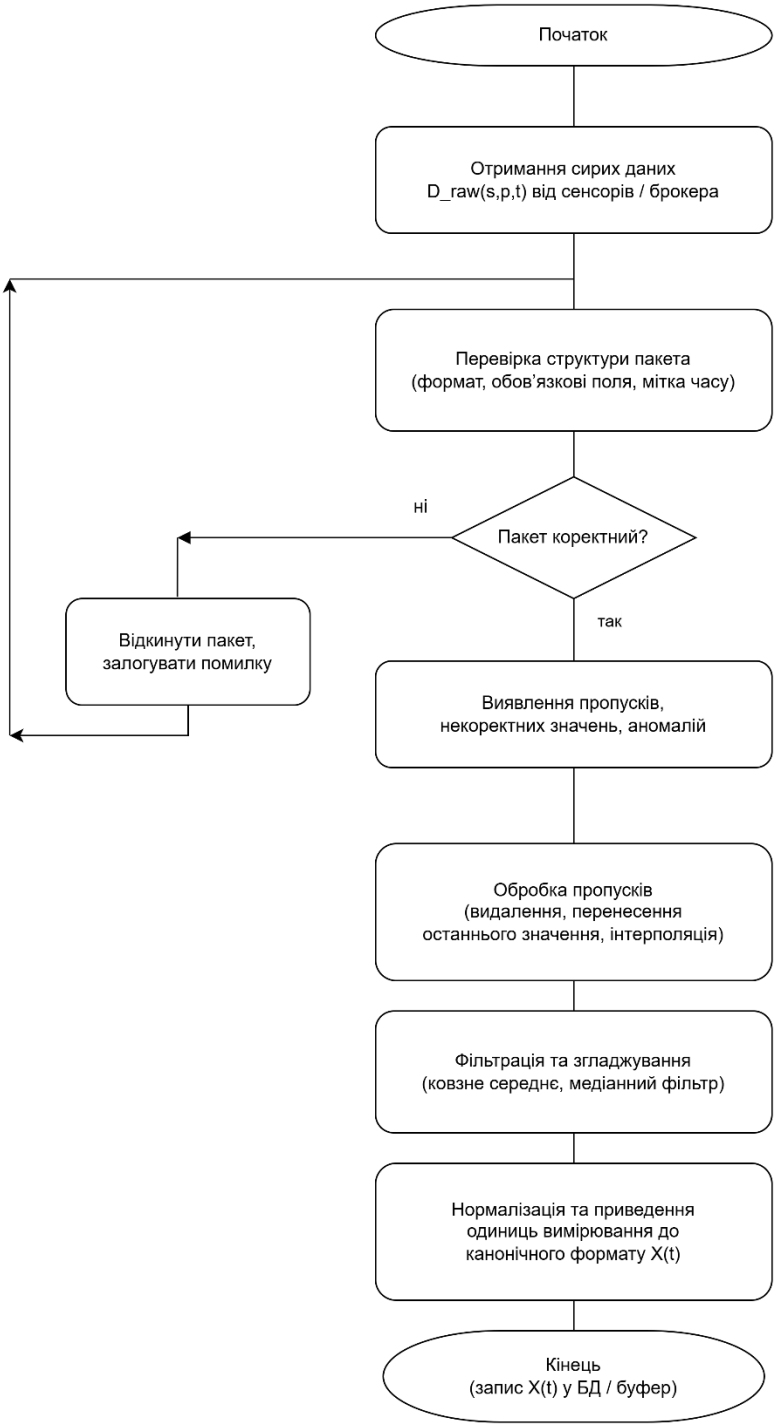


Рисунок 2.2 – Алгоритм попередньої обробки та нормалізації даних сенсорних мереж

Далі виконується виявлення пропусків та аномалій. Пропуски можуть виникати через втрату пакетів, збої мережі або тимчасову недоступність сенсорів. На цьому етапі будується часовий ряд для кожної пари сенсор, параметр та перевіряється наявність пропусків у послідовності міток часу. Також перевіряється, чи не мають значення некоректний характер. Для виявлення аномалій можуть використовуватися такі правила, як контроль діапазонів, максимальна швидкість або статистичні критерії, такі як відхилення від ковзного середнього більше певного порогу.

Після виявлення пропусків і аномальних значень виконується етап обробки пропусків. Стратегія залежить від подальших задач цифрового двійника та динаміки процесу. Якщо для конкретного параметра важлива лише груба тенденція, пропуски невеликої тривалості доцільно заповнювати методами типу last observation carried forward (перенесення останнього коректного значення), лінійною інтерполяцією між сусідніми вимірюваннями або згладжувальною інтерполяцією. Для параметрів, де точність критична (наприклад, контроль безпеки), серії пропусків можуть позначатися спеціальним статусом як «недостовірна ділянка», щоб надалі моделі контексту враховували понижену довіру до таких фрагментів. В окремих випадках надто фрагментовані ряди можуть повністю вилучатися з аналізу.

Наступний крок – фільтрація та згладжування шуму. Коректні вимірювання також містять високочастотні коливання через фізичні властивості сенсорів і зовнішні перешкоди. Для зменшення шуму застосовують ковзне середнє, медіанний фільтр, експоненційне згладжування або їх комбінування. Для сенсорних даних із ESP32, що вимірюють температуру й вологість у лабораторних умовах, зазвичай достатньо вікна в кілька секунд або хвилин залежно від періоду опитування.

Після очищення часового ряду виконується нормалізація та приведення одиниць вимірювання. Одні й ті самі фізичні величини можуть бути представлені в різних одиницях та в різних діапазонах. На цьому етапі до кожного параметра p застосовується функція перетворення

$$xp(t)=\phi p(dp(t)), \quad (2.10)$$

де, $dp(t)$ – очищене сире значення;

ϕ_p –функція калібрування.

Результатом є вектор нормалізованих параметрів

$$X(t)=(x_1(t),x_2(t),\ldots,x_m(t)), \quad (2.11)$$

який має узгоджені одиниці вимірювання, масштаби та структуру, придатну для безпосереднього використання у контекстній моделі.

Окремою задачею є часове узгодження та перерахунок часового ряду на інший крок дискретизації. Сенсорні вузли можуть мати різні періоди опитування, часові дрейфи або непостійну затримку доставки пакетів. Щоб сформувати єдиний вектор $X(t)$ значення різних параметрів необхідно привести до спільної сітки часу з кроком Δt. Для цього використовується агрегація, наприклад середнє, медіана, максимум за інтервал для кожного параметра в межах вікна $[t,t+\Delta t]$. Такий підхід дозволяє інтегрувати повільні та швидкі канали, синхронізувати показники з різних вузлів ESP32 та Raspberry Pi, а також забезпечити коректну роботу подальших аналітичних алгоритмів.

Ще одною складовою методу є кодування якості даних. Для кожного елемента $x_j(t)$х краще зберігати не лише числове значення, а і певний прапор якості (quality flag), який вказує, чи було значення інтерпольованим, чи підлягало фільтрації, чи виходило за допустимий діапазон. Це дає змогу на рівні контекстної моделі коригувати вагу різних фрагментів даних та уникати некоректних висновків у випадках, коли значна частина ряду сформована через відновлення або згладжування.

Після виконання всіх зазначених кроків модуль попередньої обробки формує послідовність нормалізованих векторів $X(t)$, які зберігаються в базі даних або буфері та передаються до функції інтерпретації контексту f_{ctx}. Отже

метод попередньої обробки та нормалізації виступає проміжною ланкою між необробленою сенсорною телеметрією та формальною контекстною моделлю цифрового двійника, забезпечуючи необхідну якість і сумісність даних.

2.3 Метод побудови та оновлення контекстної моделі цифрового двійника

В попередніх розділах було визначено формальну модель контексту цифрового двійника та описано метод попередньої обробки і нормалізації даних сенсорних мереж. Наступним кроком є формалізація методу побудови та оновлення контекстної моделі, тобто алгоритму, який перетворює нормалізовані вимірювання $X(t)$ на контекстні змінні $C(t)$, стани $\sigma(t)$ та події системи. Саме цей рівень забезпечує перехід від «низькорівневих» значень температури, вологості, освітлення до високорівневих інтерпретацій типу «норма», «перегрів», «ризик конденсації», «нестабільний режим».

Схематичний алгоритм побудови та оновлення контекстної моделі цифрового двійника наведено на рисунку 2.3.

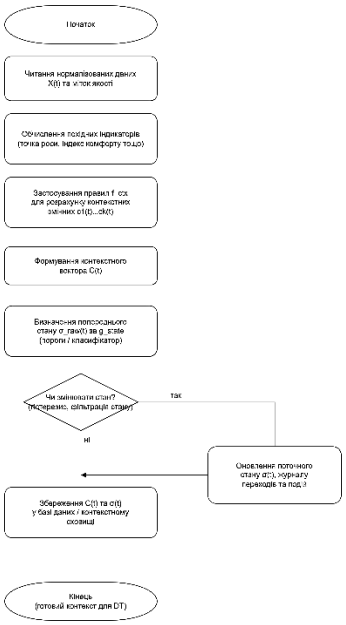


Рисунок 2.3 – Алгоритм побудови та оновлення контекстної моделі цифрового двійника

На вхід методу надходить нормалізований вектор параметрів (див 2.11), а також пов’язана з ним службова інформація: якість вимірювань, метадані про сенсори, конфігураційні пороги, інформація про зони моніторингу. Першим етапом є обчислення похідних індикаторів, які явно не вимірюються, але відіграють важливу роль в інтерпретації ситуації. Наприклад, за температурою та відносною вологістю можна обчислити точку роси, за температурою та вологістю – індекс теплового комфорту, за освітленістю – інтегральну освітленість за період. Ці похідні характеристики зручніші для формулювання правил, ніж окремі сирі параметри.

Другий етап – застосування функції інтерпретації контексту f_{ctx}, яка на основі $X(t)$ та множини похідних індикаторів формує значення контекстних змінних $ci(t)$. В найпростішому випадку така функція реалізується у вигляді набору правил. Наприклад, для контекстної змінної «ризик перегріву» можна задати правила:

– якщо температура >28∘C та індекс теплового комфорту «високий», то $c_{\text{перегрів}}(t)$=«високий»;

– якщо температура в межах 24–28 °C, а вологість помірна, то $c_{\text{перегрів}}(t)$=«середній»;

– якщо температура < 24 °C, то $c_{\text{перегрів}}(t)$=«низький».

Для контекстної змінної «ризик конденсації» можна використовувати співвідношення між поточною температурою та точкою роси: якщо різниця між температурою повітря та точкою роси менша за заданий поріг, то ризик вважається високим.

За потреби правила можуть бути не лише чіткими, а і нечіткими, коли кожній градації контекстної змінної відповідає функція приналежності, а рішення приймається на основі композиції нечітких правил. Це підвищує стійкість до шуму та зменшує чутливість до жорстких порогів.

Результатом застосування f_{ctx} є побудова вектора контексту

$$C(t)=(c1(t),c2(t),\ldots,ck(t)), \qquad (2.12)$$

у якому кожна змінна відображає певний аспект ситуації, такий як комфорт, ризики, якість даних, стабільність режиму, наявність підозрілих трендів. На цьому рівні цифровий двійник може оперувати не лише числами, а і ознаками категорій, що наближені до експертних описів.

Третій етап полягає у визначенні поточного стану цифрового двійника $\sigma(t)$ з множини можливих станів Σ. Для цього використовується функція

$$g_{\text{state}} : C(t) \rightarrow \Sigma, \tag{2.13}$$

яка реалізує алгоритм переходу від контекстних змінних до інтегральної оцінки стану. Можна виділити два основні підходи до реалізації g_{state} :

– Правило-орієнтований (rule-based) – стан визначається набором правил виду якщо($c_1(t)$=«низький ризик перегріву»∧$c_2(t)$=«низький ризик конденсації»)(c_1(t)=«низький ризик перегріву» ∧c_2(t)=«низький ризик конденсації»), то $\sigma(t)=\sigma_{\text{норма}}$ Такий підхід добре узгоджується з інженерними регламентами й легко інтерпретується.

– Навчений класифікатор – модель машинного навчання навчається на історичних даних, де кожному набору контекстних змінних відповідає «еталонний» стан, визначений експертом. Це дозволяє враховувати складні нелінійні залежності між контекстними змінними.

Незалежно від обраного підходу важливо уникати надмірної деталізації зміни станів при невеликих коливаннях параметрів. Тому в методі використовується механізм гістерезису та згладжування стану. Ідея полягає в тому, що новий стан $\sigma(t)$ змінює попередній $\sigma(t-1)$ за умови, що контекстні змінні стабільно відповідають іншій ситуації протягом певного інтервалу часу або перевищують пороги з запасом. Це запобігає частим перемиканням між режимами «норма» і «попередження» при незначних шумових відхиленнях.

Коли визначено новий стан, виконується оновлення журналу подій. Якщо $\sigma(t)\neq\sigma(t-1)$, створюється подія $e\in E$ з зазначенням пари станів

$(\sigma(t-1),\sigma(t))$, часу переходу, причини, які контекстні змінні стали тригерами та пов'язаних дій. Тому послідовність станів і подій утворює часову трасу поведінки ЦД.

Фінальним кроком алгоритму є збереження контекстного вектора та стану у базі даних або спеціалізованому контекстному сховищі. Для кожного моменту часу t фіксуються: вектор $X(t)$, контекст $C(t)$, поточний стан $\sigma(t)$, а також, за потреби, посилання на згенеровані події. Це дозволяє в подальшому виконувати аналіз історичних даних, виявляти закономірності, вдосконалювати правила та навчати нові моделі.

Метод побудови та оновлення контекстної моделі, є центральною ланкою між сенсорними даними і рівнем управління. Він забезпечує структуроване перетворення потоків вимірювань у знання про стан системи, що використовуються цифровим двійником для моніторингу, діагностики та підтримки прийняття рішень.

Висновки до розділу 2

1. Сформовано формальну контекстну модель цифрового двійника, яка поєднує в єдину систему множини сенсорних параметрів, часових міток, метаданих, контекстних змінних, станів і подій.

2. Показано, що контекстну модель можна реалізувати в різних структурних формах – як табличну модель у реляційній БД або сховищі часових рядів, як онтологію з класами та відношеннями між сенсорами, параметрами, станами та подіями, а також як граф станів із позначеними переходами. Це забезпечує гнучкість реалізації.

3. Розроблено метод попередньої обробки та нормалізації даних сенсорних мереж, який включає етапи валідації структури пакетів, виявлення та обробку пропусків і аномалій, фільтрацію шуму, перетворення одиниць вимірювання та масштабів. Запровадження прапорів якості (quality flags) для

кожного елемента дозволяє явно враховувати ступінь довіри до даних на рівні контекстної моделі та уникати помилкових висновків.

4. Запропоновано метод побудови та оновлення контекстної моделі, який перетворює нормалізовані вектори на контекстні змінні, стани та події, який , реалізованій у вигляді системи правил.

Такий підхід утворює завершений ланцюг інтеграції даних сенсорних мереж у цифровий двійник від сирих пакетів телеметрії, що надходять з вузлів ESP32, до високорівневих контекстних станів, придатних для моніторингу, діагностики та підтримки прийняття рішень, що забезпечує необхідну якість, узгодженість та семантичну насиченість даних.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ІНТЕГРАЦІЇ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

3.1 Архітектура програмного модуля інтеграції даних

Архітектура програмного модуля інтеграції даних (рисунок 3.1) побудована таким чином, щоб забезпечити безперервний потік інформації від сенсорних вузлів на базі ESP32 до контекстної моделі цифрового двійника, реалізованої на рівні програмних сервісів Raspberry Pi. Її можна розділити на три основні компоненти: підсистему прийому даних, ядро інтеграції та контекстної інтерпретації, а також інтерфейси взаємодії з цифровим двійником.

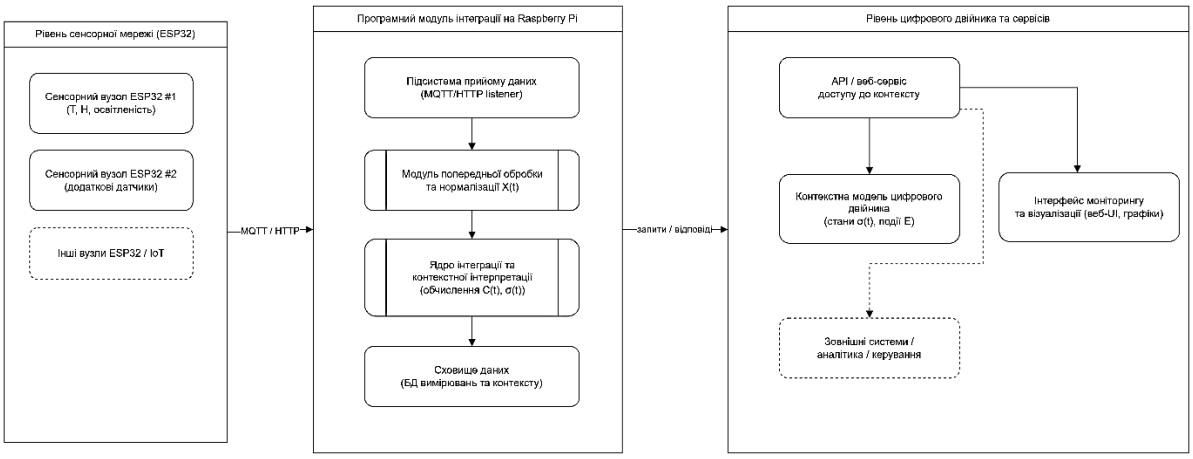


Рисунок 3.1 – Архітектура програмного модуля інтеграції даних цифрового двійника на базі Raspberry Pi та ESP32

На рівні сенсорної мережі розташовані один або кілька вузлів ESP32, до яких під’єднані датчики та інші перетворювачі. Кожен вузол періодично формує телеметричні повідомлення з виміряними значеннями та часовою міткою й передає їх у мережу за допомогою протоколів MQTT або HTTP. На цьому рівні відбувається лише первинний збір і відправлення даних, без складної логіки аналізу: ESP32 працює як «тонкий» клієнт, що шле дані на центральний вузол – Raspberry Pi.

Центральним елементом є програмний модуль інтеграції на Raspberry Pi, який реалізує основні функції прийому, обробки, нормалізації та інтерпретації даних. У його складі виділяється підсистема прийому даних, що включає MQTT/HTTP-слухачі (listener-сервіси), відповідальні за отримання повідомлень від усіх під'єднаних ESP32. Ці сервіси перевіряють базову структуру пакетів, виконують початкову валідацію та поміщають дані в буфер або чергу повідомлень для подальшої обробки.

Наступним компонентом є модуль попередньої обробки та нормалізації, який реалізує методи, описані вище. Він відповідає за очищення даних від шуму, виявлення та обробку пропусків, узгодження сенсорних потоків між собою до єдиної часової сітки, а також приведення вимірювань до канонічних одиниць і діапазонів. Результатом роботи цього модуля є вектори нормалізованих параметрів $X(t)$, які вже готові для використання на рівні контекстної моделі.

Ключовим елементом архітектури виступає ядро інтеграції та контекстної інтерпретації. В ньому реалізовано алгоритми побудови контекстних змінних $C(t)$ і визначення станів $\sigma(t)$. Ядро отримує нормалізовані дані, обчислює похідні індикатори, наприклад точку роси, , застосовує набір правил або навчений класифікатор для визначення контекстних категорій, а також оновлює поточний стан цифрового двійника з врахуванням журналу подій.

Для довготривалого збереження та подальшого аналізу використовується сховище даних, яке може бути реалізоване у вигляді реляційної БД чи time-series бази. У ньому фіксуються як сирі часи та вектори $X(t)$, так і відповідні контекстні вектори $C(t)$, стани $\sigma(t)$ та події E. Це дозволяє відтворювати історію роботи системи, навчати або донавчати моделі, а також проводити експериментальні дослідження.

Третя група компонентів – це інтерфейси взаємодії з цифровим двійником, які забезпечують доступ до контекстних даних, станів та подій з боку зовнішніх сервісів. До них належить REST/HTTP API та gRPC-інтерфейс,

через який модуль інтеграції надає дані прикладним системам, а також веб-сервіс візуалізації. Через ці інтерфейси реалізується інтеграція з високорівневою контекстною моделлю цифрового двійника, панелями моніторингу в реальному часі, аналітичними модулями та системами автоматичного керування.

В загальному, архітектура програмного модуля інтеграції даних будується за принципом розділених рівнів: сенсорний (ESP32), інтеграційний і рівень цифрового двійника. Така структуризація спрощує розробку, тестування та подальше розширення системи.

3.2 Реалізація програмного модуля інтеграції

Базовим програмним оточенням для реалізації модуля інтеграції є одноплатний комп’ютер Raspberry Pi, на якому встановлено операційну систему сімейства Linux, інтерпретатор мови програмування Python та необхідні серверні компоненти (MQTT-брокер, веб-сервіс тощо). Логіка роботи модуля організована у вигляді набору сервісів і фонових процесів, кожен з яких відповідає за окрему підфункцію: прийом даних, попередню обробку, оновлення контекстної моделі, забезпечення доступу до даних через прикладний інтерфейс програмування.

На Raspberry Pi виділяються три логічні сервіси:

– сервіс прийому телеметрії від ESP32 (listener-сервіс);

– сервіс обробки, нормалізації та збереження даних;

– сервіс контекстної інтерпретації та оновлення станів цифрового двійника;

– сервіс REST-API для доступу до вимірювань, контекстних змінних та станів.

В найпростішій конфігурації ці сервіси можуть бути реалізовані у вигляді окремих модулів в одному застосунку, які запускаються як незалежні потоки або асинхронні задачі. В більш комплексному варіанті вони можуть

бути оформлені як окремі системні служби, що спрощує розгортання та відмовостійкість.

Сервіс прийому даних реалізує підписку на теми MQTT-брокера, які надсилають вузли ESP32. Кожен пакет телеметрії містить ідентифікатор сенсора, мітку часу та набір параметрів. В програмному коді це реалізується як нескінченний цикл, в якому викликається callback-функція при надходженні нового повідомлення.

Приклад коду, який показує лише логіку прийому та збереження одного повідомлення представлена в коді на рисунку 3.2.

```
def on_message(client, userdata, msg):
    payload = json.loads(msg.payload.decode("utf-8"))
    # мінімальна валідація
    if "node_id" not in payload or "ts" not in payload:
        log_error("invalid_format", msg.payload)
        return

    # збереження «сирих» значень у БД
    cur.execute(
        "INSERT INTO raw_measurements(node_id, ts, temperature, humidity, light) "
        "VALUES (?, ?, ?, ?, ?)",
        (
            payload["node_id"],
            payload["ts"],
            payload.get("temperature"),
            payload.get("humidity"),
            payload.get("light"),
        ),
    )
    conn.commit()
```

Рисунок 3.2 – Прийом та збереження одного повідомлення

Всередині callback проводиться базова перевірка формату, тобто чи є наявність обов’язкових полів, коректність типів даних, після чого повідомлення поміщається у внутрішню чергу або тимчасову таблицю бази даних для подальшої обробки. В разі виявлення помилки пакет реєструється в журналі, що дозволяє надалі аналізувати якість роботи сенсорної мережі.

Наступний компонент – модуль, який реалізує метод попередньої обробки, який описаний вище. На програмному рівні він періодично вибирає з черги або з таблиці сирих даних всі нові записи, виконує виявлення пропусків, некоректних значень та шуму, застосовує обрану стратегію обробки пропусків, здійснює зведення до спільної часової сітки та перетворення одиниць вимірювання.

Типова структура таблиці нормалізованих даних може включати поля: ідентифікатор запису, час t, ідентифікатор сенсорної зони або об’єкта, набір полів для параметрів, temperature_c, humidity_rel, light_lux, а також додаткові службові поля (quality_flag, source_node). Така структура полегшує подальшу аналітику та побудову графіків. Приклад ядра нормалізації одного запису представлено на рисунку 3.3.

```python
def normalize_measurement(raw):
    t_raw = raw["temperature"]
    h_raw = raw["humidity"]
    l_raw = raw["light"]

    # контроль фізично допустимого діапазону температури
    if t_raw is None or t_raw < -40 or t_raw > 85:
        temp_c = None
    else:
        temp_c = t_raw

    # обрізання відносної вологості до [0;100]
    hum_rel = None if h_raw is None else max(0, min(100, h_raw))

    # освітленість не може бути від’ємною
    light_lux = None if l_raw is None else max(0.0, l_raw)

    return {
        "node_id": raw["node_id"],
        "ts":      raw["ts"],
        "temperature_c": temp_c,
        "humidity_rel":  hum_rel,
        "light_lux":     light_lux,
    }
```

Рисунок 3.3 – Ядро нормалізації одного запису:

Третій компонент – сервіс, який періодично зчитує з таблиці нормалізованих даних найсвіжіші вектори $X(t)$, обчислює похідні індикатори, застосовує правило-орієнтовану функцію f_{ctx} або навчений класифікатор для отримання контекстних змінних $C(t)$, а далі за функцією g_{state} визначає поточний стан $\sigma(t)$ цифрового двійника. В реалізації може бути використано механізм гістерезису, який перевіряє, чи достатньо стабільно новий стан відрізняється від попереднього, перш ніж записати його в базу.

Результати роботи цього модуля зберігаються в окремих таблицях контексту і стану (рисунок 3.4). Таблиця контексту містить часові позначки та значення контекстних змінних, а таблиця станів – послідовність станів

```python
def compute_context(temp_c, hum_rel):
    # приклад контекстної змінної "перегрів"
    if temp_c is None:
        overheat = "unknown"
    elif temp_c > 28:
        overheat = "high"
    elif temp_c >= 24:
        overheat = "medium"
    else:
        overheat = "low"

    # приклад контекстної змінної "ризик конденсації"
    if temp_c is None or hum_rel is None:
        cond_risk = "unknown"
    elif hum_rel > 80 and temp_c < 22:
        cond_risk = "high"
    elif hum_rel > 60:
        cond_risk = "medium"
    else:
        cond_risk = "low"

    return {"overheat": overheat, "condensation_risk": cond_risk}


def compute_state(ctx):
    if ctx["overheat"] == "high":
        return "ALARM_OVERHEAT"
    if ctx["condensation_risk"] == "high":
        return "ALARM_CONDENSATION"
    if ctx["overheat"] == "medium" or ctx["condensation_risk"] == "medium":
        return "WARNING"
    return "NORMAL"

```

Рисунок 3.4 – Мінімальний набір правил для контексту та стану

Структура бази відповідає логіці модулів: таблиця сирих даних, таблиця нормалізованих вимірювань, таблиця контексту, таблиця станів та таблиця журналу помилок. Для взаємодії з цифровим двійником та інтерфейсом моніторингу реалізується REST-API, у якому передбачено методи для отримання останнього значення контексту, історії станів за заданий інтервал часу, а також агрегованих статистичних показників.

Таким чином, реалізація програмного модуля інтеграції на Raspberry Pi забезпечує замкнений цикл обробки від прийому даних від ESP32 через очищення та нормалізацію до побудови контекстної моделі і збереження станів, доступ до яких надається зовнішнім компонентам цифрового двійника через стандартизований програмний інтерфейс.

3.3 Інтерфейс взаємодії з контекстною моделлю цифрового двійника

Інтерфейс взаємодії з контекстною моделлю цифрового двійника виконує роль шлюзу в стан кіберфізичної системи. Його завданням є наочне відображення поточних контекстних змінних, станів цифрового двійника, історії переходів між станами, а також забезпечення можливості моніторингу у режимі, близькому до реального часу. У розробленому рішенні інтерфейс

реалізовано у вигляді веб-додатку, що працює поверх REST-API, описаного в підрозділі 3.2

Веб-інтерфейс (рисунок 3.5) умовно складається з кількох основних екранів: головної панелі моніторингу (dashboard), екрану історії станів та деталізованого перегляду контексту окремого вузла [30]. На головній панелі відображається поточний стан цифрового двійника $\sigma(t)$ у вигляді великого індикатора типу «світлофор»: зелений колір – NORMAL, жовтий – WARNING, червоний – ALARM_. Поруч розміщуються числові віджети з останніми значеннями температури, вологості, освітленості та контекстних змінних.

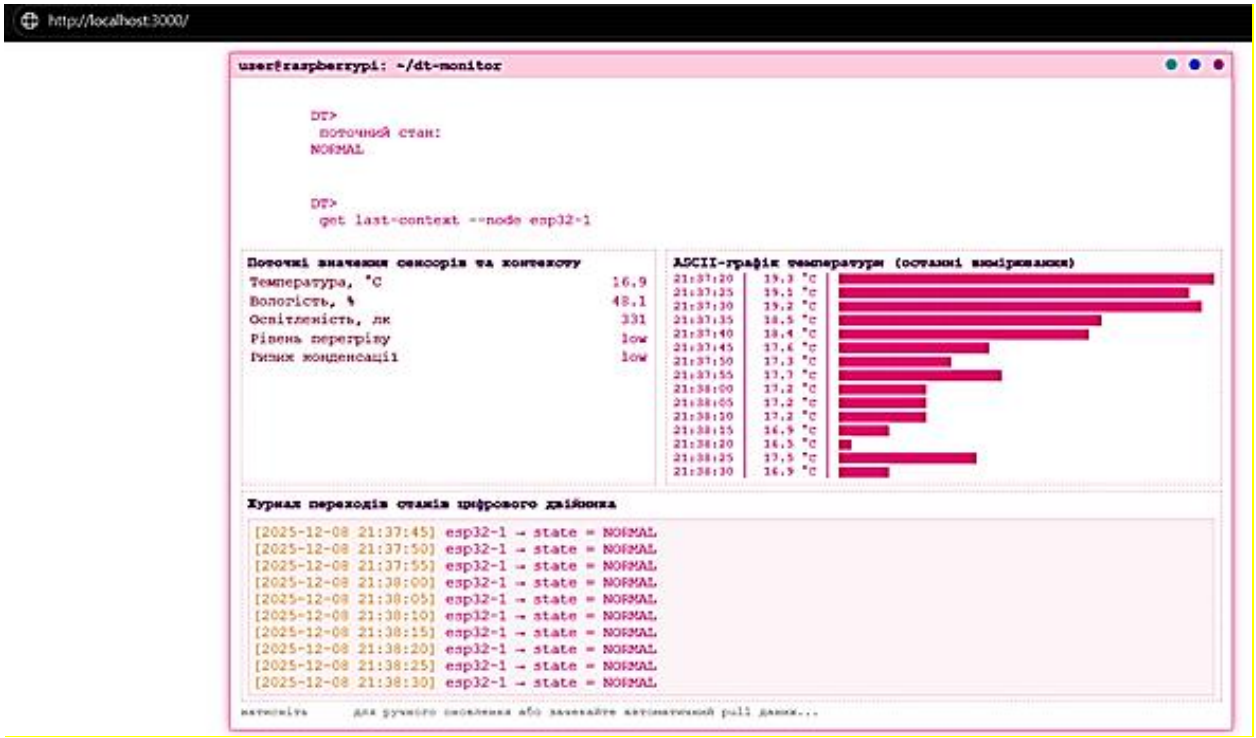


Рисунок 3.5 – Веб-інтерфейс моніторингу

Для моніторингу в режимі, близькому до реального часу, застосовується періодичне опитування REST-API із невеликим інтервалом, наприклад, 5–10 секунд . Це дозволяє оновлювати індикатори та графіки без перезавантаження сторінки (Рисунок 3.6).

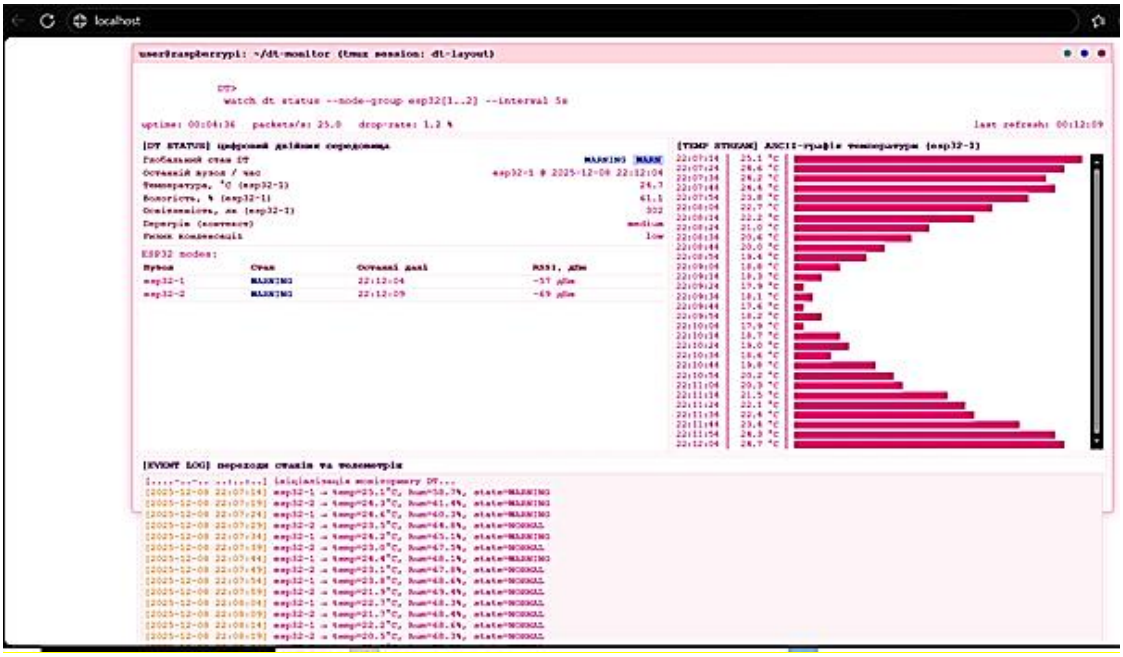


Рисунок 3.6 – Моніторинг в режимі, близькому до реального часу

Другий важливий елемент інтерфейсу – графіки часових рядів (рисунок 3.7). Для візуалізації даних доцільно відображати принаймні два види графіків:

– графік зміни температури, вологості та освітленості у часі на основі таблиці normalized_measurements;

– графік зміни станів цифрового двійника $\sigma(t)$.

Рисунок 3.7 – Графіки часових рядів

На екрані історії станів зручно подати таблицю переходів між станами з часовими позначками.

3.4 Аналіз результатів експериментальних досліджень

В даному розділі представлено результати експериментального дослідження роботи програмного модуля інтеграції даних сенсорних мереж та його впливу на формування контекстної моделі цифрового двійника. Також представлено порівняння запропонованого підходу з базовим варіантом, у якому вимірювання з ESP32 безпосередньо заносяться до бази даних цифрового двійника без застосування фільтрації, нормалізації та контекстної інтерпретації.

Експерименти проводилися на основі запису даних з двох вузлів ESP32, підключених до Raspberry Pi, з періодом опитування 5 секунд (рисунок 3.8). Упродовж кількох годин реєструвалися значення температури, відносної вологості та освітленості у лабораторному приміщенні. Під час вимірювань навмисно створювалися зміни умов, що дозволило змоделювати типові робочі ситуації. Усі сирі вимірювання зберігалися в таблиці raw_measurements, а результати після застосування алгоритмів попередньої обробки – у таблиці normalized_measurements.

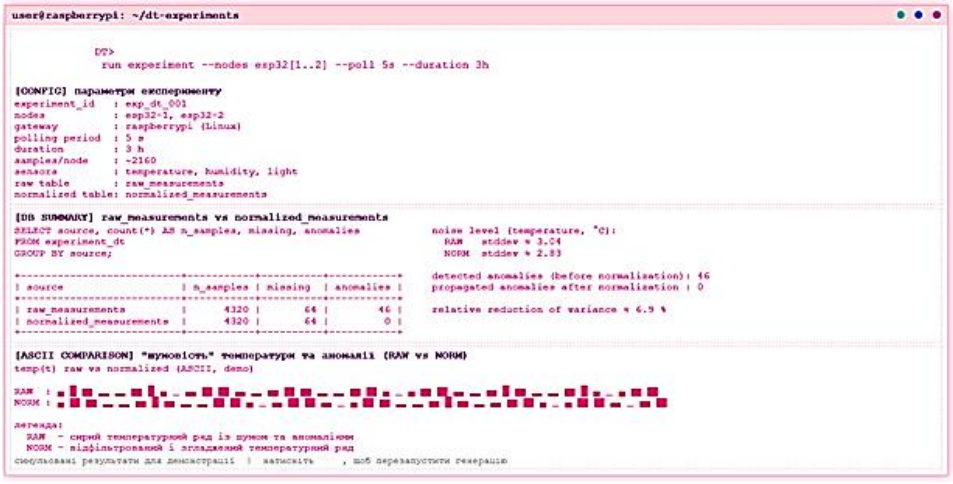


Рисунок 3.8 – Термінальний звіт про результати експерименту

Порівняння дає змогу відзначити, що при простому підході спостерігається підвищена варіативність вимірювань і більша кількість стрибків температури та вологості, тоді як використання запропонованого модуля призводить до зменшення дисперсії та усунення короткочасних сплесків, які можуть бути пов’язані з похибками сенсорів чи мережевими затримками.

Візуальне зіставлення часових рядів температури, вологості та освітленості для сирих і нормалізованих даних наведено на рисунку 3.9.

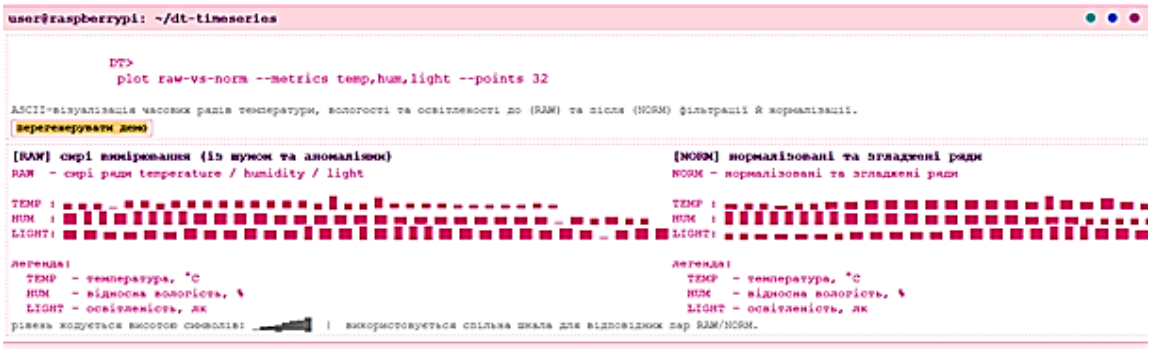


Рисунок 3.9 – Зіставлення часових рядів

Графік відображає зміну трьох параметрів в часі для сирого сигналу характерні траєкторії з поодинокими різкими змінами, тоді як нормалізовані ряди виглядають більш згладженими і краще відображають реальну динаміку мікроклімату.

Другий блок експериментів був спрямований на оцінювання якості контекстної інтерпретації. Для кожного моменту часу формувалися стани цифрового двійника $\sigma(t)$ за двома варіантами: на основі прямого порівняння температури та вологості з фіксованими порогами і запропонованим, що спирається на контекстну модель з системою правил, гістерезисом та додатковими змінними.

Для ілюстрації роботи модуля станів реалізовано окремий термінальний інтерфейс, який у компактній формі показує часову діаграму $\sigma(t)$ у вигляді послідовності символів NORMAL / WARNING / ALARM. Приклад такого

інтерфейсу наведено на рисунку 3.10. У верхньому рядку представлено часову шкалу, у нижньому – відповідну послідовність станів: нормальний режим позначається символом N зеленого кольору, попереджувальний – W жовтого, аварійний – A червоного. Подібне подання допомагає оператору швидко оцінити структуру подій, тривалість інтервалів підвищеного ризику та зіставити їх із динамікою сенсорних вимірювань.

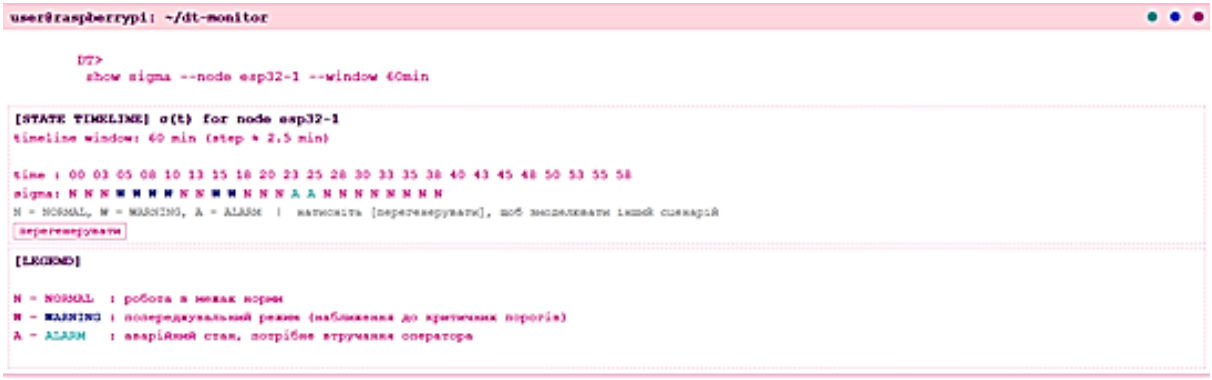


Рисунок 3.10 – Термінальний інтерфейс відображення часової діаграми станів σ(t) цифрового двійника (NORMAL/WARNING/ALARM) для вузла ESP32-1

Окрему увагу приділено інтегрованому tmux- дашборду, який поєднує в одному вікні агреговані метрики часу роботи системи, packets/s, drop-rate, таблицю станів кількох ESP-вузлів і ASCII-графік температури. Інтерфейс, представлений на рисунку 3.11.

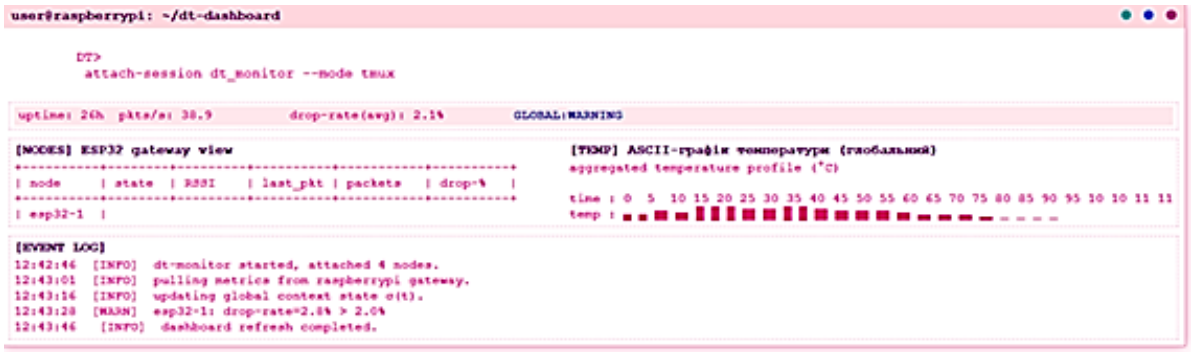


Рисунок 3.11 – «tmux- термінальний дашборд цифрового двійника

Він дає змогу одночасно спостерігати локальні параметри та узагальнений стан цифрового двійника. Запропонований модуль інтеграції може масштабуватися на декілька сенсорних вузлів без втрати читабельності інформації та керованості системи.

До переваг підходу належать можливість функціонування в режимі, наближеному до реального часу, прозора структура правил формування контекстних станів і зручна інтеграція з термінальними веб-інтерфейсами на Raspberry Pi. Але якість результатів суттєво залежить від коректного налаштування порогів і параметрів фільтрації, а також потребує додаткового підтвердження в умовах реальних виробничих процесів з більш складною структурою сенсорної мережі та динаміки середовища.

Висновки до розділу 3

1. Розроблено та реалізовано архітектуру програмного модуля інтеграції даних, яка забезпечує безперервний потік інформації від сенсорних вузлів на базі ESP32 до контекстної моделі цифрового двійника на Raspberry Pi.

2. Модуль інтеграції на Raspberry Pi організований як набір спеціалізованих сервісів: listener-сервіс для прийому MQTT/HTTP-повідомлень, сервіс нормалізації та збереження даних, сервіс контекстної інтерпретації і оновлення станів, а також REST-API для зовнішнього доступу.

3. Розроблено веб-інтерфейс і термінальні дашборди, які забезпечують взаємодію з контекстною моделлю цифрового двійника.

4. Проведено експериментальні дослідження з використанням двох вузлів ESP32 показали, що застосування алгоритмів попередньої обробки зменшує варіативність вимірювань, усуває короткочасні сплески і помилки, пов’язані з сенсорами та мережевими затримками.

5. Оцінка якості контекстної інтерпретації показала, що використання системи правил та похідних індикаторів дозволило зменшити кількість хибних

спрацьовувань режимів WARNING/ALARM порівняно з простим пороговим підходом.

6. Програмна реалізація підтвердила працездатність запропонованого методу інтеграції даних сенсорних мереж у цифровий двійник і його придатність до роботи в режимі, наближеному до реального часу.

ВИСНОВКИ

1. Обґрунтовано доцільність реалізації інтеграційного модуля на рівні edge-вузла Raspberry Pi, який приймає телеметрію від вузлів ESP32, виконує локальну обробку та надає цифровому двійнику вже структуровані контекстні дані.

2. Сформовано формальну контекстну модель цифрового двійника, в якій параметри сенсорних вимірювань, контекстні змінні, стани та події описано в єдиному просторі.

3. Розроблено й реалізовано метод попередньої обробки та нормалізації потоків сенсорних даних, що включає фільтрацію шуму, виявлення та компенсацію пропусків, узгодження часових масштабів, видалення аномалій та перетворення вимірювань з ознаками якості.

4. Запропоновано метод побудови та оновлення контекстної моделі цифрового двійника, який базується на системі правил.

5. Створено архітектуру програмного модуля інтеграції даних, в якій виокремлено сервіси прийому (listener для MQTT/HTTP), нормалізації та збереження вимірювань, контекстної інтерпретації й оновлення станів. Така компонентна побудова спрощує модифікацію і масштабування рішення, дозволяє гнучко змінювати алгоритми обробки даних і легко інтегрувати модуль у різні варіанти цифрових двійників.

6. Реалізовано набір інтерфейсів користувача у вигляді веб-панелі моніторингу, що відображають поточні значення параметрів мікроклімату, контекстні змінні, часові графіки вимірювань і діаграми станів $\sigma(t)$.

7. Проведені експериментальні дослідження підтвердили працездатність запропонованого методу інтеграції.

8. Практична цінність отриманих результатів полягає в тому, що модуль може слугувати універсальною основою для створення цифрових двійників у різних сферах: від навчальних стендів і наукових установ до комерційних IoT-рішень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Grieves M., Vickers J. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems // Transdisciplinary Perspectives on Complex Systems / eds. F.-J. Kahlen, S. Flumerfelt, A. Alves. Cham : Springer, 2017. P. 85–113.

2. Hu W., Zhang T., Deng X., Liu Z., Tan J. Digital twin: A state-of-the-art review of its enabling technologies, applications and challenges // Journal of Intelligent Manufacturing and Special Equipment. 2021. Vol. 2, No. 1. P. 1–34.

3. Kritzinger W., Karner M., Traar G., Henjes J., Sihn W. Digital Twin in Manufacturing: A Categorical Literature Review and Classification // IFAC-PapersOnLine. 2018. Vol. 51, No. 11. P. 1016–1022.

4. Gubbi J., Buyya R., Marusic S., Palaniswami M. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions // Future Generation Computer Systems. 2013. Vol. 29, No. 7. P. 1645–1660.

5. Yick J., Mukherjee B., Ghosal D. Wireless Sensor Network Survey // Computer Networks. 2008. Vol. 52, No. 12. P. 2292–2330.

6. Lee J., Bagheri B., Kao H.-A. A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems // Manufacturing Letters. 2015. Vol. 3. P. 18–23.

7. Perera C., Zaslavsky A., Christen P., Georgakopoulos D. Context Aware Computing for the Internet of Things: A Survey // IEEE Communications Surveys & Tutorials. 2014. Vol. 16, No. 1. P. 414–454.

8. Dey A. K. Understanding and Using Context // Personal and Ubiquitous Computing. 2001. Vol. 5, No. 1. P. 4–7.

9. MQTT Version 3.1.1. OASIS Standard. 2014. 81 p.

10. Shelby Z., Hartke K., Bormann C. The Constrained Application Protocol (CoAP). RFC 7252. IETF, 2014. 112 p.

11. Blázquez-García A., Conde A., Mori U., Lozano J. A. A Review on Outlier/Anomaly Detection in Time Series Data // ACM Computing Surveys. 2021. Vol. 54, No. 3. Article 56.

12. Hundman K., Constantinou V., Laporte C., Colwell I., Soderstrom T. Detecting Spacecraft Anomalies Using LSTMs and Nonparametric Dynamic Thresholding // Proc. 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18). 2018. P. 387–395.

13. Жураковський Б. Ю., Зенів І. О. Технології інтернету речей : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2019. 180 с.

14. Петренко С., Дегтярьова Н. Мова програмування Python: основи програмування : навч. посіб. Суми : СумДПУ імені А. С. Макаренка, 2023. 101 с.

15. Shchehlov V., Morozova O. Методи та технології розроблення цифрових двійників для гарантоздатних систем індустріального інтернету речей // Системи управління, навігації та зв'язку : зб. наук. пр. 2022. Т. 4, № 70. С. 127–137.

16. Tao F., Cheng J., Qi Q., Zhang M., Zhang H., Sui F. Digital twin-driven product design, manufacturing and service // Computers in Industry. 2018. Vol. 100. P. 128–143.

17. Parnianifard A., Jearavongtakul S., Sasithong P., Sinpan N., Poomrittigul S., Bajpai A., Wuttisittikulkij L. Digital-twins towards cyber-physical systems: a brief survey // Engineering Journal. 2022. Vol. 26, No. 9. P. 47–61.

18. Fuller A., Fan Z., Day C., Barlow C. Digital Twin: Enabling Technologies, Challenges and Open Research // IEEE Access. 2020. Vol. 8. P. 108952–108971.

19. Qi Q., Tao F. Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison // IEEE Access. 2018. Vol. 6. P. 3585–3593.

20. Jane V. A., Arockiam L. Survey on IoT data preprocessing // Turkish Journal of Computer and Mathematics Education. 2021. Vol. 12, No. 9. P. 238–244.

21. Zanella A., Bui N., Castellani A., Vangelista L., Zorzi M. Internet of Things for Smart Cities // IEEE Internet of Things Journal. 2014. Vol. 1, No. 1. P. 22–32.

22. Sodhro A. H., Pirbhulal S., de Albuquerque V. H. C. Towards an Optimal Resource Management for IoT Based Green and Sustainable Smart Cities // Journal of Cleaner Production. 2019. Vol. 220. P. 1167–1179.

23. Alshathri S., Hemdan E. E. D., El-Shafai W., Sayed A. Digital twin-based automated fault diagnosis in industrial IoT applications // Computers, Materials & Continua. 2023. Vol. 75, No. 1. P. 183–196.

24. Kramp T., Van Kranenburg R., Lange S. Introduction to the Internet of Things // Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model. Berlin ; Heidelberg : Springer Berlin Heidelberg, 2013. P. 1–10.

25. Ray P. P., Dash D., De D. Edge computing for Internet of Things: A survey, e-healthcare case study and future direction // Journal of Network and Computer Applications. 2019. Vol. 140. P. 1–22.

26. Islam M., Dukyil A. S., Alyahya S., Habib S. An IoT enable anomaly detection system for smart city surveillance // Sensors. 2023. Vol. 23, No. 4. Article 2358.

27. Bandyopadhyay D., Sen J. Internet of Things: Applications and Challenges in Technology and Standardization // Wireless Personal Communications. 2011. Vol. 58, No. 1. P. 49–69.

28. Крайник О. В. Вимоги до бездротових сенсорних мереж // Інформаційні моделі, системи та технології : матеріали XII наук.-техн. конф. 2024. С. 136–136.

29. Архітектура та технології Інтернету речей : навч. посіб. / І. В. Пулеко, А. А. Єфіменко. Житомир : Державний університет «Житомирська політехніка», 2022. 234 с.

30. Шинкарчук М. Б., Осолінський О. Р., Програмний модуль інтеграції даних сенсорних мереж для цифрового двійника на базі ESP32 та Raspberry Pi,

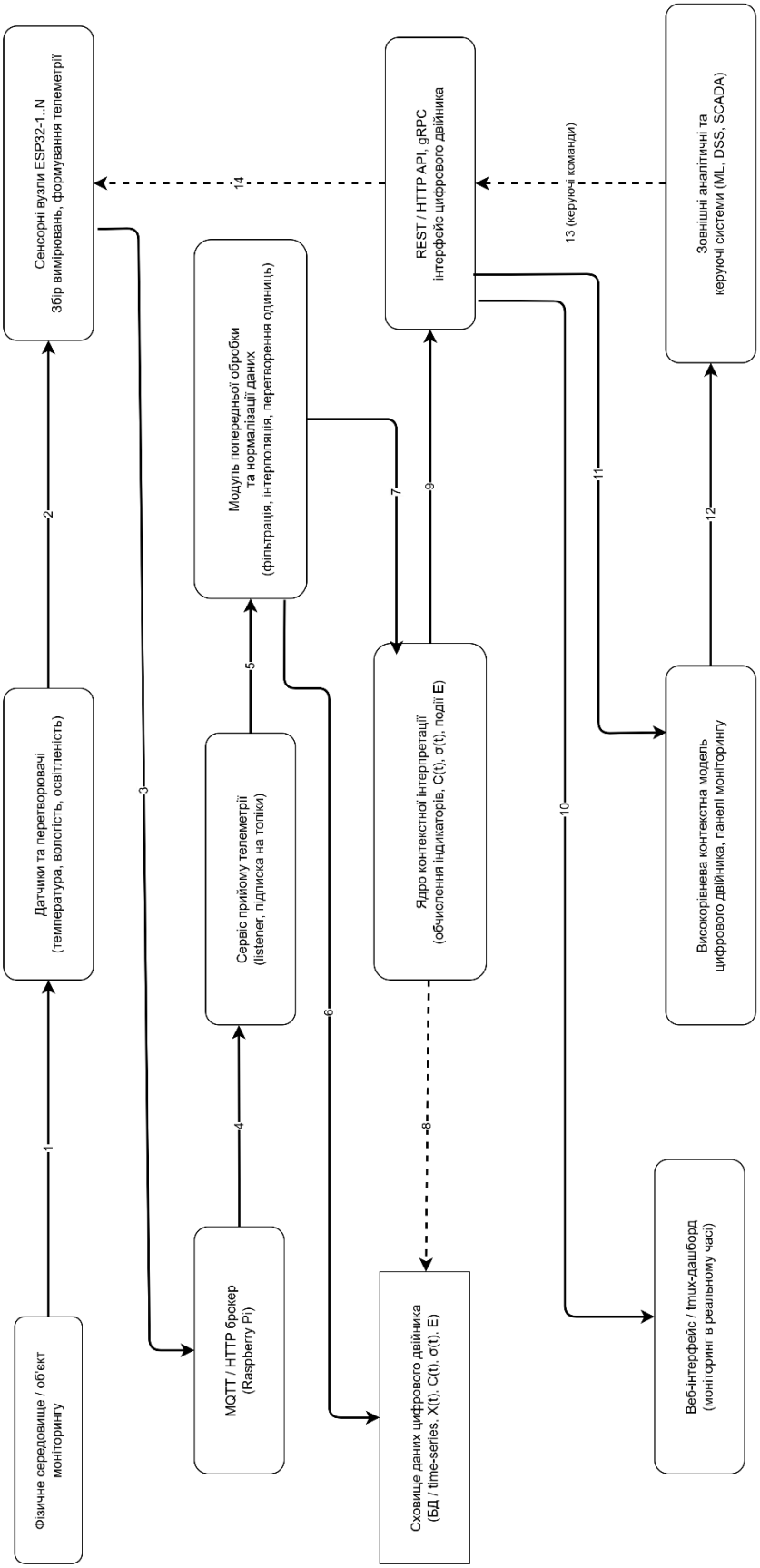
VI Міжнародна наукова конференція «Теорія модернізації в контексті сучасної світової науки», 2025 р., м. Івано-Франківськ, Україна. (прийнято до друку)

31. Шинкарчук М. Б., Осолінський О. Р., Метод інтеграції даних сенсорних мереж у контекстну модель цифрового двійника, IX Всеукраїнська студентська конференція «Науковий простір: аналіз, сучасний стан, тренди та перспективи», 2025 р., м. Вінниця, Україна. (прийнято до друку)

32. Комар М.П., Саченко А.О., Васильків Н.М., Загородня Д.І. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки» за другим (магістерським) рівнем вищої освіти. – Тернопіль: ЗУНУ, 2024. – 32 с.

Додаток А

Функціональна схема програмного модуля інтеграції даних сенсорних мереж у цифровий двійник



- Потік 1 – вимірювання фізичних параметрів.
- Потік 2 – формування телеметрії на ESP32.
- Потік 3 – передача телеметрії у мережу.
- Потік 4 – прийом даних listener-сервісами.
- Потік 5 – попередня обробка та нормалізація.
- Потік 6 – збереження нормалізованих даних у сховище.
- Потік 7 – контекстна інтерпретація.
- Потік 8 – запис контекстних даних та станів.
- Потік 9 – доступ до даних через REST/HTTP API та gRPC.
- Потік 10 – візуалізація в режимі, наближеному до реального часу.
- Потік 11 – інтеграція з високорівневою контекстною моделлю цифрового двійника.
- Потік 12 – формування керуючих впливів (зворотний канал).

Додаток Б

Код вузла ESP32

```
/*
 * Вузол сенсорної мережі ESP32 для цифрового двійника
 * Збирає температуру, вологість та освітленість і надсилає їх на Raspberry Pi через
MQTT.
  */

#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// ---------------------- Налаштування мережі ----------------------

const char* WIFI_SSID     = "YOUR_WIFI_SSID";
const char* WIFI_PASSWORD = "YOUR_WIFI_PASSWORD";
const char* MQTT_HOST = "192.168.0.10";
const uint16_t MQTT_PORT = 1883;
const char* MQTT_TOPIC_RAW = "dt/esp32-1/raw";
const char* NODE_ID = "esp32-1";
const unsigned long PUBLISH_INTERVAL_MS = 5000;
// ---------------------- Налаштування датчиків ----------------------
// Датчик температури та вологості (DHT22 на цифровому піні)
#define DHTPIN 4
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);
// Аналоговий датчик освітленості (фоторезистор тощо) на ADC-піні
const int LIGHT_PIN = 34;
// ---------------------- Глобальні змінні ----------------------
WiFiClient espClient;
PubSubClient mqttClient(espClient);
unsigned long lastPublishMs = 0;
// ---------------------- Функції ----------------------
void connectToWiFi() {
  Serial.print("Підключення до Wi-Fi: ");
  Serial.println(WIFI_SSID);
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  uint8_t attempts = 0;
  while (WiFi.status() != WL_CONNECTED && attempts < 30) {
    delay(500);
    Serial.print(".");
    attempts++;
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println();
    Serial.print("Wi-Fi підключено, IP: ");
    Serial.println(WiFi.localIP());
  } else {
    Serial.println();
    Serial.println("Не вдалося підключитися до Wi-Fi");
  }
}
void connectToMQTT() {
  // Поки не підключені до MQTT – пробуємо
  while (!mqttClient.connected()) {
    Serial.print("Підключення до MQTT-брокера ");
    Serial.print(MQTT_HOST);
    Serial.print(":");
    Serial.println(MQTT_PORT);
    String clientId = "esp32-node-";
    clientId += String((uint32_t)ESP.getEfuseMac(), HEX);
    if (mqttClient.connect(clientId.c_str())) {
      Serial.println("MQTT підключено");
    } else {
      Serial.print("Помилка MQTT, код = ");
      Serial.print(mqttClient.state());
      Serial.println(" – повторна спроба через 3 с");
```

```
      delay(3000);
    }
  }
}
void setupWiFiAndMQTT() {
  connectToWiFi();
  mqttClient.setServer(MQTT_HOST, MQTT_PORT);
  connectToMQTT();
}
bool readSensors(float& tC, float& hRel, int& lightRaw) {
  // Зчитування DHT
  float h = dht.readHumidity();
  float t = dht.readTemperature(); // за замовчуванням °C
  if (isnan(h) || isnan(t)) {
    Serial.println("Помилка зчитування з DHT22");
    return false;
  }
  int light = analogRead(LIGHT_PIN);
  tC = t;
  hRel = h;
  lightRaw = light;
  return true;
}
void publishTelemetry() {
  float tC = NAN;
  float hRel = NAN;
  int lightRaw = 0;
  if (!readSensors(tC, hRel, lightRaw)) {
    // Якщо помилка датчика –пропустити пакет, або надіслати спеціальні значення
    Serial.println("Телеметрія не відправлена через помилку датчика");
    return;
  }
  unsigned long tsMs = millis();
  float lightNorm = (float)lightRaw / 4095.0f;
  char payload[256];
  snprintf(
    payload,
    sizeof(payload),
    "{"
      "\"node_id\":\"%s\","
      "\"ts_ms\":%lu,"
      "\"t_c\":%.2f,"
      "\"h_rel\":%.2f,"
      "\"light_raw\":%d,"
      "\"light_norm\":%.3f"
    "}",
    NODE_ID,
    tsMs,
    tC,
    hRel,
    lightRaw,
    lightNorm
  );
  Serial.print("Публікація в MQTT: ");
  Serial.println(payload);
  bool ok = mqttClient.publish(MQTT_TOPIC_RAW, payload);
  if (!ok) {
    Serial.println("Помилка публікації MQTT");
  }
}
// ---------------------- Стандартні функції Arduino ----------------------
void setup() {
  Serial.begin(115200);
  delay(2000);
  Serial.println();
  Serial.println("Старт вузла ESP32 для цифрового двійника");
  dht.begin();
  analogReadResolution(12); // 0..4095
  setupWiFiAndMQTT();
  lastPublishMs = millis();
}
```

```
void loop() {
  if (!mqttClient.connected()) {
    connectToMQTT();
  }
  mqttClient.loop();
  unsigned long now = millis();
  if (now - lastPublishMs >= PUBLISH_INTERVAL_MS) {
    lastPublishMs = now;
    publishTelemetry();
  }
  delay(10);
}
```

Додаток В
Код для Raspberry Pi – edge-модуля інтеграції

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Модуль інтеграції даних сенсорних мереж на Raspberry Pi
-------------------------------------------------------
Функції:
- Підключення до MQTT-брокера (Mosquitto на Raspberry Pi).
- Підписка на топіки dt/+/raw (усі вузли ESP32).
- Попередня обробка: валідація, виявлення аномалій, згладжування, нормалізація.
- Формування контекстних змінних та інтегрального стану (NORMAL/WARNING/ALARM).
- Запис сирих, нормалізованих та контекстних даних у SQLite.
Draw(s,p,t) → X(t) → C(t) → σ(t), E.
"""

import json
import time
import signal
import sys
import sqlite3
from collections import defaultdict

import paho.mqtt.client as mqtt

# ---------------------- НАЛАШТУВАННЯ ----------------------

# MQTT на Raspberry Pi
MQTT_HOST = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC = "dt/+/raw"   # dt/esp32-1/raw, dt/esp32-2/raw, ...
# База даних на Pi
DB_PATH = "dt_edge.db"
# Параметри згладжування (експоненційне ковзне середнє)
EMA_ALPHA = 0.3
# ---------------------- ГЛОБАЛЬНІ СТРУКТУРИ ----------------------
# Згладжені значення по вузлах: node_id -> { "t_c": ..., "h_rel": ..., "light_norm": ...
}
ema_state = defaultdict(dict)
# Попередній стан для формування подій: node_id -> "NORMAL"/"WARNING"/"ALARM"
prev_state = {}
# Підключення до БД
conn = None
# MQTT-клієнт
mqtt_client = None


# ---------------------- БАЗА ДАНИХ ----------------------

def init_db():
    global conn
    conn = sqlite3.connect(DB_PATH, check_same_thread=False)
    cur = conn.cursor()
    # Сирі дані (як прийшли з ESP32)
    cur.execute("""
        CREATE TABLE IF NOT EXISTS raw_measurements (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            node_id TEXT NOT NULL,
            ts_edge_ms INTEGER NOT NULL,     -- час на Pi
            ts_node_ms INTEGER NOT NULL,     -- час на ESP32 (millis)
            t_c REAL,
            h_rel REAL,
            light_raw INTEGER,
            light_norm REAL
        )
    """)
    # Нормалізовані дані + якість
    cur.execute("""
        CREATE TABLE IF NOT EXISTS normalized_data (
```

```
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            node_id TEXT NOT NULL,
            ts_edge_ms INTEGER NOT NULL,
            t_c REAL,
            h_rel REAL,
            light_norm REAL,
            q_tinyint INTEGER,    -- інтегральний прапор якості (0..3)
            q_flags TEXT          -- текстовий опис якості
        )
    """)
    # Контекстні змінні та стан
    cur.execute("""
        CREATE TABLE IF NOT EXISTS context_state (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            node_id TEXT NOT NULL,
            ts_edge_ms INTEGER NOT NULL,
            heat_risk TEXT,
            cond_risk TEXT,
            comfort TEXT,
            state TEXT            -- NORMAL / WARNING / ALARM
        )
    """)
    # Події переходу станів
    cur.execute("""
        CREATE TABLE IF NOT EXISTS state_events (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            node_id TEXT NOT NULL,
            ts_edge_ms INTEGER NOT NULL,
            state_from TEXT,
            state_to TEXT,
            reason TEXT
        )
    """)
    conn.commit()


# ---------------------- ДОПОМІЖНІ ФУНКЦІЇ ----------------------

def ema_update(node_id: str, key: str, value: float) -> float:
    """Оновлення експоненційного ковзного середнього для параметра key вузла node_id."""
    if key not in ema_state[node_id]:
        ema_state[node_id][key] = value
    else:
        prev = ema_state[node_id][key]
        ema_state[node_id][key] = EMA_ALPHA * value + (1.0 - EMA_ALPHA) * prev
    return ema_state[node_id][key]
def preprocess_and_normalize(payload: dict):
    """
    Попередня обробка та нормалізація:
    - валідація структури
    - контроль діапазонів
    - згладжування
    - приведення до канонічного діапазону
    Повертає (node_id, t_norm, h_norm, light_norm_smooth, q_int, q_flags) або None, якщо
пакет некоректний.
    """
    required = ["node_id", "ts_ms", "t_c", "h_rel", "light_raw", "light_norm"]
    for k in required:
        if k not in payload:
            print(f"[WARN] Пропущене поле {k} у пакеті: {payload}")
            return None

    node_id = str(payload["node_id"])
    ts_node_ms = int(payload["ts_ms"])
    t_c = float(payload["t_c"])
    h_rel = float(payload["h_rel"])
    light_norm = float(payload["light_norm"])
    # Початкові прапори якості
    q_flags = []
    q_score = 3  # 3 – найкраща якість, 0 – найгірша
    # Простий контроль діапазонів
```

```
    if t_c < -40 or t_c > 80:
        q_flags.append("t_out_of_range")
        q_score = min(q_score, 1)
    if h_rel < 0 or h_rel > 100:
        q_flags.append("h_out_of_range")
        q_score = min(q_score, 1)
    if light_norm < 0 or light_norm > 1.2:
        q_flags.append("light_out_of_range")
        q_score = min(q_score, 1)
    # Якщо зовсім нереальні дані – відкидаємо
    if q_score <= 0:
        print(f"[WARN] Сильно аномальний пакет від {node_id}, ігноруємо")
        return None
    # Згладжування
    t_smooth = ema_update(node_id, "t_c", t_c)
    h_smooth = ema_update(node_id, "h_rel", h_rel)
    light_smooth = ema_update(node_id, "light_norm", light_norm)
    if t_smooth != t_c or h_smooth != h_rel or light_smooth != light_norm:
        q_flags.append("ema_filtered")
        q_score = min(q_score, 2)
    # Нормалізація до [0;1] (прикладна, для контексту)
    # Припустимо, що робочий діапазон температури – 0..40 °C
    t_norm = (t_smooth - 0.0) / 40.0
    t_norm = max(0.0, min(1.0, t_norm))
    # Вологість – 0..100 %
    h_norm = h_smooth / 100.0
    h_norm = max(0.0, min(1.0, h_norm))
    # Освітленість уже прийшла як norm ~[0;1], але згладжена
    light_norm_smooth = max(0.0, min(1.0, light_smooth))
    return {
        "node_id": node_id,
        "ts_node_ms": ts_node_ms,
        "t_c": t_smooth,
        "h_rel": h_smooth,
        "light_norm": light_norm_smooth,
        "t_norm": t_norm,
        "h_norm": h_norm,
        "q_score": q_score,
        "q_flags": ",".join(q_flags) if q_flags else "ok"
    }
def compute_context_and_state(rec: dict):
    """
    Формування контекстних змінних та інтегрального стану.
    Простий rule-based підхід:
      - heat_risk: low/medium/high
      - cond_risk: low/medium/high
      - comfort: good/neutral/bad
      - state: NORMAL/WARNING/ALARM
    """
    node_id = rec["node_id"]
    t = rec["t_c"]
    h = rec["h_rel"]
    # ---------- Ризик перегріву ----------
    if t < 24:
        heat_risk = "low"
    elif 24 <= t <= 28:
        heat_risk = "medium"
    else:
        heat_risk = "high"
    # ---------- Ризик конденсації ----------
    if h < 60:
        cond_risk = "low"
    elif 60 <= h <= 80:
        cond_risk = "medium"
    else:
        if 5 <= t <= 25:
            cond_risk = "high"
        else:
            cond_risk = "medium"
    # ---------- Тепловий комфорт ----------
    if 21 <= t <= 25 and 40 <= h <= 60:
```

```
        comfort = "good"
    elif 19 <= t <= 28 and 30 <= h <= 70:
        comfort = "neutral"
    else:
        comfort = "bad"
    # ---------- Інтегральний стан ----------
    # Правило:
    # - ALARM, якщо heat_risk == high або cond_risk == high або comfort == bad при дуже
високій T
    # - WARNING, якщо хоча б один ризик == medium
    # - NORMAL, якщо всі ризики low і comfort good/neutral
    if heat_risk == "high" or cond_risk == "high" or (comfort == "bad" and t > 30):
        state = "ALARM"
    elif heat_risk == "medium" or cond_risk == "medium" or comfort == "bad":
        state = "WARNING"
    else:
        state = "NORMAL"

    return {
        "node_id": node_id,
        "heat_risk": heat_risk,
        "cond_risk": cond_risk,
        "comfort": comfort,
        "state": state
    }
def store_all(raw_payload: dict, norm: dict, ctx: dict):
    """Запис у всі таблиці: сирі дані, нормалізовані, контекст + події."""
    global conn, prev_state
    cur = conn.cursor()
    ts_edge_ms = int(time.time() * 1000)
    # 1. Сирі дані
    cur.execute("""
        INSERT INTO raw_measurements (node_id, ts_edge_ms, ts_node_ms, t_c, h_rel,
light_raw, light_norm)
        VALUES (?, ?, ?, ?, ?, ?, ?)
    """, (
        raw_payload["node_id"],
        ts_edge_ms,
        int(raw_payload["ts_ms"]),
        float(raw_payload["t_c"]),
        float(raw_payload["h_rel"]),
        int(raw_payload["light_raw"]),
        float(raw_payload["light_norm"])
    ))
    # 2. Нормалізовані дані
    cur.execute("""
        INSERT INTO normalized_data (node_id, ts_edge_ms, t_c, h_rel, light_norm,
q_tinyint, q_flags)
        VALUES (?, ?, ?, ?, ?, ?, ?)
    """, (
        norm["node_id"],
        ts_edge_ms,
        float(norm["t_c"]),
        float(norm["h_rel"]),
        float(norm["light_norm"]),
        int(norm["q_score"]),
        norm["q_flags"]
    ))
    # 3. Контекст і стан
    cur.execute("""
        INSERT INTO context_state (node_id, ts_edge_ms, heat_risk, cond_risk, comfort,
state)
        VALUES (?, ?, ?, ?, ?, ?)
    """, (
        ctx["node_id"],
        ts_edge_ms,
        ctx["heat_risk"],
        ctx["cond_risk"],
        ctx["comfort"],
        ctx["state"]
    ))
```

```
    # 4. Подія переходу стану
    node_id = ctx["node_id"]
    new_state = ctx["state"]
    old_state = prev_state.get(node_id)
    if old_state is None:
        prev_state[node_id] = new_state
    elif old_state != new_state:
        reason = f"state_change:{old_state}->{new_state}"
        cur.execute("""
            INSERT INTO state_events (node_id, ts_edge_ms, state_from, state_to, reason)
            VALUES (?, ?, ?, ?, ?)
        """, (
            node_id,
            ts_edge_ms,
            old_state,
            new_state,
            reason
        ))
        prev_state[node_id] = new_state
    conn.commit()
    # Короткий лог в консоль
    print(f"[{node_id}] T={norm['t_c']:.2f}°C H={norm['h_rel']:.1f}% "
          f"heat={ctx['heat_risk']}  cond={ctx['cond_risk']}  comfort={ctx['comfort']}
state={ctx['state']}")
# ---------------------- MQTT CALLBACKS ----------------------
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print(f"[MQTT] Підключено до брокера {MQTT_HOST}:{MQTT_PORT}")
        client.subscribe(MQTT_TOPIC)
        print(f"[MQTT] Підписка на {MQTT_TOPIC}")
    else:
        print(f"[MQTT] Помилка підключення, код rc={rc}")
def on_message(client, userdata, msg):
    try:
        payload_str = msg.payload.decode('utf-8')
        data = json.loads(payload_str)
    except Exception as e:
        print(f"[ERROR] Неможливо розпарсити JSON: {e}")
        return
    # Попередня обробка та нормалізація
    norm = preprocess_and_normalize(data)
    if norm is None:
        return
    # Обчислення контекстних змінних і стану
    ctx = compute_context_and_state(norm)
    # Збереження в БД
    store_all(data, norm, ctx)
# ---------------------- ОСНОВНИЙ ЦИКЛ ----------------------
def signal_handler(sig, frame):
    print("\n[INFO] Завершення роботи...")
    if mqtt_client is not None:
        mqtt_client.disconnect()
    if conn is not None:
        conn.close()
    sys.exit(0)
def main():
    global mqtt_client
    print("[INIT] Ініціалізація бази даних...")
    init_db()
    print("[INIT] Ініціалізація MQTT-клієнта...")
    mqtt_client = mqtt.Client()
    mqtt_client.on_connect = on_connect
    mqtt_client.on_message = on_message
    mqtt_client.connect(MQTT_HOST, MQTT_PORT, keepalive=60)
    print("[RUN] Запуск основного циклу обробки...")
    mqtt_client.loop_forever()
if __name__ == "__main__":
    signal.signal(signal.SIGINT, signal_handler)
    main()
```

Додаток Г

Апробація отриманих результатів

МЕТОД ІНТЕГРАЦІЇ ДАНИХ СЕНСОРНИХ МЕРЕЖ У КОНТЕКСТНУ МОДЕЛЬ ЦИФРОВОГО ДВІЙНИКА

Шинкарчук Микола Богданович

Магістр спеціальності “Комп’ютерні науки”

Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович

к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління

Західноукраїнський національний університет, Україна

Вступ

Сучасний етап розвитку кіберфізичних систем характеризується активним dпровадженням цифрових двійників, які забезпечують безперервний моніторинг, аналіз та оптимізацію роботи складних об’єктів протягом dсього життєвого циклу. Цифрові двійники застосовуються у промисловості, енергетиці, транспорті, розумних містах, аграрному секторі та багатьох інших галузях, де необхідно в режимі, наближеному до реального часу, відстежувати стан обладнання, прогнозувати відмови та підтримувати прийняття рішень на основі даних.

Для побудови цифрового двійника потрібна розвинена інфраструктура Інтернету речей та сенсорних мереж, яка генерує потоки телеметричних даних від фізичних об’єктів до віртуальної моделі.

Окремою проблемою є перехід від низькорівневих вимірювань до контекстних змінних, інтегральних станів та подій, які безпосередньо використовуються для моніторингу та підтримки прийняття рішень. Тому актуальним є розроблення методу інтеграції даних сенсорних мереж у

цифровий двійник, який поєднує попередню обробку та нормалізацію вимірювань із побудовою контекстної моделі.

Метод попередньої обробки та нормалізації даних сенсорних мереж

Першим кроком інтеграції є попередня обробка сенсорних даних. Потоки вимірювань можуть містити пропуски, одиничні збої, короткочасні викиди та шум, пов’язаний з особливостями сенсорів та каналів зв’язку. Для підвищення якості даних застосовується послідовність операцій:

1. До часових рядів вимірювань застосовуються згладжування ковзним середнім, що зменшує випадкові коливання без суттєвого спотворення тренду.

2. Вимірювання з некоректними мітками часу, відсутні пакети або явно некоректні значення позначаються як пропуски й заміщуються інтерполяцією з врахуванням сусідніх точок.

3. Для кожного параметра визначаються допустимі діапазони та базові статистичні характеристики; вимірювання, які переходять ці обмеження, маркуються як аномальні і обробляються окремо, тобто відкидаються або замінюються інтерпольованими значеннями.

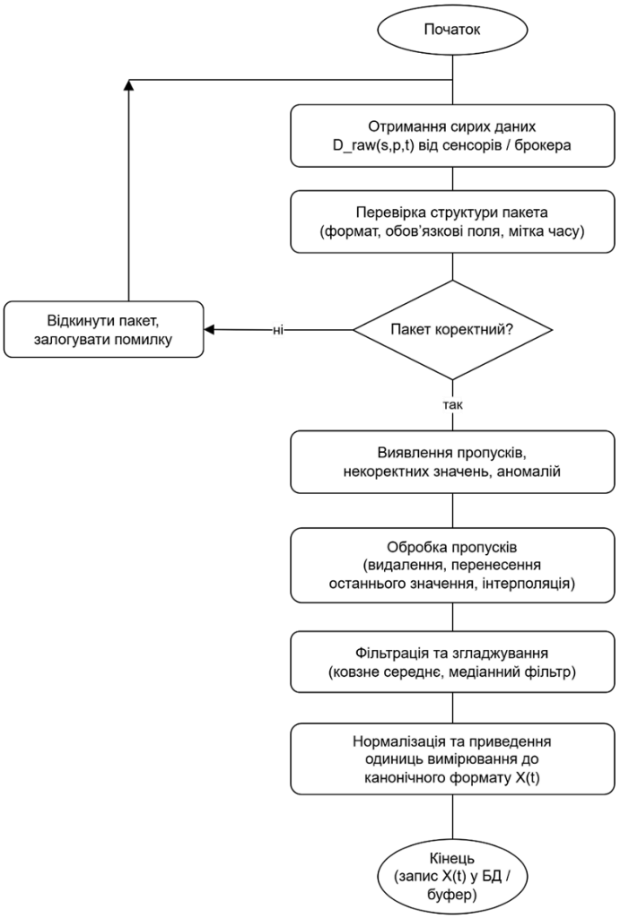


Рис. 1. **Алгоритм попередньої обробки та нормалізації даних сенсорних мереж**

Після очищення даних виконується нормалізація, яка приводить вимірювання до уніфікованих шкал. Для параметрів з різними одиницями вимірювання застосовується лінійне перетворення до відрізка 0;1 або до стандартизованого діапазону з нульовим середнім та одиничним відхиленням. Це необхідно для подальшого об'єднання параметрів у контекстні змінні та порівняння їх впливу

Побудова та оновлення контекстної моделі

На основі нормалізованих векторів вимірювань формується контекстна модель цифрового двійника. Для кожної контекстної змінної задаються правила інтерпретації, які відображають діапазони значень сенсорних параметрів у визначені категорії. Наприклад, температурі може відповідати розбиття на низьку, нормальну та високу з врахуванням заданих порогів.

Контекстні змінні будуються як:

– логічні предикати істинна / хибна;

– лінгвістичні змінні з нечіткими функціями належності;

– агреговані показники, що поєднують кілька параметрів.

Інтегральний стан цифрового двійника визначається набором правил, які співвідносять комбінації контекстних змінних з рівнями “Норма”, “Попередження”, “Тривога”. Для уникнення частих коливань стану застосовується гістерезис та часові пороги: перехід у більш критичний стан відбувається миттєво, а повернення в менш критичний вимагає певного періоду стабільних вимірювань.

Оновлення контекстної моделі здійснюється, коли надходить новий вектор вимірювань від сенсорної мережі. На кожному кроці відбувається:

– застосування фільтрації та нормалізації;

– обчислення значень контекстних змінних;

– визначення нового інтегрального стану;

– реєстрація подій.

Рис. 2. **Алгоритм побудови та оновлення контекстної моделі цифрового двійника**

Висновки. Було запропоновано метод інтеграції даних сенсорних мереж у контекстну модель цифрового двійника, який поєднує формалізований опис контекстної моделі з удосконаленим підходом до попередньої обробки і нормалізації даних.

Використання методів фільтрації, виявлення пропусків та аномальних значень зменшує вплив шуму й нестабільності каналів зв'язку, а нормалізація параметрів дає можливість гнучко комбінувати різнорідні сенсорні дані. Правила побудови контекстних змінних і станів забезпечують перехід від низькорівневих вимірювань до симслових подій та індикаторів, що напряму придатні для моніторингу, діагностики та підтримки прийняття рішень у цифровому двійнику.

Список використаних джерел:

1. Grieves M., Vickers J. Digital twin: mitigating unpredictable, undesirable emergent behavior in complex systems. In: Kahlen F.-J., Flumerfelt S., Alves A. (eds.). Transdisciplinary perspectives on complex systems. Cham : *Springer*, 2017. P. 85–113.
2. Kritzinger W., Karner M., Traar G., Henjes J., Sihn W. Digital twin in manufacturing: a categorical literature review and classification. **IFAC-PapersOnLine**. 2018. Vol. 51, No. 11. P. 1016–1022.
3. Gubbi J., Buyya R., Marusic S., Palaniswami M. Internet of things (IoT): a vision, architectural elements, and future directions. *Future Generation Computer Systems*. 2013. Vol. 29, No. 7. P. 1645–1660.
4. Perera C., Zaslavsky A., Christen P., Georgakopoulos D. Context aware computing for the Internet of Things: a survey. *IEEE Communications Surveys & Tutorials*. 2014. Vol. 16, No. 1. P. 414–454.
5. Blázquez-García A., Conde A., Mori U., Lozano J. A. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*. 2021. Vol. 54, No. 3. Article 56.

6. Tao F., Cheng J., Qi Q., Zhang M., Zhang H., Sui F. Digital twin-driven product design, manufacturing and service. *Computers in Industry*. 2018. Vol. 100. P. 128–143.
7. Qi Q., Tao F. Digital twin and big data towards smart manufacturing and Industry 4.0: 360 degree comparison. *IEEE Access*. 2018. Vol. 6. P. 3585–3593.

ПРОГРАМНИЙ МОДУЛЬ ІНТЕГРАЦІЇ ДАНИХ СЕНСОРНИХ МЕРЕЖ ДЛЯ ЦИФРОВОГО ДВІЙНИКА НА БАЗІ ESP32 ТА RASPBERRY PI

Шинкарчук Микола Богданович
Магістр спеціальності “Комп’ютерні науки”
Західноукраїнський національний університет, Україна
Науковий керівник: Осолінський Олександр Романович
ORCID ID: 0000-0002-0136-395X
к.т.н., доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет
Україна

Вступ

Розробка цифрових двійників потребує не лише контекстної моделі, а і надійної програмно-апаратної інфраструктури, яка забезпечує збір, попередню обробку й доставку даних в режимі, наближеному до реального часу. На рівні периферії (edge) цю роль відіграють сенсорні вузли та вузли шлюзів, які реалізують перетворення телеметрії в структуровані потоки даних для цифрового двійника.

В даній роботі представлено модуль попередньої обробки, нормалізації та контекстної інтерпретації даних, який забезпечує взаємодію з контекстною моделлю цифрового двійника через стандартизовані інтерфейси.

Архітектура програмного модуля інтеграції

Архітектура побудована за багатошаровим підходом. На фізичному рівні працюють сенсорні вузли на базі ESP32, до яких підключено датчики. Кожен вузол збирає дані з датчиків, формує телеметричні повідомлення та передає їх по Wi-Fi.

На рівні edge-інфраструктури використовується Raspberry Pi як шлюз і вузол попередньої обробки. Він приймає дані від кількох ESP32, виконує фільтрацію, нормалізацію, формує контекстні змінні й передає структуровані дані до сервісів цифрового двійника. Комунікація між сенсорними вузлами та Raspberry Pi реалізовано через MQTT-брокер.

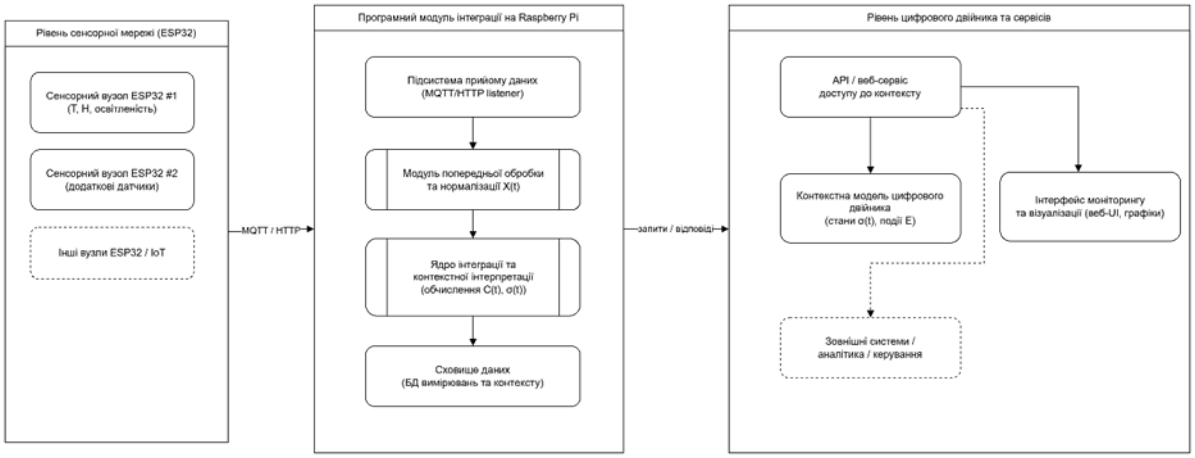


Рис 1. **Архітектура програмного модуля інтеграції даних цифрового двійника на базі Raspberry Pi та ESP32**

Програмна частина вузла ESP32 виконує такі основні функції:

– ініціалізація датчиків та мережевого інтерфейсу Wi-Fi;
– періодичний збір вимірювань з визначеним інтервалом дискретизації;
– формування телеметричного повідомлення у форматі JSON;
– передавання даних на адресу шлюзу Raspberry Pi або MQTT-брокера;
– контроль підключення до мережі, обробка помилок датчиків.

На Raspberry Pi реалізовано наступні логічні сервіси:

– Сервіс приймання даних.
– Сервіс попередньої обробки.
– Сервіс нормалізації та контекстної інтерпретації.
– Сервіс збереження та інтерфейсів.

Веб-інтерфейс і взаємодія з цифровим двійником

Для моніторингу параметрів розроблено веб-інтерфейс, який відображає:

– поточні значення сенсорних параметрів;

– контекстні змінні;

– інтегральний стан (“Норма”, “Попередження”, “Тривога”) у вигляді індикаторів типу світлофор;

– історичні графіки вимірювань та станів за обраний період.

Веб-клієнт взаємодіє з REST-API Raspberry Pi, отримуючи оновлення. Цей інтерфейс може бути інтегрований у ширшу панель цифрового двійника або використовуватися як автономний інструмент в лабораторному стенді.

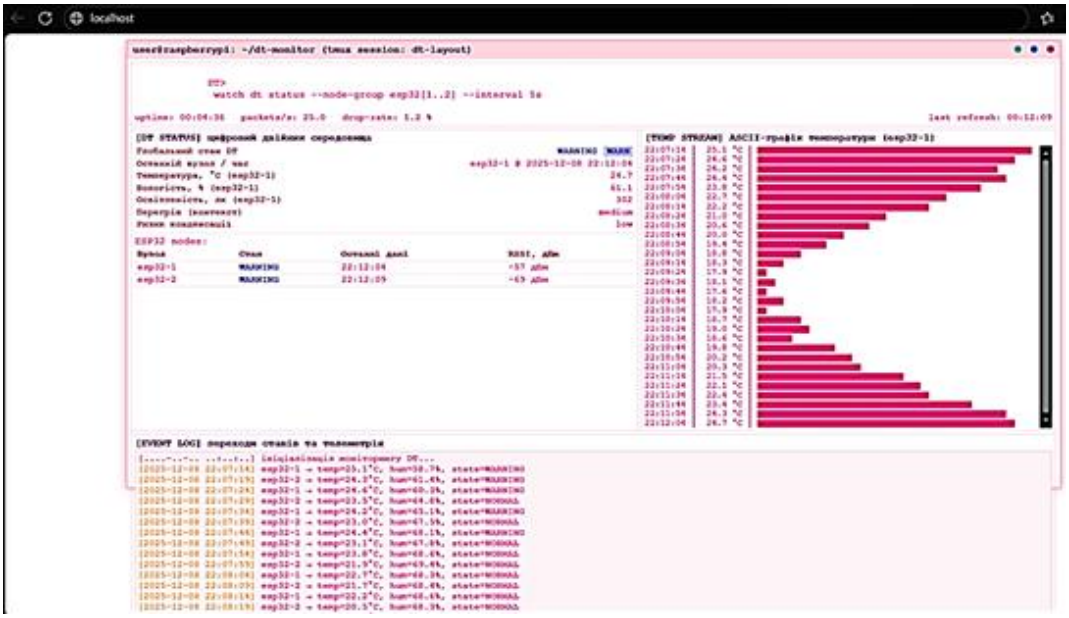


Рис 2. **Моніторинг в режимі, близькому до реального часу**

Рис 3. **Графіки часових рядів**

Експериментальні дослідження

Експериментальні дослідження проводилися на лабораторному стенді, що включав кілька вузлів ESP32 з датчиками температури, вологості та освітленості, а також один вузол Raspberry Pi, який виконував роль шлюзу та edge-модуля інтеграції. Для оцінки ефективності були порівняні:

– якість даних цифрового двійника до та після попередньої обробки;

– стабільність і коректність переходів між інтегральними станами;

– затримка між фізичною подією та оновленням стану в системі.

Рис 4. **Термінальний звіт про результати експерименту**

Результати показали, що застосування програмного модуля інтеграції дозволяє суттєво знизити кількість некоректних вимірювань, покращити структурованість даних цифрового двійника та підвищити чутливість до критичних змін стану при прийнятній затримці оновлення

Висновки. Було представлено архітектуру та програмну реалізацію модуля інтеграції даних сенсорних мереж у цифровий двійник на базі ESP32 та Raspberry Pi. Запропоноване рішення охоплює повний цикл обробки даних на рівні периферії: від збору вимірювань сенсорними вузлами до формування контекстних змінних і інтегральних станів, доступних через стандартизовані інтерфейси.

Список використаних джерел:

8. Hu W., Zhang T., Deng X., Liu Z., Tan J. Digital twin: a state-of-the-art review of its enabling technologies, applications and challenges. *Journal of Intelligent Manufacturing and Special Equipment*. 2021. Vol. 2, No. 1. P. 1–34.
9. Yick J., Mukherjee B., Ghosal D. Wireless sensor network survey. *Computer Networks*. 2008. Vol. 52, No. 12. P. 2292–2330.
10. Жураковський Б. Ю., Зенів І. О. Технології інтернету речей : *навч. посіб*. Київ : КПІ ім. Ігоря Сікорського, 2021. 271 с.
11. Zanella A., Bui N., Castellani A., Vangelista L., Zorzi M. Internet of things for smart cities. *IEEE Internet of Things Journal*. 2014. Vol. 1, No. 1. P. 22–32.
12. Sodhro A. H., Pirbhulal S., de Albuquerque V. H. C. Towards an optimal resource management for IoT based green and sustainable smart cities. *Journal of Cleaner Production*. 2019. Vol. 220. P. 1167–1179.
13. Alshathri S., Hemdan E. E. D., El-Shafai W., Sayed A. Digital twin-based automated fault diagnosis in industrial IoT applications. Computers, *Materials & Continua. 2023*. Vol. 75, No. 1. P. 183–196.
14. Islam M., Dukyil A. S., Alyahya S., Habib S. An IoT enable anomaly detection system for smart city surveillance. *Sensors*. 2023. Vol. 23, No. 4. 2358.

Громадська організація «Міжнародний центр наукових досліджень».
Номер запису в Реєстрі громадських об'єднань: 1499141.
Адреса: вул. Зодчих, буд. 40, офіс 103; м. Вінниця, Вінницька обл., 21037
Організація функціонує як відокремлений підрозділ ТОВ «UKRLOGOS Group».
ЄДРПОУ: 44574528
IBAN: UA433052990000026002046104529
Банк ВФ АТ КБ «ПриватБанк»; МФО 305299
Свідоцтво суб'єкта видавничої справи: ДК № 7880 від 22.06.2023.

ДОВІДКА

ПРО ПРИЙНЯТТЯ НАУКОВОЇ РОБОТИ ДО ПУБЛІКАЦІЇ

13.12.2025

Шановний(і) авторе(и):
Шинкарчук Микола Богданович,

Організаційний комітет з радістю повідомляє, що наукову роботу «ПРОГРАМНИЙ МОДУЛЬ ІНТЕГРАЦІЇ ДАНИХ СЕНСОРНИХ МЕРЕЖ ДЛЯ ЦИФРОВОГО ДВІЙНИКА НА БАЗІ ESP32 ТА RASPBERRY PI» прийнято до публікації в збірнику за матеріалами VI Міжнародної наукової конференції «Теорія модернізації в контексті сучасної світової науки» (19.12.2025; м. Івано-Франківськ, Україна).

Опублікована робота буде доступна з 19.12.2025 за посиланням:
https://archives.mcnd.org.ua/index.php/conference-proceeding/issue/view/19.12.2025

. . . . . .

Електронні сертифікати учасників конференції будуть доступні з 19 грудня.

Конференцію зареєстровано в базі даних науково-технічних заходів УкрІНТЕІ (Посвідчення № 603 від 10.06.2025). Матеріали конференції знаходитимуться в відкритому доступі (Open Access) на умовах ліцензії CC BY-SA 4.0 International.

З повагою,

Віце-президент МЦНД
Голова оргкомітету конференції
НАСТАСІЯ РАБЕЙ

МЦНД
МІЖНАРОДНИЙ ЦЕНТР НАУКОВИХ ДОСЛІДЖЕНЬ

МОЛОДІЖНА НАУКОВА ЛІГА

Громадська організація «Молодіжна наукова ліга».
Номер запису в Єдиному державному реєстрі громадських об'єднань: 1508433.
Адреса: вул. Зодчих, буд. 40, офіс 103; м. Вінниця, Вінницька обл., 21037
Організація функціонує як відокремлений підрозділ ТОВ «UKRLOGOS Group».
ЄДРПОУ: 44574528
IBAN: UA783052990000026003016111950
Банк: ВФ АТ КБ «ПриватБанк»; МФО 305299
Свідоцтво суб'єкта видавничої справи: ДК № 7172 від 21.10.2020.

ДОВІДКА

ПРО ПРИЙНЯТТЯ ТЕЗ ДО ПУБЛІКАЦІЇ

13.12.2025

Шановний(і) авторе(и):
Шинкарчук Микола Богданович,

Організаційний комітет з радістю повідомляє, що тези «МЕТОД ІНТЕГРАЦІЇ ДАНИХ СЕНСОРНИХ МЕРЕЖ У КОНТЕКСТНУ МОДЕЛЬ ЦИФРОВОГО ДВІЙНИКА» прийняті до публікації в збірнику за матеріалами IX Всеукраїнської студентської наукової конференції «Науковий простір: аналіз, сучасний стан, тренди та перспективи» (19.12.2025, м. Вінниця, Україна).

Опубліковані тези будуть доступні з 19.12.2025 за посиланням:
https://archive.liga.science/index.php/conference-proceedings/issue/view/ukr-19.12.2025

. . . . . .

Електронні сертифікати учасників конференції та подяки науковим керівникам будуть доступні з 19 грудня. Розсилка замовлених друкованих примірників, сертифікатів та подяк відбудеться до 16 січня.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та Інформаційному бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення № 461 від 10.06.2025).

З повагою,

Директор Молодіжної наукової ліги
Голова оргкомітету конференції
ІГОР КОРЕНЮК