Хмельницький національний університет

Факультет інформаційних технологій

Кафедра автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

КВАЛІФІКАЦІЙНА РОБОТА

Інтелектуальна система моніторингу технологічних даних

Назва теми

Рівень вищої освіти	Перший (бакалаврський)
Галузь знань	17 – Електроніка, автоматизація та електронні комунікації
Спеціальність	174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма	«Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

КвРАКІТР. 2023152.01.16.ПЗ

Шифр

Виконав здобувач 3 курсу, група АКІТРс-23-1 (Шифр) — Підпис — Артем ПЕРЕЙМА (Ім'я, ПРІЗВИЩЕ)

Керівник канд. техн. наук, доц. (Науковий ступінь, учене звання) — Підпис — Микола ФЕДУЛА (Ім'я, ПРІЗВИЩЕ)

Нормоконтролер канд. техн. наук, доц. (Науковий ступінь, учене звання) — Підпис — Галина РАДЕЛЬЧУК (Ім'я, ПРІЗВИЩЕ)

До захисту допускаю:

Завідувач кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки — Підпис — Людмила КОРЕЦЬКА (Ім'я, ПРІЗВИЩЕ)

17.06.2026 р.

Дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Автоматизації, комп'ютерно-інтегрованих технологій та робототехніки
Рівень вищої освіти перший (бакалаврський)
Галузь знань 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітня програма Автоматизація, комп'ютерно-інтегровані технології та робототехніка

ЗАТВЕРДЖУЮ
Завідувач кафедри АКІТтаР
Людмила КОРЕЦЬКА
07 лютого 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Переймі Артему Юрійовичу
Прізвище, ім'я, по батькові студента

1 Тема роботи Інтелектуальна система моніторингу технологічних даних на прикладі чистої кімнати GMP класу С фармацевтичного виробництва

Керівник роботи Федула Микола Васильович, канд. техн. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, учене звання

Затверджено наказом ректора університету від 20.01 2026 р. № 7

2 Строк подання студентом роботи на кафедру 02.06.2026 р.

3 Вихідні дані до роботи Матеріали переддипломної практики

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Огляд предметної області. Передача технологічних даних від контролерів. Проєктування інтелектуальної системи моніторингу технологічних даних. Алгоритмічне та програмне проєктування системи. Випробування програмного прототипу та аналіз результатів.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Структурна схема системи, схема передачі даних від контролерів, ER-модель бази даних, блок-схема конвеєра ШІ, контур операторських рішень, графіки трендів, прогнозу, аномальності, масштабування та екранні форми панелі оператора.

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Антиплагіат	Федула М. В., доцент кафедри АКІТтаР		
Нормоконтроль	Радельчук Г. І., доцент кафедри АКІТтаР		

7 Дата видачі завдання 07 лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

Назва розділу кваліфікаційної роботи	Строк виконання	Примітка
1 Вибір та затвердження теми кваліфікаційної роботи; розробка завдання на кваліфікаційну роботу; складання календарного графіка виконання кваліфікаційної роботи	04.09.2025 – 25.09.2025	виконано
2 Ознайомлення з предметною областю. Формулювання мети та задач дослідження визначення об'єкта та предмета дослідження	26.09.2025 – 24.10.2025	виконано
3 Робота над розділом 1 - аналіз предметної області та постановка задачі	25.10.2025 – 12.12.2025	виконано
4 Робота над розділом 2 – проєктування інтелектуальної системи моніторингу	13.12.2025 – 06.02.2026	виконано
5 Робота над розділом 3 – програмна реалізація та тестування	07.02.2026 – 03.04.2026	виконано
6 Остаточне коригування кваліфікаційної роботи з урахуванням зауважень керівника; оформлення кваліфікаційної роботи як документа відповідно до вимог	04.04.2026 – 08.05.2026	виконано
7 Отримання супровідних документів (відгуку керівника, рецензії, довідки про перевірку на плагіат); нормоконтроль	11.05.2026 – 22.05.2026	виконано
8 Підготовка до захисту та захист кваліфікаційної роботи	Червень 2026 року	виконано

Студент _____ Артем ПЕРЕЙМА
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Микола ФЕДУЛА
Підпис Ім'я, ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інтелектуальна система моніторингу технологічних даних».

Автор роботи: Артем ПЕРЕЙМА.

Керівник роботи: Микола ФЕДУЛА.

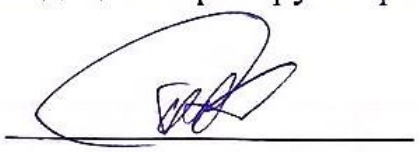
Пояснювальна записка: 65 с., 13 рис., 2 дод., 40 джерел.

Графічна частина: 12 презентаційних слайдів.

ІНТЕЛЕКТУАЛЬНИЙ МОНІТОРИНГ, ТЕХНОЛОГІЧНІ ДАНІ, ШТУЧНИЙ ІНТЕЛЕКТ, ВИЯВЛЕННЯ АНОМАЛІЙ, ПРОГНОЗУВАННЯ.

Мета роботи: розробка інтелектуальної системи моніторингу технологічних даних, здатної приймати телеметрію від промислового шару збору даних, виявляти аномалії, класифікувати стан технологічного процесу, прогнозувати зміну параметрів та формувати рекомендації оператору.

У роботі розглянуто передавання технологічних даних від сенсорів через контролери, мікроконтролерні вузли та проміжний шлюз до рівня моніторингу й аналітики. Для кожного вимірювання передбачено збереження контексту: джерела даних, зони, сенсора, протоколу, часової мітки та ознаки якості. Розроблена система дає змогу відображати поточний стан технологічного середовища, виявляти аномальні режими, оцінювати ризик відхилення параметрів і підтримувати прийняття операторських рішень. Окремо передбачено демонстрацію масштабування для великої кількості сенсорів. Практичне значення роботи полягає у демонстрації повного шляху технологічних даних: від формування вимірювання на рівні контролера до аналітичного висновку, рекомендації оператору та фіксації прийнятого рішення.



Підпис студента

02.06.2026

Дата

ЗМІСТ

ВСТУП

Сучасні промислові та виробничі об’єкти формують великі обсяги технологічних даних, що характеризують стан обладнання, якість продукції, стабільність процесу та безпеку експлуатації. Традиційний підхід, за якого оператор спостерігає за показниками на панелі SCADA і реагує лише після виходу параметрів за межі допустимих значень, уже не відповідає вимогам до швидкості прийняття рішень, простежуваності та якості виробництва.

Особливо важливою є задача моніторингу для фармацевтичного виробництва, де від параметрів мікроклімату, каскаду тиску, кількості часток у повітрі та стану фільтрації залежить відповідність умовам GMP. Для чистих кімнат критичним є не тільки факт перевищення параметра, а й контекст: джерело даних, зона, контролер, протокол польового рівня, якість вимірювання та момент формування показника.

Актуальність роботи зумовлена переходом від простого збору даних до інтелектуального моніторингу. Такий моніторинг має не лише відображати значення параметрів, а й автоматично виявляти аномалії, оцінювати стан системи, прогнозувати зміну технологічних показників, пояснювати ситуацію оператору та фіксувати прийняті рішення.

У роботі розроблено програмний прототип інтелектуальної системи моніторингу технологічних даних. Як приклад використано чисту кімнату GMP класу C, однак архітектура системи є узагальненою і може бути адаптована до інших об’єктів: виробничих ліній, енергетичних установок, лабораторних комплексів або систем мікроклімату.

Мета роботи – розробити інтелектуальну систему моніторингу технологічних даних з промисловим шаром збору телеметрії, ШІ-аналітикою, панеллю оператора та безпечним контуром операторських рекомендацій.

Для досягнення мети необхідно виконати такі завдання:

– проаналізувати предметну область інтелектуального моніторингу технологічного обладнання;

– обґрунтувати вибір прикладу чистої кімнати GMP і визначити перелік контрольованих параметрів;

– розробити архітектуру передачі даних від ПЛК, STM32 та шлюзу Raspberry Pi до серверного рівня;

– визначити структуру MQTT-повідомлення з метаданими джерела, протоколу та якості вимірювання;

– реалізувати серверний рівень для прийому, збереження, аналізу та видачі телеметрії через REST і WebSocket;

– впровадити методи ШІ для виявлення аномалій, класифікації стану, прогнозування та формування рекомендацій;

– розробити україномовну вебпанель оператора з відображенням промислової топології та операторського контуру;

– перевірити працездатність системи unit, API та демонстраційними тестами, включно з режимом понад 10 тис. сенсорів.

Об’єкт дослідження – процес моніторингу технологічних даних промислового об’єкта. Предмет дослідження – методи та програмні засоби збору, нормалізації, збереження, аналізу і візуалізації технологічної телеметрії.

Практичне значення роботи полягає у створенні програмного прототипу, який демонструє повний шлях даних від сенсорного рівня до інтелектуальної аналітики й операторського рішення.

Практична цінність роботи полягає у поєднанні трьох аспектів які часто розглядаються окремо промислового збору даних методів інтелектуального аналізу та операторського контуру прийняття рішень. У роботі показано що для технологічного моніторингу недостатньо зберігати лише числове значення параметра. Необхідно також зберігати джерело вимірювання протокол часову мітку якість даних і зв'язок із поточним станом об'єкта.

Обраний приклад чистої кімнати GMP класу C дає змогу продемонструвати роботу системи на об’єкті, де стабільність параметрів має прямий технологічний зміст. Температура, вологість, каскад тиску, стан фільтрації та кількість часток у повітрі не є абстрактними навчальними показниками. Вони пов’язані з якістю фармацевтичного виробництва, тому система повинна не лише показувати значення, а й допомагати оцінювати ризик відхилення.

Обмеження роботи визначено як програмний прототип, що демонструє архітектуру і логіку інтелектуального моніторингу. Реальне підключення до промислового обладнання, сертифікація, валідація за GMP та промислова кібербезпека виходять за межі роботи, однак у проєкті сформовано структуру, яку можна розширити для такого впровадження.

Методи дослідження включають аналіз предметної області, моделювання архітектури системи, проєктування структури даних, програмну реалізацію прототипу, емуляцію контролерів, тестування програмних модулів та експериментальну перевірку сценаріїв роботи. Такий набір методів відповідає прикладному характеру кваліфікаційної роботи.

Очікуваним результатом є програмний комплекс, який можна показати під час захисту як цілісний технологічний сценарій: емулятор контролерів генерує дані, шлюз нормалізує повідомлення, серверний рівень зберігає і аналізує телеметрію, а панель оператора відображає стан системи, рекомендації та журнал дій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Інтелектуальний моніторинг технологічного обладнання

Інтелектуальний моніторинг технологічного обладнання передбачає безперервний збір параметрів, їх аналіз у контексті стану об’єкта та формування рекомендацій щодо експлуатації. На відміну від класичної сигналізації, така система працює з часовими рядами, історією подій, якістю даних і залежностями між параметрами.

У наукових і прикладних роботах інтелектуальний моніторинг розглядається як поєднання збору технологічних параметрів, їх структурованого збереження, аналізу стану об’єкта та підтримки прийняття рішень оператором [1-4].

У роботі Б. О. Пальчевського підкреслюється, що моніторинг технологічних, експлуатаційних і виробничих параметрів потрібний для діагностування стану компонентів обладнання, коригування технологічних параметрів і визначення термінів технічного обслуговування. Це відповідає логіці розроблюваної системи, де показники не лише зберігаються, а й аналізуються для прийняття рішень.

Сучасна система моніторингу має бути багаторівневою. На нижньому рівні працюють датчики, мікроконтролери, віддалені модулі вводу-виводу та ПЛК. На середньому рівні здійснюється агрегація і нормалізація даних. На верхньому рівні працюють програмний інтерфейс, база даних, алгоритми машинного навчання та інтерфейс оператора.

У статті А. І. Лагойди та співавторів показано актуальність інтеграції мікроконтролерних систем із SCADA. Автори виділяють проблему обмеженості збору даних безпосередньо на об’єктах експлуатації та затримок у прийнятті рішень. Для цієї роботи важливим є сам принцип: мікроконтролерний рівень забезпечує оперативний збір, а вища система виконує аналіз і візуалізацію.

У промисловому середовищі значення має також зв’язок з MES/ERP-рівнем. Матеріали щодо рішення ПРОНЕТ для автоматизації виробництва акцентують увагу на типових проблемах: дані про виробничий процес не передаються в інформаційні системи, різне обладнання не обмінюється даними, складно локалізувати причину браку, а ручне введення підсилює людський фактор. Розроблювана система вирішує ці проблеми на рівні моніторингу та аналітики.

Інтелектуальна структура починається з сенсорного рівня, який відповідає за формування фізичних вимірювань. Отримані показники передаються на контролерний рівень, що забезпечує первинне опитування датчиків та виконання локальної логіки. Далі інформація надходить на шлюзовий рівень (наприклад, через RPI-GW-01 та протокол MQTT QoS 1), завдання якого полягає в нормалізації повідомлень та їх надійній доставці до серверного рівня. Тут, на базі рішень типу FastAPI, SQLite та scikit-learn, здійснюється довготривале збереження даних, робота програмних інтерфейсів, WebSocket-з'єднань та аналіз у конвеєрі штучного інтелекту. Фінальним етапом є операторський рівень, представлений як веб-панель, яка слугує для візуалізації процесів, відображення ШІ-рекомендацій, ведення журналу дій та, головне, для прийняття оперативних рішень.

Поділ системи на рівні важливий не лише для опису архітектури, а й для розподілу відповідальності між компонентами. Сенсорний рівень відповідає за фізичне вимірювання, контролерний рівень – за первинне опитування і локальну логіку, шлюзовий рівень – за нормалізацію та передавання, серверний рівень – за збереження й аналіз, а операторський рівень – за прийняття рішень.

У реальному виробництві помилка може виникнути на будь-якому з цих рівнів. Наприклад, сенсор може давати шум, контролер може втратити зв’язок з модулем вводу-виводу, шлюз може отримати дані із затримкою, а серверний рівень може працювати з неповним набором параметрів. Тому система повинна зберігати не лише значення, а й інформацію про шлях його надходження.

Для інтелектуального моніторингу особливо важливо накопичувати історію показників. Окреме поточне значення може бути допустимим, але його тренд може вказувати на наближення до небезпечного режиму. Наприклад, повільне зростання перепаду тиску на HEPA-фільтрі може свідчити про поступове засмічення, навіть якщо поточне значення ще не перевищило граничного рівня [5, 7, 9].

Застосування методів ШІ у такій системі має допоміжний характер. Алгоритм не замінює оператора і не є регулятором. Його роль полягає в тому, щоб виділити нетипову поведінку, підсумувати контекст, запропонувати можливу причину і сформувати рекомендацію, яку людина може прийняти або відхилити.

У цьому підході простежуваність є такою ж важливою, як і точність алгоритму. Якщо система показує аномалію, оператор повинен бачити, з якого контролера надійшов показник, коли він був сформований, який протокол використовувався і яка якість вимірювання була присвоєна. Без цього інтелектуальний висновок складно перевірити.

Таким чином, інтелектуальна система моніторингу технологічних даних розглядається не як окремий алгоритм машинного навчання, а як комплексна інформаційна система, де алгоритми працюють у контексті промислової топології, часових рядів, бази даних і операторських процедур.

1.2 Чиста кімната GMP класу C як об’єкт моніторингу

Для демонстрації роботи системи обрано чисту кімнату GMP класу C. Такий об’єкт є вдалим прикладом технологічної системи, де важлива стабільність параметрів, простежуваність подій і швидке реагування на відхилення. На відміну від навчальної HVAC-системи, чиста кімната має чіткий виробничий контекст: якість середовища безпосередньо пов’язана з якістю фармацевтичної продукції.

Нормативну основу моніторингу чистої кімнати в роботі сформовано з актуальних міжнародних стандартів ISO, українських документів МОЗ щодо належної виробничої практики, європейських настанов GMP, документів WHO, PIC/S та GAMP 5 [5-17].

Основними контрольованими параметрами є температура, відносна вологість, концентрація CO2, каскад диференційного тиску, потужність припливно-витяжної установки, кількість часток у повітрі, перепад тиску на HEPA-фільтрах, швидкість повітряного потоку та стан шлюзових дверей. Кожен параметр має не тільки числове значення, а й джерело: контролер, зона, сенсор, протокол і ознаку якості даних.

Чиста кімната як об’єкт автоматизації має розподілену природу. Частина параметрів може зчитуватися ПЛК через PROFINET IO, частина – локальними STM32-вузлами через ADC, I2C або GPIO. Це створює потребу в шлюзовому рівні, який уніфікує формат даних і передає їх на серверний рівень незалежно від фізичного способу отримання.

На відміну від звичайної системи мікроклімату, де відхилення показників впливає переважно на комфорт персоналу, у чистій кімнаті стандарту GMP кожен параметр має суворе виробниче і регуляторне значення для стабільності технологічного процесу, умов зберігання матеріалів або відповідності внутрішнім процедурам якості. Для цього система безперервно відстежує температуру та вологість через ПЛК/PROFINET або STM32 з метою контролю мікроклімату та запобігання відхиленням умов виробництва. Паралельно з цим рівень CO2, що надходить від ПЛК або локального вузла, виступає непрямим показником присутності персоналу в зоні. Особлива увага приділяється повітряним потокам: каскад тиску, який контролюється через ПЛК/PROFINET, критично важливий для підтримки постійного позитивного тиску класу C, тоді як моніторинг стану дверей шлюзу через STM32/GPIO дозволяє вчасно фіксувати ризики порушення цього бар'єра. Нарешті, безпеку середовища замикає контроль чистоти повітря шляхом підрахунку часток через STM32/I2C

та оперативна діагностика ефективності фільтрації на основі показників HEPA, які зчитуються через STM32/ADC.

Чиста кімната відрізняється від звичайної системи мікроклімату тим, що її параметри мають виробниче і регуляторне значення. Якщо в офісному приміщенні відхилення температури переважно впливає на комфорт, то у фармацевтичному виробництві воно може вплинути на стабільність технологічного процесу, умови зберігання матеріалів або відповідність внутрішнім процедурам якості.

Каскад диференційного тиску є одним із ключових параметрів. Його задача полягає у підтриманні правильного напрямку руху повітря між зонами. Якщо тиск знижується або змінюється напрямок перепаду, зростає ризик перетікання забрудненого повітря в чистішу зону. Саме тому для такого параметра важливо бачити не лише останнє значення, а й тренд і джерело вимірювання.

Стан HEPA-фільтрів доцільно контролювати через перепад тиску. Занадто низький перепад може свідчити про проблему з ущільненням або каналом вимірювання, а занадто високий – про засмічення фільтра або неправильний режим вентиляції. У прототипі цей параметр надходить від STM32-вузла, що демонструє роботу локального мікроконтролерного збору даних.

Стан шлюзових дверей є прикладом дискретного параметра. На відміну від температури або вологості, він не має плавної динаміки, але може пояснювати різкі зміни тиску або кількості часток. Тому система повинна підтримувати не тільки числові аналогові показники, а й подієві дані від дискретних входів.

Контроль часток у повітрі є показником чистоти середовища. У реальних системах такі дані можуть надходити від спеціалізованих лічильників часток, однак для демонстраційного прототипу достатньо емулювати відповідний сенсорний вузол. Важливо, що архітектура не залежить від конкретної фізичної моделі сенсора.

Таким чином, об’єкт моніторингу містить параметри різної природи: аналогові, дискретні, подієві, повільні та швидкі. Це робить чисту кімнату

зручним прикладом для демонстрації універсальної системи моніторингу технологічних даних.

1.3 Передача технологічних даних від контролерів

Передача даних у системі побудована як багаторівневий процес. На польовому рівні сенсори підключені до ПЛК або STM32-вузлів. ПЛК працює циклічно: у циклі опитування він зчитує входи, виконує логіку, оновлює виходи та формує станові змінні. STM32-вузли виконують локальне опитування низькорівневих сенсорів, перетворення сигналів і первинну фільтрацію.

PROFINET у роботі розглядається як польовий промисловий протокол між ПЛК та віддаленими модулями вводу-виводу. Реальний стек PROFINET у програмному прототипі не реалізується, оскільки задача роботи полягає не у створенні драйвера польової шини, а у демонстрації архітектури збору, нормалізації та інтелектуального аналізу технологічних даних.

Промислова передача даних повинна враховувати вимоги кібербезпеки, сумісності та стандартизованого опису даних, що відображено у IEC 62443, OPC UA, PROFINET, MQTT і Sparkplug [18-23].

Шлюз Raspberry Pi виконує роль проміжного вузла. Він агрегує дані від ПЛК і STM32, додає метадані джерела, формує уніфіковане JSON-повідомлення і публікує його в брокер MQTT з QoS 1. Серверний рівень підписується на теми MQTT, зберігає показники у SQLite, запускає конвеєр ШІ і передає живі оновлення на панель оператора через WebSocket.

У такій архітектурі MQTT не замінює PROFINET. PROFINET залишається польовим рівнем для промислового обміну між ПЛК та модулями вводу-виводу, а MQTT використовується як легкий протокол передавання нормалізованої телеметрії до інформаційного рівня системи.

Узагальнену схему передачі даних від контролерів до серверного рівня наведено на рисунку 1.1.

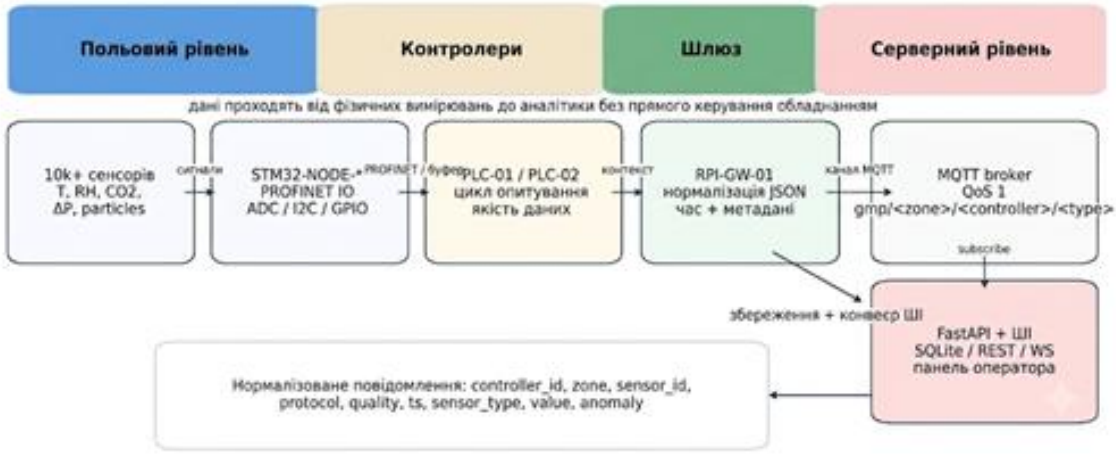


Рисунок 1.1 – Схема передачі даних від контролерів до серверного рівня

На рівні ПЛК дані формуються у межах циклу опитування. Типовий цикл включає зчитування входів, виконання програми керування, оновлення виходів і підготовку змінних для зовнішнього обміну. Для системи моніторингу важливо розуміти, що значення має момент формування, а не лише момент надходження на серверний рівень.

STM32-вузол у запропонованій архітектурі виконує роль локального edge-вузла. Він може працювати з аналоговими входами, цифровими входами, інтерфейсами I2C або SPI, а також виконувати просту фільтрацію. Такі вузли доцільні там, де встановлення повноцінного ПЛК економічно або технічно недоцільне.

Шлюз Raspberry Pi використовується як проміжна ланка між різнорідними контролерами і серверним рівнем. Він приховує різницю між джерелами даних, приводить повідомлення до єдиної структури, додає метадані та забезпечує передавання через MQTT. Це спрощує серверний рівень, оскільки він працює з одним нормалізованим форматом.

Важливо, що протокол MQTT у цій архітектурі не є польовою шиною. Він використовується для передавання вже підготовленої телеметрії на

інформаційний рівень. Польовий обмін між ПЛК та модулями вводу-виводу залишається у зоні PROFINET або інших промислових протоколів.

Такий поділ дозволяє пояснити, чому в роботі не реалізується реальний стек PROFINET. Для дипломної задачі достатньо емулювати його роль у топології та показати, як дані після промислового рівня потрапляють у систему аналітики. Це відповідає темі інтелектуального моніторингу технологічних даних, а не розроблення драйвера польової мережі.

Наявність кількох контролерів у моделі важлива для демонстрації масштабованості. Якщо система може правильно розрізняти PLC-01, STM32-NODE-03 та інші вузли, вона може будувати топологію джерел, знаходити застарілі джерела даних і показувати оператору, де саме виникла проблема.

1.4 Аналіз існуючих рішень та постановка задачі

Існуючі рішення можна умовно поділити на SCADA-системи, IoT-платформи, MES-системи та спеціалізовані системи аналітики. SCADA добре підходить для диспетчерського керування і візуалізації, але не завжди має достатньо гнучкі інструменти машинного навчання. IoT-платформи спрощують збір даних, проте часто орієнтовані на хмарну інфраструктуру. MES-системи забезпечують виробничу простежуваність, але не замінюють оперативну технічну діагностику.

Для побудови програмного прототипу доцільно використовувати відкриті серверні, брокерні та аналітичні технології, зокрема FastAPI, MQTT-клієнт, SQLite, Python, Raspberry Pi, STM32, Mosquitto та scikit-learn [24-31].

У статті про інтелектуальну систему моніторингу біоматеріалу для біогазової установки наведено приклад поєднання IoT, хмарної обробки та машинного навчання. Важливим є висновок про користь адаптивного донавчання моделей на реальних експлуатаційних даних. У цій роботі

аналогічний принцип враховано через накопичення історії вимірювань і тренування моделей після досягнення мінімальної кількості даних.

Постановка задачі полягає у розробленні системи, яка приймає дані з різних джерел, зберігає їх з метаданими, виявляє аномалії, класифікує стан, прогнозує параметри, формує рекомендації та демонструє масштабування до понад 10 тисяч сенсорів у режимі навантажувальної симуляції.

1.5 Критерії якості інтелектуального моніторингу

Для оцінювання розроблюваної системи важливо визначити критерії, за якими можна зробити висновок про її придатність для технологічного моніторингу. Система повинна приймати не лише значення параметра, а й службову інформацію про джерело, час, протокол і якість вимірювання. Без таких метаданих неможливо відрізнити технологічне відхилення від проблеми каналу зв’язку або конкретного сенсора.

Для чистої кімнати затримка між подією на польовому рівні та появою інформації на панелі оператора має бути достатньо малою, щоб оператор міг вчасно відреагувати. У прототипі це забезпечується через підписник MQTT і WebSocket-трансляцію. REST-інтерфейс використовується для історії та аналітичних запитів, тоді як WebSocket відповідає за живі оновлення.

Простий сигнал «аварія» недостатній для дипломної задачі інтелектуального моніторингу. Система має показати, який параметр відхилився, від якого контролера він надійшов, яку оцінку аномальності отримано, який стан визначив класифікатор і яку дію доцільно запропонувати оператору.

Навіть якщо система застосовує ШІ, вона не повинна автоматично змінювати режими обладнання без підтвердження людини. Тому в роботі реалізовано підхід з участю оператора: рекомендація формується автоматично, але рішення приймає людина, а результат заноситься в журнал.

Для досягнення високої якості системи моніторингу в її прототипі реалізовано комплекс взаємопов'язаних критеріїв. Повнота даних забезпечується фіксацією метаданих controller_id, protocol та quality, що дає ефект повної простежуваності джерела. Висока оперативність і швидке оновлення панелі оператора досягаються завдяки використанню технологій MQTT та WebSocket. Критерій пояснюваності реалізовано через генерацію рекомендацій мовної моделі або локальних правил, що забезпечує зрозуміле для оператора обґрунтування рішень. Безпека системи гарантується обов'язковим підтвердженням дій оператором, завдяки чому повністю виключається пряме автономне керування. Зрештою, масштабованість архітектури та її спроможність працювати з великими обсягами даних перевірена шляхом успішної симуляції 10 тисяч сенсорів.

Аналітичний рівень такої системи може спиратися на методи прогнозування часових рядів, машинного навчання, статистичного контролю якості, оцінювання ризиків ШІ, WebSocket-оновлення, графічну візуалізацію та інтеграцію з мовними моделями [32-40].

2 ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ

2.1 Загальна архітектура системи

Розроблена система має модульну архітектуру. На нижньому рівні працює емулятор промислового шлюзу, який імітує PLC-01 та STM32-NODE-* контролери. PLC-01 генерує основні параметри HVAC/GMP, а STM32-вузли генерують локальні показники часток, HEPA ΔP, повітряного потоку і стану дверей шлюзу.

Серверний рівень реалізовано на FastAPI. Він приймає дані через підписник MQTT, зберігає їх у SQLite, виконує конвеєр ШІ, надає REST-інтерфейс для історії та аналітики, а також транслює поточні показники через WebSocket.

Архітектура орієнтована на локальне розгортання. Зовнішня велика мовна модель через OpenRouter використовується опційно; за відсутності ключа система повертає локальний висновок за правилами.

Архітектуру програмного прототипу інтелектуальної системи моніторингу наведено на рисунку 2.1.

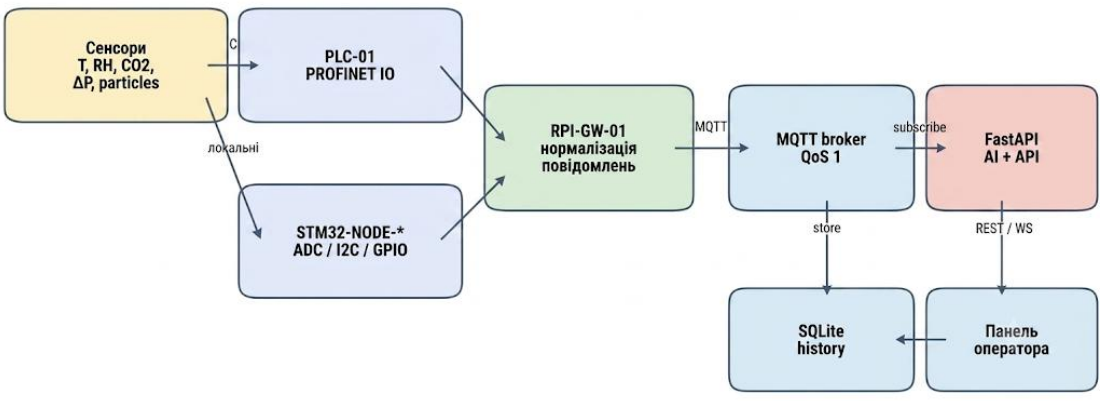


Рисунок 2.1 – Архітектура програмного прототипу інтелектуальної системи моніторингу

Архітектура побудована за принципом слабкого зв’язування компонентів. Емулятор контролерів не звертається безпосередньо до бази даних або моделей ШІ, а лише публікує повідомлення. Серверний рівень не залежить від фізичної реалізації сенсорів, оскільки працює з нормалізованою телеметрією. Панель оператора не звертається напряму до MQTT, а отримує дані через програмний інтерфейс і WebSocket.

Такий підхід спрощує заміну окремих компонентів. Наприклад, у демонстраційному режимі джерелом даних є емулятор, а в реальному впровадженні його можна замінити промисловим шлюзом або OPC UA-сервером. При цьому структура таблиць, REST-точки і логіка панелі оператора можуть залишитися незмінними.

Важливою архітектурною вимогою є локальність. Для дипломної демонстрації система повинна запускатися на одному комп’ютері, не залежати від хмарної бази даних і залишатися працездатною без зовнішньої мовної моделі. Саме тому SQLite, локальні правила і резервна модель прогнозування є виправданими рішеннями для прототипу.

Розподіл між історичними запитами і живими оновленнями також є принциповим. REST-точки зручні для отримання історії, статистики та топології, тоді як WebSocket краще підходить для оперативного оновлення показників. Це дозволяє не перевантажувати клієнт постійними HTTP-запитами.

Для промислової системи важливо, щоб додавання нового параметра не вимагало перероблення всієї архітектури. У запропонованій структурі новий sensor_type може бути доданий на рівні шлюзу, збережений у загальній таблиці вимірювань і показаний у панелі оператора. Лише для включення параметра в модель стану може знадобитися розширення ознакового вектора.

Архітектура також відокремлює аналітичний висновок від керування. Навіть якщо система виявляє критичний стан, вона не подає команду на обладнання напряму. Замість цього формується рекомендація, яка проходить через операторський контур і фіксується в журналі дій.

2.2 Модель даних і MQTT-повідомлення

Ключовою вимогою до системи є збереження не лише значення параметра, а й контексту його походження. Для цього MQTT-повідомлення розширено метаданими controller_id, controller_type, zone, sensor_id, protocol, quality, ts, sensor_type, value, unit, anomaly та anomaly_type.

Цей набір метаданих телеметрії виконує чітко визначені інженерні завдання. Ідентифікатори controller_id та sensor_id слугують для точного визначення конкретного ПЛК, STM32-вузла або самого сенсора, тоді як controller_type класифікує тип джерела, відокремлюючи сучасні контролери та шлюзи від старих видавців повідомлень. Просторову прив'язку забезпечує параметр zone, що вказує на виробничу зону або клас чистої кімнати, а технічний спосіб отримання даних фіксується через польовий протокол protocol (PROFINET, ADC, I2C, GPIO або MQTT). Стабільність та валідність вимірювань контролюються за допомогою прапорця якості quality, який може набувати значень GOOD, UNCERTAIN або BAD. Нарешті, для проведення верифікаційних тестів передбачено ознаку штучно інжектованої аномалії anomaly, деталізовану через anomaly_type, яка класифікує тип збою.

Система підтримує два формати MQTT-тем. Для зворотної сумісності залишено старий напрям, наприклад hvac/sensors/temperature або hvac/sensors/humidity. Для промислового сценарію рекомендовано тему із зазначенням зони та контролера: gmp/grade-c/PLC-01/temperature або gmp/grade-c/STM32-NODE-03/hepa_dp. За такою темою панель оператора може одразу визначити, з якого контролера прийшли дані, у якій зоні встановлено сенсор і який параметр вимірюється.

Для реалізації цих сценаріїв налаштовано роботу з двома типами топіків. Застарілий формат (Legacy) використовує пряму адресацію за типом сенсора в межах загальної системи кондиціювання, що демонструє приклад теми hvac/sensors/temperature. Натомість прогресивний Рекомендований формат будує

ієрархію послідовно через класифікацію зони, конкретного контролера та вимірюваної величини, що на практиці виглядає як gmp/grade-c/PLC-01/temperature.

База даних містить таблиці sensor_readings, system_states, llm_analyses, predictions, injected_anomalies, control_state та operator_actions. Окремі таблиці control_state і operator_actions потрібні для демонстрації безпечного операторського контуру, де ШІ формує рекомендації, але рішення фіксує людина.

ER-схему бази даних системи моніторингу наведено на рисунку 2.2.

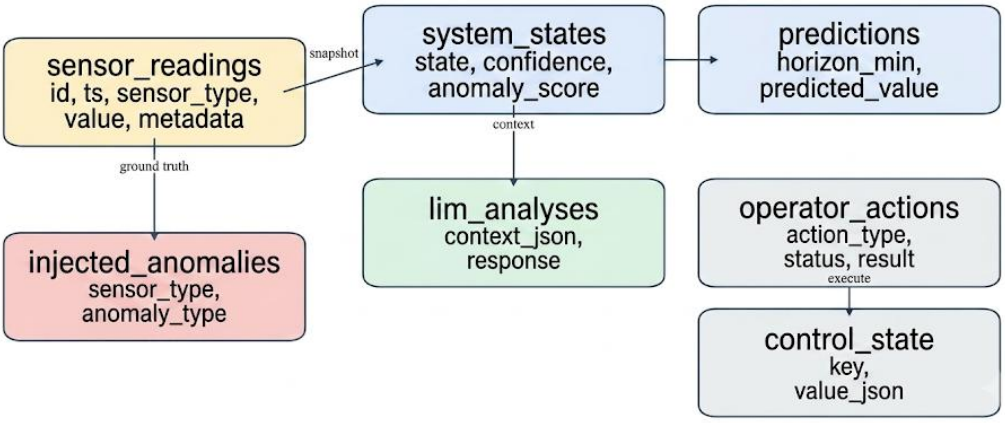


Рисунок 2.2 – ER-схема бази даних системи моніторингу

Метадані в повідомленні виконують роль технічного паспорта вимірювання. Якщо sensor_type показує, що саме виміряно, то controller_id і sensor_id показують, де саме це було виміряно. Поле protocol вказує на спосіб отримання даних на польовому рівні, а quality дозволяє відокремити надійні показники від сумнівних.

Поле zone потрібне для систем, де один серверний рівень може обслуговувати кілька приміщень або ділянок. У прототипі використано зону grade-c, але структура дозволяє додати grade-b, складську зону, технічне приміщення або окрему виробничу лінію без зміни таблиці вимірювань.

Наявність ts у повідомленні дозволяє зберігати час формування вимірювання. Це важливо, оскільки час доставки на серверний рівень може відрізнятися від часу реального вимірювання. У системах із кількома контролерами така різниця може впливати на правильність аналізу трендів і причинно-наслідкових зв’язків.

Старий формат повідомлення підтримується для демонстрації зворотної сумісності. У реальних системах модернізація рідко відбувається одномоментно: частина вузлів може вже передавати розширені метадані, а частина ще працювати за старою схемою. Серверний рівень не повинен відкидати такі дані, якщо їх можна коректно зберегти.

Використання єдиної таблиці sensor_readings спрощує історичний аналіз. У ній зберігаються як основні параметри моделі стану, так і додаткові технологічні показники. Це дозволяє розширювати систему без створення окремої таблиці під кожен новий тип сенсора.

Окремі таблиці для станів, прогнозів і висновків мовної моделі потрібні для аудиту. Оператор або керівник може повернутися до історії і побачити не лише значення параметра, а й те, який стан система визначила в той момент, який прогноз сформувала і яку рекомендацію видала.

2.3 Методи штучного інтелекту

Конвеєр ШІ складається з чотирьох основних блоків: виявлення аномалій, класифікація стану, короткострокове прогнозування та інтерпретація результатів. Виявлення аномалій виконується алгоритмом Isolation Forest. Його перевагою є можливість навчання на переважно нормальних даних без потреби у повністю розміченому наборі.

Ознаковий вектор включає температуру, вологість, CO2, тиск, потужність припливно-витяжної установки та часові ознаки. Результатом є anomaly_score, де від’ємні значення відповідають потенційно аномальному режиму. Така оцінка

зберігається разом із показником у sensor_readings і використовується для класифікації стану.

Класифікація стану виконується у два рівні. Перший рівень – детерміновані правила, які працюють одразу після запуску системи. Другий рівень – Random Forest, який може навчатися після накопичення достатньої кількості записів. Стан системи приймає значення Normal, Warning або Critical.

Прогнозування температури виконується на горизонті +30 хв. У системі передбачено декілька можливих моделей: Prophet, LSTM або резервна модель AR(1). Такий підхід дозволяє зберегти працездатність навіть тоді, коли важкі бібліотеки прогнозування не встановлені.

Аналітик на основі великої мовної моделі використовується для пояснення поточного стану українською мовою. Він отримує агрегований контекст: актуальні значення, середні за 1 годину, anomaly_score, стан системи та прогноз температури. Важливо, що мовна модель не керує обладнанням напряму, а лише формує рекомендацію оператору.

Для забезпечення такої функціональності конвеєр обробки даних використовує комплекс взаємопов'язаних методів аналітики. Спочатку за допомогою алгоритму Isolation Forest виконується виявлення аномалій, що дозволяє отримати бінарну мітку збою та кількісну оцінку anomaly_score. На наступному етапі відбувається класифікація поточного стану системи на категорії Normal, Warning або Critical, яка базується на поєднанні детермінованих правил та моделі Random Forest. Паралельно з цим математичні моделі Prophet, LSTM або авторегресія AR(1) відповідають за прогнозування динаміки процесу, видаючи очікуване значення температури з випередженням у 30 хвилин. Отримані результати передаються на рівень інтерпретації, де за допомогою сервісу OpenRouter або набору локальних правил генерується фінальна текстова рекомендація для персоналу. Завершує цей ланцюжок фіксація операторських дій, де на основі синергії правил ШІ та остаточного рішення людини формується детальний журнал прийнятих рішень.

Послідовність роботи конвеєра ШІ для обробки технологічних даних наведено на рисунку 2.3.

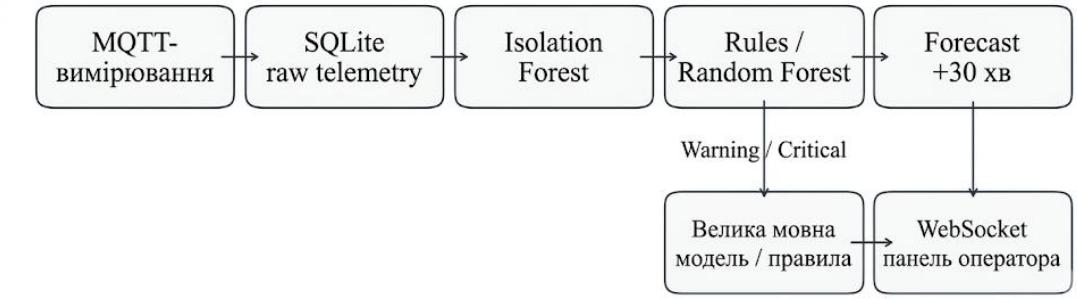


Рисунок 2.3 – Блок-схема конвеєра ШІ для обробки технологічних даних

Isolation Forest обрано як метод виявлення аномалій, оскільки він добре підходить для задач, де аномальні приклади складно зібрати заздалегідь. У технологічних системах більшість історичних даних відповідає нормальному режиму, а аварійні або небажані стани трапляються рідко. Це робить методи, що не потребують повної розмітки всіх класів, практично корисними.

Класифікація стану на рівні правил потрібна для надійного старту системи. Машинна модель потребує достатньої кількості даних, тоді як правила можуть працювати одразу після запуску. Наприклад, перевищення температури або падіння диференційного тиску можна інтерпретувати без навчання, якщо відомі допустимі межі.

Random Forest у прототипі використовується як модель, що може покращувати класифікацію після накопичення історії. Його перевагою є стійкість до шуму, можливість працювати з різними ознаками та відносна простота інтерпретації порівняно з глибокими нейронними мережами.

Прогнозування температури на +30 хв потрібне не для точного довгострокового передбачення, а для раннього попередження оператора. Якщо система бачить тенденцію до виходу за межі допустимого діапазону, оператор

може перевірити режим вентиляції або охолодження до того, як відхилення стане критичним.

Мовна модель у системі не приймає рішення самостійно. Вона отримує вже підготовлений контекст і формує текстове пояснення українською мовою. Такий підхід корисний для оператора, тому що числові показники, оцінка аномальності і стан системи перетворюються на стислий висновок з рекомендацією.

Резервні локальні правила є необхідними для надійності демонстрації. Якщо ключ OpenRouter відсутній або інтернет недоступний, система все одно повинна сформувати технічно коректний висновок. Це важливо і для захисту, і для промислового підходу, де зовнішній сервіс не повинен бути єдиною точкою відмови.

2.4 Контур рекомендацій та операторського керування

Для підсилення практичної цінності системи реалізовано контур операторського керування. Він демонструє не автоматичне втручання в обладнання, а контрольовану процедуру: рекомендація ШІ або правил, рішення оператора, симульований стан керування, журнал дій.

Система формує рекомендовані дії на основі поточних параметрів. Наприклад, при низькому каскаді тиску пропонується підняти pressure_setpoint_pa і fan_speed_percent; при високій температурі – знизити cooling_setpoint_c; при зростанні HEPA ΔP – запланувати перевірку фільтра; при відкритті шлюзу – перевірити airlock interlock.

Оператор може відхилити рекомендацію або виконати її як симуляцію. Виконання змінює лише таблицю control_state, тобто не має прямого впливу на реальне обладнання. Усі дії записуються в operator_actions із часом, типом дії, об’єктом, повідомленням, джерелом, статусом і результатом.

Такий підхід відповідає вимогам безпеки промислових систем: ШІ не є безпосереднім регулятором, а виступає аналітичним асистентом оператора. Це

особливо важливо для GMP-середовища, де будь-яка зміна режиму має бути простежуваною.

Контур прийняття операторських рішень із підтвердженням людиною наведено на рисунку 2.4.

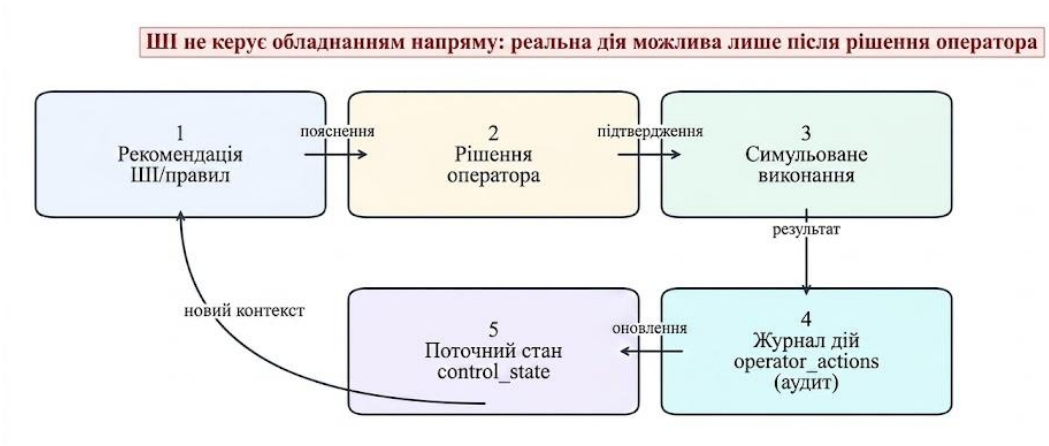


Рисунок 2.4 – Контур прийняття операторських рішень із підтвердженням людиною

Операторський контур у роботі побудований як безпечна симуляція. Це означає, що натискання кнопки застосування рекомендації не надсилає реальну команду на обладнання, а змінює лише симульований стан у таблиці control_state. Такий підхід дозволяє показати логіку керування без ризику для реального об’єкта.

Журнал operator_actions виконує роль аудиторського сліду. Для кожної дії зберігається час, тип дії, ціль, джерело, статус і результат. У GMP-середовищі така простежуваність принципова, оскільки зміни режимів повинні бути пояснюваними і відтворюваними.

Рекомендації формуються на основі поточного стану і технологічних правил. Наприклад, якщо тиск нижчий за допустимий рівень, система не повинна просто написати «аварія». Вона повинна запропонувати перевірити каскад, стан дверей шлюзу, швидкість вентилятора або уставку тиску.

Відхилення рекомендації також є важливою дією. Якщо оператор відхиляє пропозицію системи, цей факт може бути записаний у журнал. У майбутньому такі дані можуть використовуватися для оцінювання якості рекомендацій і донавчання моделей.

Контур із підтвердженням людиною дозволяє поєднати переваги автоматичного аналізу та відповідальність оператора. ШІ швидко обробляє дані й формує висновок, але остаточне рішення залишається за людиною, що відповідає вимогам безпечної експлуатації технологічних систем.

2.5 Функціональні та нефункціональні вимоги

Під час проєктування системи визначено функціональні та нефункціональні вимоги. Функціональні вимоги описують, що саме повинна виконувати система: приймати телеметрію, зберігати дані, аналізувати стан, відображати топологію, формувати рекомендації та підтримувати історію рішень. Нефункціональні вимоги описують якість реалізації: сумісність, масштабованість, надійність, локальність розгортання, простоту демонстрації та безпечну поведінку при відсутності зовнішнього інтерфейсу великої мовної моделі.

Система повинна бути сумісною з двома форматами MQTT-тем. Старий формат використовується лише для зворотної сумісності, наприклад hvac/sensors/temperature. Рекомендований формат використовується для промислового сценарію, наприклад gmp/grade-c/PLC-01/temperature, оскільки містить зону і контролер уже на рівні теми. Метадані всередині повідомлення дублюють цю інформацію для збереження в БД та відображення на панелі оператора.

Важливою нефункціональною вимогою є локальна працездатність. Серверний рівень, база даних, брокер MQTT, емулятор шлюзу і панель оператора можуть працювати на одному комп’ютері без хмарної інфраструктури. Це дає

змогу демонструвати систему під час захисту навіть у випадку нестабільного інтернет-з’єднання. Якщо ключ OpenRouter не заданий, точка ручного аналізу повертає висновок за локальними правилами, а не помилку користувацького сценарію.

Додавання нового sensor_type не повинно вимагати повної зміни архітектури. Новий параметр може бути доданий в емулятор шлюзу, збережений у sensor_readings і показаний на панелі оператора. Для включення його в конвеєр ШІ потрібне лише розширення набору ознак або окрема модель, але транспортний рівень і рівень зберігання залишаються без змін.

Така гнучкість архітектури забезпечує повну відповідність між вимогами до проєкту та їхньою практичною програмною реалізацією. Зворотна сумісність зі старими повідомленнями досягається здатністю серверного рівня приймати топіки формату hvac/sensors/# без додаткових метаданих, тоді як для надійної промислової ідентифікації джерела впроваджено обов'язкові параметри controller_id, sensor_id, protocol та quality. Живі оновлення панелі оператора реалізовані через виділений WebSocket-ендпоінт /ws/live. За аналітику поточного стану відповідає каскад із моделей Isolation Forest, Random Forest з детермінованими правилами та алгоритмів прогнозування, а доступне пояснення для оператора генерується через OpenRouter або локальні правила. Безпечне керування процесами гарантується архітектурним контуром із обов'язковим підтвердженням дій оператором та використанням симульованого стану control_state. Крім того, у системі передбачено повний аудит рішень завдяки логуванню в таблицю operator_actions, а потенціал до вертикального та горизонтального масштабування підтверджено спеціальним скриптом тестування.

2.6 Алгоритм обробки одного вимірювання

Алгоритм обробки одного вимірювання починається на шлюзовому рівні. Контролер формує початкове значення, додає час вимірювання та ознаку якості даних. Шлюз приводить повідомлення до уніфікованого JSON-формату і публікує його в брокер MQTT. Серверний рівень отримує повідомлення через підписник MQTT, виконує синтаксичну перевірку повідомлення і перетворює value до числового типу.

Після цього вимірювання зберігається в таблицю sensor_readings. Якщо повідомлення містить anomaly=true та anomaly_type, система додатково записує контрольну подію в injected_anomalies. Це потрібно для оцінювання якості виявлення аномалій у тестовому середовищі, де відомо, коли саме була штучно створена аномалія.

Конвеєр ШІ запускається лише для п’яти основних параметрів: temperature, humidity, co2, pressure і power. Це зроблено тому, що модель стану системи очікує повний вектор цих параметрів. Додаткові параметри, наприклад particles або hepa_dp, зберігаються і показуються на панелі оператора, але не порушують роботу класифікатора.

Коли в поточному зрізі даних є повний набір основних параметрів, система обчислює anomaly_score, класифікує стан, оновлює прогноз температури та за потреби запускає аналітик на основі великої мовної моделі. Результат повертається у WebSocket-повідомлення, завдяки чому панель оператора оновлюється без перезавантаження сторінки.

Послідовність обробки кожного вимірювання відбувається у межах чіткого наскрізного процесу. Усе починається з приймання MQTT-повідомлення з тем hvac/# або gmp/#, яке одразу передається на етап перевірки та нормалізації для виділення уніфікованих даних типу sensor_type, value, unit та відповідних метаданих. Наступним кроком є обов'язковий запис отриманої інформації у SQLite з формуванням нового рядка в таблиці sensor_readings. Після успішної фіксації в базі даних активується конвеєр ШІ, який розраховує показник аномальності anomaly_score, визначає поточний стан (state) та будує прогноз

(prediction). Залежно від отриманих результатів, система може опційно задіяти велику мовну модель для генерації зрозумілої текстової рекомендації оператору. Фінальним етапом обробки є трансляція сформованого пакета через WebSocket, що й забезпечує миттєве живе оновлення вебпанелі користувача.

2.7 Модель бази даних

База даних обрана SQLite, оскільки для дипломного прототипу важлива простота локального запуску. SQLite не потребує окремого сервера, підтримує транзакції, індекси і достатньо добре працює для демонстраційного навантаження. Для промислового впровадження рівень зберігання може бути замінений на PostgreSQL або базу часових рядів без зміни MQTT і ШІ-рівнів.

Таблиця sensor_readings є центральною. Вона містить timestamp, sensor_type, value, unit, is_anomaly, anomaly_score і метадані джерела. Індекси за timestamp, sensor_type і controller_id дають змогу швидко отримувати історію параметра, останні значення та підсумок топології.

Таблиця system_states зберігає результати класифікації стану системи. Таблиця predictions зберігає останні прогнози температури. Таблиця llm_analyses зберігає текстові висновки мовної моделі і контекст, на основі якого вони були сформовані. Це важливо для аудиту, оскільки оператор повинен мати змогу пояснити, чому система видала певну рекомендацію.

Таблиці control_state і operator_actions додані для операторського контуру. control_state містить поточні симульовані уставки припливно-витяжної установки та інші керовані значення. operator_actions є журналом рішень: у ньому фіксується дія, ціль, повідомлення, джерело, статус і результат. Така структура відповідає принципу простежуваності змін у GMP-середовищі.

Цей принцип простежуваності підтримується цілісною логічною моделлю бази даних, де кожна таблиця виконує свою специфічну роль у зберіганні інформації. Фундаментом моделі є таблиця sensor_readings, яка накопичує сиру

телеметрію, супутні метадані та ознаки виявлених аномалій. Результати аналізу цих даних розподіляються за напрямами: таблиця system_states зберігає поточні результати класифікації станів системи (Normal, Warning або Critical), а predictions фіксує короткострокові та довгострокові прогнози температури на заданий горизонт планування. Інтерпретація подій відображається у таблиці llm_analyses, яка містить агрегований контекст та відповіді аналітика на основі великої мовної моделі. Для проведення верифікації та навчання алгоритмів передбачено таблицю injected_anomalies, що виступає як еталонний набір даних (ground truth) для тестових аномалій. Нарешті, контур безпеки та взаємодії з користувачем замикається таблицями control_state, яка відображає симульовані уставки та поточні стани керування, та operator_actions, що діє як захищений журнал для повного аудиту операторських рішень.

2.8 Обробка часу, затримок і якості даних

Для технологічних систем важливо не лише отримати значення параметра, а й коректно інтерпретувати момент його формування. У запропонованій архітектурі часова мітка створюється на рівні джерела або шлюзу і передається у полі ts. Серверний рівень зберігає цю часову мітку без перезапису поточним серверним часом, якщо вона надійшла у повідомленні. Це дозволяє відрізнити реальний час вимірювання від часу доставки повідомлення.

Проблема затримок особливо важлива для систем із кількома контролерами. ПЛК може працювати з циклом опитування у десятки мілісекунд, STM32-вузли можуть мати власний період опитування, а шлюз публікує повідомлення батчами або послідовно. Тому в повідомленні передбачено поля джерела і протоколу, а точка топології показує last_seen для кожного контролера. Якщо контролер давно не передавав дані, панель оператора може позначити його як застаріле джерело.

Quality flag використовується для відображення довіри до вимірювання. У демонстраційному прототипі застосовано значення GOOD, UNCERTAIN і BAD. GOOD означає штатне вимірювання, UNCERTAIN може відповідати підозрі на шум або нестабільний канал, BAD – помилці сенсора або некоректному значенню. Навіть якщо значення з quality=BAD збережене в базі, оператор бачить його якість і може не використовувати його для прийняття рішення.

Додатково передбачено режим сумісності для старих повідомлень. Якщо повідомлення не містить метаданих, серверний рівень не відкидає його, а зберігає як старе вимірювання. Це важливо для поступової модернізації виробничої системи: старі вузли можуть працювати паралельно з новими шлюзами, а серверний рівень зберігає єдиний програмний інтерфейс для панелі оператора.

Для забезпечення високої надійності в умовах такої гетерогенної інфраструктури система реалізує гнучкі механізми обробки якості та часової достовірності даних. Наприклад, у ситуації, коли контролер з якихось причин не оновлював дані, система реагує присвоєнням статусу status=stale у топології мережі, завдяки чому оператор на панелі одразу бачить проблемне джерело. Якщо від сенсорів надходить сигнал із прапорцем BAD quality, вимірювання все одно фіксується у базі, але зберігається із відповідною ознакою якості, сигналізуючи оператору, що поточне значення не є надійним для аналітики. У разі отримання старого повідомлення без розширеної структури, система автоматично записує його з полем metadata=NULL та групує у категорію LEGACY-MQTT, забезпечуючи безперебійну сумісність із застарілими вузлами. Нарешті, під час виявлення штучно внесеного збою (Injected anomaly) дані заносяться до спеціальної таблиці injected_anomalies, що дає оператору та розробникам інструмент для точної оцінки ефективності роботи детектора аномалій.

У системах моніторингу з кількома джерелами важливо враховувати не лише значення параметра, а й порядок його надходження. Якщо дані від різних

контролерів приходять із різною затримкою, серверний рівень повинен зберігати їх із власними часовими мітками, а не припускати, що всі показники сформовані одночасно.

Для поточного стану системи використовується поняття актуального зрізу. Це набір останніх значень основних параметрів, які достатньо близькі за часом і можуть бути використані для класифікації. Якщо один із параметрів давно не оновлювався, система повинна або позначити стан як неповний, або повідомити оператору про застаріле джерело.

Якість даних також впливає на аналітику. Наприклад, показник з quality=BAD не слід використовувати як надійну підставу для технологічної рекомендації. Водночас його не потрібно повністю втрачати: збереження такого вимірювання допомагає діагностувати проблеми сенсора або каналу зв'язку.

У прототипі якість даних використовується насамперед для візуалізації і топології, але в реальному впровадженні вона може впливати на вагу ознак у моделі. Наприклад, алгоритм може ігнорувати BAD-значення або знижувати довіру до рекомендації, якщо частина вхідних параметрів має статус UNCERTAIN.

Окремою проблемою є пропуски даних. Якщо сенсор тимчасово не передає значення, система не повинна автоматично трактувати це як нормальний режим. Вона має відобразити відсутність оновлення, зберегти час останнього показника і дати оператору змогу побачити, який вузол потребує перевірки.

Підхід із метаданими дозволяє відрізняти технологічне відхилення від інформаційної проблеми. Наприклад, падіння тиску може бути реальною технологічною подією, але може бути і наслідком несправного датчика або втрати зв'язку з контролером. Без controller_id, protocol, quality і last_seen такі ситуації складно розрізнити.

Для чистої кімнати це особливо важливо, оскільки помилкова інтерпретація може спричинити неправильну реакцію. Якщо система прийме несправне вимірювання за реальне падіння тиску, оператор може почати

перевіряти вентиляцію, хоча фактична проблема знаходиться на рівні сенсора або кабельного підключення.

Таким чином, обробка часу, затримок і якості даних є не другорядною деталлю, а необхідною умовою достовірного інтелектуального моніторингу. Саме цей шар перетворює набір числових значень на контрольовану технологічну телеметрію.

Висновки до другого розділу. Спроєктовано архітектуру інтелектуальної системи моніторингу, визначено модель даних, методи конвеєра ШІ та безпечний контур операторських рекомендацій із журналюванням дій.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Структура програмного комплексу

Програмний комплекс реалізовано мовою Python з використанням FastAPI для серверного рівня, paho-mqtt для підписника MQTT, SQLite для локальної бази даних, scikit-learn для моделей машинного навчання та Chart.js для візуалізації на панелі оператора.

Цей набір технологій розподілений по чітко визначеній структурі каталогів програмного комплексу. Організація кодової бази починається з папки config/, яка відповідає за централізоване налаштування всієї системи за допомогою інструментів Pydantic Settings. Автономне тестування та генерацію потоків даних забезпечує модуль sensor_emulator/, що містить емулятори сенсорів, ПЛК, STM32-вузлів та промислового шлюзу. Ядром платформи виступає каталог backend/, де зосереджено застосунок FastAPI, підписник MQTT, рівень взаємодії з базою даних та маршрути програмного інтерфейсу. Обробка, прогнозування та інтелектуальний аналіз подій винесені в окрему директорію ai_pipeline/, яка об'єднує моделі Isolation Forest, Random Forest, модулі прогнозування та аналітик на основі великої мовної моделі. Візуальна частина представлена в dashboard/, де реалізовано україномовну вебпанель оператора, а допоміжні процеси, такі як навчання моделей, генерація звітів та симуляція роботи 10 тисяч сенсорів, виконуються через сценарії з папки scripts/. Нарешті, стабільність і надійність усього коду контролюється через каталог tests/, призначений для unit та API тестів.

Структура програмного комплексу побудована так, щоб розділити відповідальність між каталогами. Конфігурація винесена окремо, емулятори не змішуються з серверною логікою, модулі ШІ не залежать від HTML-інтерфейсу, а тести можуть перевіряти окремі частини системи незалежно одна від одної.

У директорії sensor_emulator зосереджено логіку формування телеметрії. Це дає змогу запускати систему без реального обладнання, але зберігати

промислову логіку джерел: ПЛК, STM32-вузли, шлюз, протоколи, ідентифікатори сенсорів і ознаки якості даних.

Компонент backend відповідає за приймання і зберігання даних, а також за надання програмного інтерфейсу. Саме тут зосереджено інтеграцію між MQTT, SQLite, WebSocket і аналітичними модулями. Такий підхід дозволяє розглядати серверний рівень як центральний вузол інформаційної системи.

Папка ai_pipeline містить аналітичну логіку. Винесення моделей і правил в окремий модуль спрощує тестування та подальше розширення. Наприклад, можна замінити модель прогнозування або додати новий алгоритм класифікації без переписування маршрутів FastAPI.

Розділ dashboard містить клієнтську частину. Він не виконує складну аналітику самостійно, а отримує вже підготовлені дані від серверного рівня. Це зменшує дублювання логіки і дозволяє підтримувати єдину інтерпретацію стану системи.

3.2 Серверний рівень, база даних і програмний інтерфейс

Серверний рівень запускається як застосунок FastAPI. Під час старту ініціалізується база даних SQLite, завантажуються моделі конвеєра ШІ, запускається підписник MQTT і монтується статична панель оператора. Для живих оновлень використовується WebSocket-точка /ws/live.

Основні REST-точки: /sensors/latest, /sensors/history, /analytics/summary, /analytics/topology, /analytics/forecast, /analytics/metrics, /analytics/ai/analyze, /analytics/recommendations, /analytics/actions та /analytics/control/state.

Особливістю реалізації є підтримка міграції SQLite без окремої системи Alembic. Якщо існуюча база даних була створена старішою версією програми, функція init_db додає відсутні nullable-колонки для метаданих контролера. Це забезпечує сумісність старого повідомлення з новим промисловим форматом.

Підписник MQTT приймає повідомлення зі старої теми hvac/sensors/# і нової теми gmp/#. Для кожного повідомлення визначається sensor_type, timestamp, value, unit, metadata та ознаки інжектованої аномалії. Після збереження початкового вимірювання запускається конвеєр ШІ для основних параметрів temperature, humidity, co2, pressure і power.

Для забезпечення взаємодії користувача з системою розроблено нові точки інтелектуального та операторського контуру, представлені відповідними програмними інтерфейсами. Зокрема, ендпоінт /analytics/topology відповідає за відображення топології мережі, показуючи активні джерела (ПЛК, STM32-вузли, шлюзи) та використовувані польові протоколи. Маршрут /analytics/ai/analyze надає можливість ручного запуску ШІ-аналізу безпосередньо оператором, а візуалізація сформованих системою порад забезпечується через /analytics/recommendations, що повертає список рекомендованих дій. Для повного аудиту процесів передбачено точку /analytics/actions, яка веде захищений журнал прийнятих оператором рішень. Нарешті, ендпоінт /analytics/control/state служить для моніторингу та відображення поточного симульованого стану припливно-витяжної установки, повітряного шлюзу та інших критичних вузлів автоматизації.

Під час приймання MQTT-повідомлення серверний рівень виконує декілька перевірок. Він визначає тип сенсора, перетворює значення до числового формату, перевіряє наявність метаданих і заповнює відсутні поля для старого формату. Це дозволяє уникнути аварійного завершення обробки через неповне повідомлення.

Збереження даних у SQLite виконується одразу після нормалізації. Такий порядок важливий, тому що навіть якщо аналітичний модуль тимчасово недоступний, початкове вимірювання не втрачається. Для систем моніторингу історія даних часто має не менше значення, ніж поточний висновок.

WebSocket-оновлення формується після обробки актуального вимірювання. Панель оператора отримує не лише саме значення, а й стан

системи, прогноз, ознаку аномальності та інформацію про джерело. Завдяки цьому інтерфейс може оновлюватися без перезавантаження сторінки.

Точка /analytics/topology потрібна для відображення промислового шару. Вона агрегує дані за controller_id, показує останній час активності, протоколи і кількість сенсорів. Саме ця точка демонструє, що система працює не з абстрактним набором датчиків, а з топологією контролерів.

Точки операторського контуру дають змогу відокремити рекомендації від виконаних дій. Список рекомендацій може змінюватися відповідно до поточного стану, тоді як журнал дій зберігає історію того, що реально підтвердив або відхилив оператор.

3.3 Панель оператора та промислова топологія

Панель оператора реалізовано як багатосторінковий вебінтерфейс з розділами: Огляд, Контролери, Датчики, Аналітика та Керування. Більшість інтерфейсу подано українською мовою, за винятком власних назв і технічних стандартів: PLC, STM32, Raspberry Pi, MQTT, PROFINET, FastAPI, OpenRouter.

Розділ Огляд відображає стан зони, впевненість ШІ, прогноз температури на +30 хв та кількість активних сенсорів. Розділ Контролери показує шлях даних, список протоколів і таблицю активних джерел. Розділ Датчики містить поточні показники з джерелом, протоколом і ознакою якості даних.

Розділ Аналітика містить графіки оцінки аномальності і прогнозу, а також кнопку ручного звернення до ШІ. Розділ Керування демонструє симульований стан припливно-витяжної установки, ручне застосування уставок, безпечний режим, список рекомендованих дій і журнал рішень.

Операторський інтерфейс побудовано не як маркетингову сторінку, а як робочий інструмент: таблиці, компактні панелі, статуси, графіки та дії розміщені так, щоб оператор бачив поточний стан і міг швидко перейти до джерела проблеми.

Візуалізація розробленого програмного комплексу представлена у вигляді набору екранних форм та графіків панелі оператора, що відображають архітектурні рівні, топологію джерел даних та результати роботи інтелектуального конвеєра (рисунки 3.1–3.7).

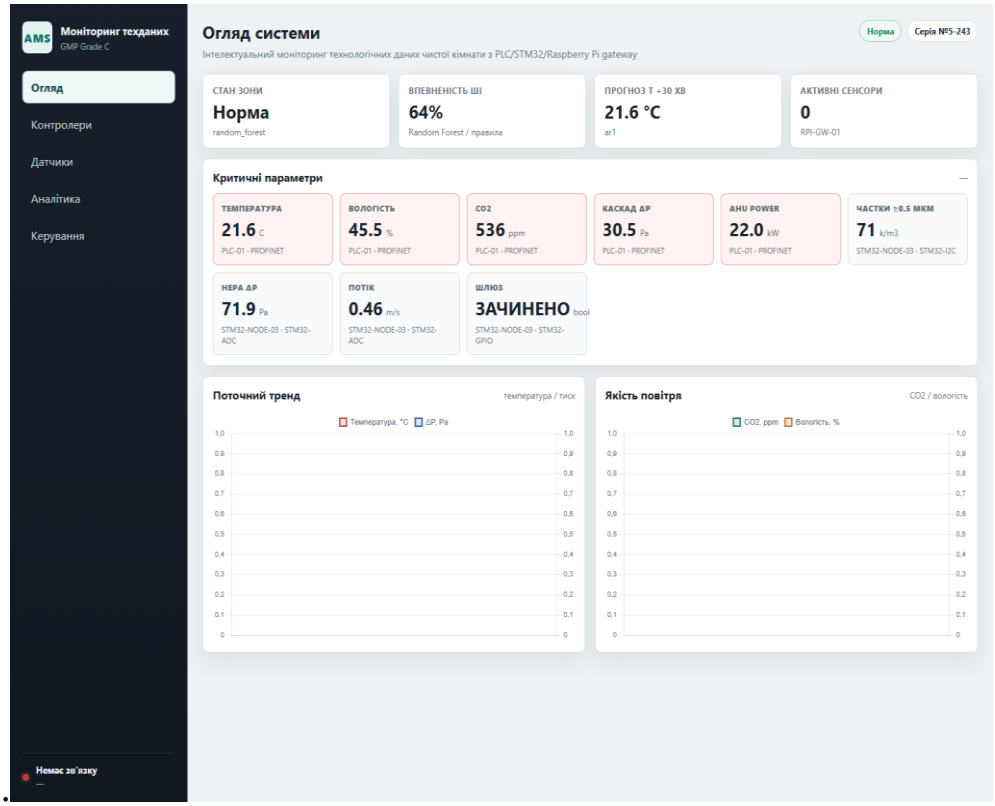


Рисунок 3.1 – Екранна форма панелі оператора у режимі огляду системи

Для детального моніторингу підключеного обладнання та аналізу мережевої структури передбачено спеціалізований інтерфейс. На рисунку 3.2 зображено екранну форму розділу контролерів і промислової топології, яка дозволяє відстежувати статус кожного вузла (ПЛК або STM32) та використовувані польові протоколи.

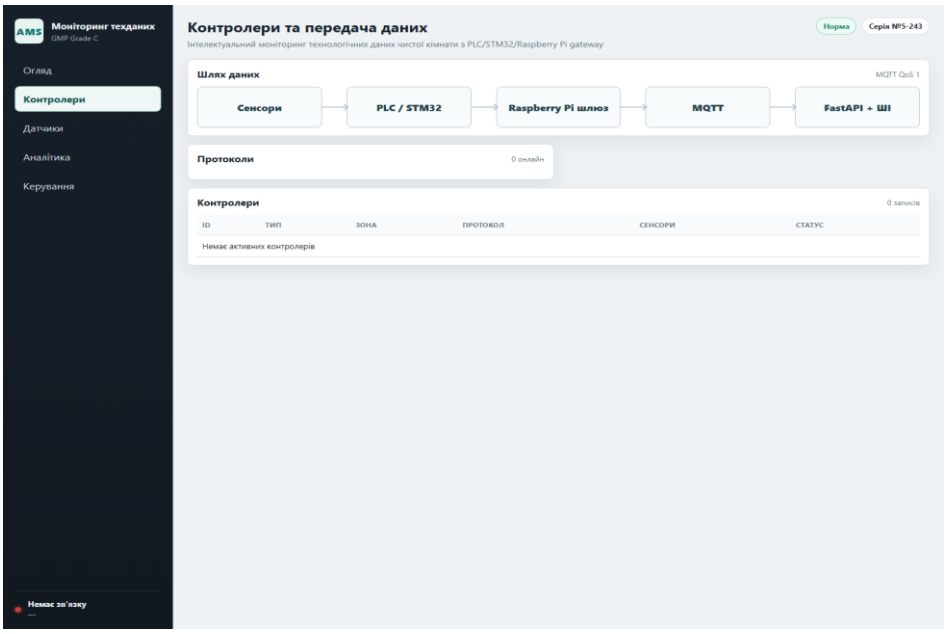


Рисунок 3.2 – Екранна форма розділу контролерів і промислової топології

Результати математичної обробки та інтерпретації технологічних параметрів виводяться в окрему робочу зону користувача. На рисунку 3.3 зображено екранну форму розділу ШІ-аналітики, де відобовжуються поточні оцінки стану та згенеровані мовною моделлю текстові рекомендації для персоналу.

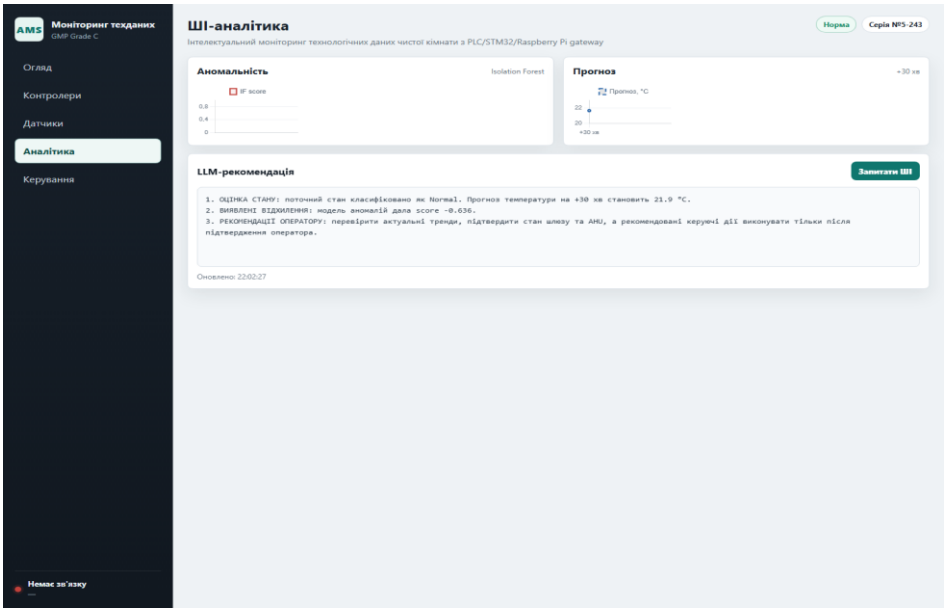


Рисунок 3.3 – Екранна форма розділу ШІ-аналітики

Безпечна взаємодія оператора з автоматизованою системою та фіксація його рішень реалізовані через інтерфейс зворотного зв'язку. На рисунку 3.4 зображено екранну форму розділу операторського керування, що містить симульовані уставки та захищений журнал аудиту дій.

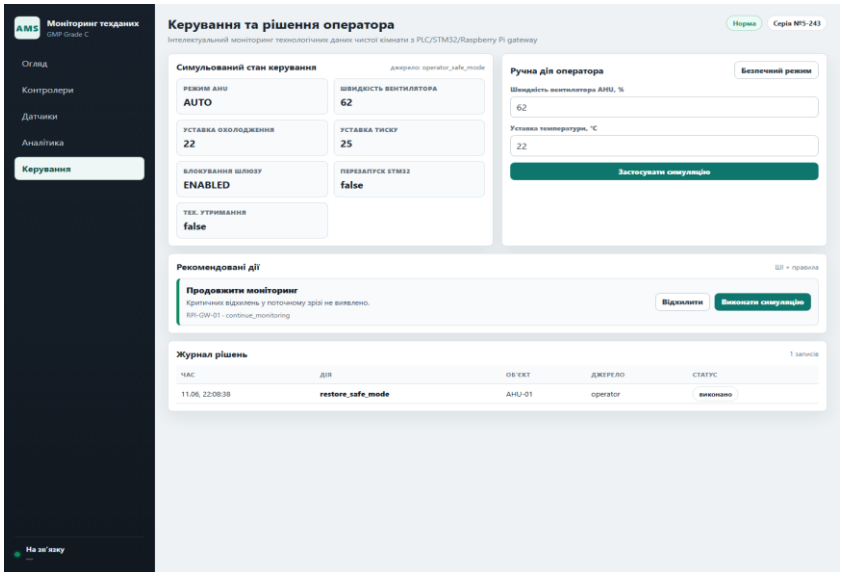


Рисунок 3.4 – Екранна форма розділу операторського керування

На рисунках 3.5–3.7 представлено графічні результати роботи аналітичного модуля системи моніторингу, що включають візуалізацію часових трендів, оцінку аномальності технологічного процесу та предиктивне моделювання.

Рисунок 3.5 – Графік тренду температури та каскаду диференційного тиску

Рисунок 3.6 – Графік оцінки аномальності Isolation Forest

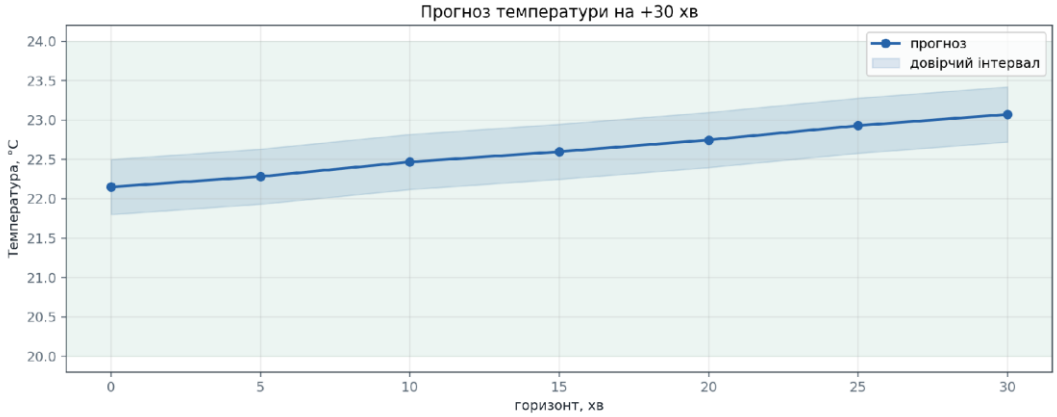


Рисунок 3.7 – Прогноз температури на горизонт +30 хв

Екранні форми панелі оператора мають демонструвати не лише значення параметрів, а й логіку роботи всієї системи. Тому розділ Огляд зосереджений на поточному стані, розділ Контролери – на промисловій топології, розділ Аналітика – на висновках ШІ, а розділ Керування – на прийнятті рішень.

Важливо, що інтерфейс показує джерело останнього вимірювання. Якщо оператор бачить температуру або перепад тиску, він одночасно бачить, від якого контролера або вузла надійшов цей показник. Це скорочує час пошуку причини проблеми.

Графік тренду потрібний для оцінювання динаміки. У технологічних процесах небезпечним може бути не лише перевищення межі, а й стійкий рух у

напрямку цієї межі. Тому відображення історії допомагає оператору діяти превентивно.

Графік оцінки аномальності показує результат роботи алгоритму Isolation Forest у часовому контексті. Якщо оцінка різко змінюється, оператор може зіставити це з показниками температури, тиску, CO2 або станом дверей шлюзу.

Прогноз температури на +30 хв використовується як індикатор майбутнього ризику. Він не замінює технологічний регламент, але допомагає оператору побачити, чи продовжується небажана тенденція і чи потрібна додаткова перевірка системи вентиляції або охолодження.

3.4 Емуляція контролерів і навантажувальне тестування

Для демонстрації промислового шару збору даних реалізовано sensor_emulator/industrial_gateway.py. Він створює PLC-01 і декілька STM32-NODE-* вузлів. PLC-01 публікує основні GMP/HVAC параметри з protocol=PROFINET, а STM32-вузли публікують локальні дані з protocol=ADC/I2C/GPIO.

Кожне повідомлення містить controller_id, controller_type, zone, sensor_id, protocol, quality, timestamp, sensor_type, value, unit і ознаки аномалій. Це дозволяє серверному рівню відобразити не тільки останнє значення параметра, а й останнє джерело показника, польовий протокол і стан якості даних.

Окремо реалізовано scripts/simulate_10k_sensors.py. Цей скрипт не навантажує панель оператора в реальному часі, а генерує батчі телеметрії для демонстрації масштабування понад 10 тисяч сенсорів. У результаті виводяться метрики: кількість сенсорів, кількість контролерів, повідомлень за секунду, затримка батча, кількість відкинутих та некоректних вимірювань.

Залежно від поставлених завдань тестування, система підтримує кілька спеціалізованих режимів емуляції. Режим промислового шлюзу (industrial_gateway) орієнтований на демонстрацію роботи протоколу MQTT у

реальному часі, генеруючи повноцінні потоки повідомлень для серверного рівня та миттєвого відображення на панелі оператора. Для перевірки зворотної сумісності передбачено режим старого видавця, який імітує поведінку застарілих вузлів шляхом надсилання спрощених топіків виду hvac/sensors/{sensor_type}. Нарешті, режим симуляції 10 тисяч сенсорів фокусується виключно на стрес-тестуванні та перевірці масштабованості архітектури, видаючи фінальний підсумок щодо загальної пропускної здатності платформи та затримок під час екстремальних навантажень.

Результати демонстрації масштабування системи понад 10 тисяч сенсорів наведено на рисунку 3.8.

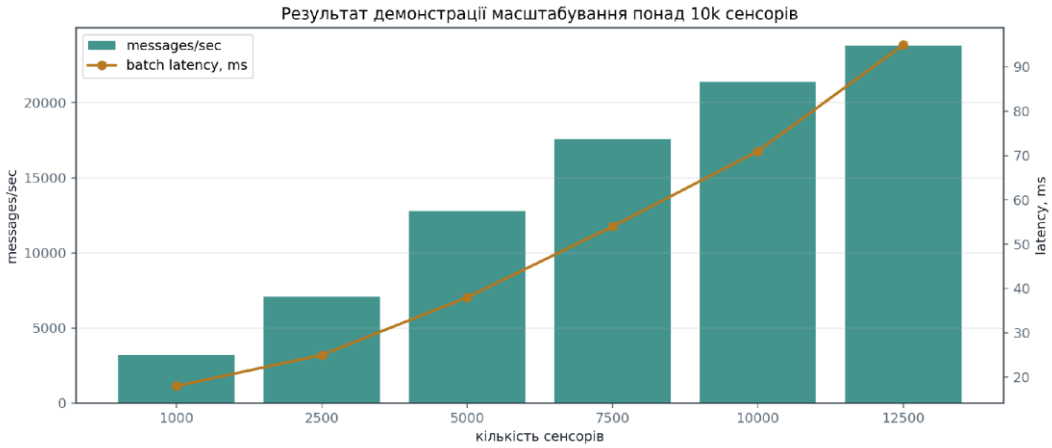


Рисунок 3.8 – Графік демонстрації масштабування понад 10 тисяч сенсорів

Навантажувальна симуляція відокремлена від основного демонстраційного режиму свідомо. Панель оператора призначена для зручного перегляду поточного стану, а не для одночасного виведення десяти тисяч карток сенсорів. Тому великий сценарій перевіряє транспортну і структурну частину архітектури, а не візуалізацію кожного окремого сенсора.

У режимі 10 тисяч сенсорів дані групуються за контролерами. Це ближче до реального промислового підходу, де велика кількість фізичних точок не підключається безпосередньо до серверного рівня, а проходить через

контролери, модулі вводу-виводу або шлюзи. Завдяки цьому система працює з організованою топологією, а не з хаотичним списком вимірювань.

Метрика повідомлень за секунду показує, наскільки швидко симулятор може сформувати і передати телеметрію. Вона не є абсолютною характеристикою майбутньої промислової системи, але дозволяє оцінити порядок навантаження і порівняти різні параметри запуску: кількість сенсорів, контролерів і розмір батча.

Затримка батча показує, скільки часу потрібно на формування і обробку групи повідомлень. Для технологічного моніторингу це важливо, тому що надто велика затримка може зробити дані менш корисними для оперативного реагування. У реальному впровадженні такі метрики допомагають підібрати частоту опитування і розмір черг.

Кількість некоректних або відкинутих вимірювань використовується як індикатор якості потоку даних. Якщо система починає отримувати багато некоректних значень, це може свідчити про помилку в конфігурації, перевантаження каналу або несправність окремого вузла.

Для дипломної роботи важливо, що масштабування показане окремим сценарієм, а не лише заявлене в тексті. Команда запуску симуляції дозволяє відтворити експеримент і побачити числові результати: кількість сенсорів, кількість контролерів, пропускну здатність і затримку.

Такий сценарій також підкреслює перевагу нормалізованого повідомлення. Якщо всі джерела передають дані в єдиній структурі, збільшення кількості сенсорів не потребує зміни серверної логіки. Змінюється лише кількість записів і навантаження на обробку.

У майбутньому навантажувальну симуляцію можна розширити чергами задач, окремими робочими процесами та серверною базою даних. Проте навіть у поточному вигляді вона демонструє, що архітектура не обмежена кількома навчальними датчиками і може бути пояснена як основа для масштабованого моніторингу.

Процес емуляції відіграє ключову роль у перевірці життєздатності розробленої архітектури оскільки дозволяє відтворити реальні умови експлуатації без необхідності залучення дороговартісного промислового обладнання. Програмний емулятор промислового шлюзу безперервно генерує потоки телеметрії які максимально наближені до сигналів справжніх датчиків температури вологості та тиску. Кожен віртуальний вузол формує власні пакети даних враховуючи специфіку обраного протоколу передачі та поточний стан імітованого середовища. Завдяки цьому вдається створити складну топологію мережі де одночасно функціонують віртуальні програмовані логічні контролери та мікроконтролерні плати. Система автоматично додає до кожного повідомлення унікальні ідентифікатори джерела позначки часу та маркери якості даних що дозволяє серверному рівню коректно сортувати та зберігати отриману інформацію.

Навантажувальне тестування виступає наступним логічним етапом перевірки під час якого система піддається екстремальним обсягам вхідних даних. Спеціально розроблений сценарій генерує тисячі одночасних підключень імітуючи роботу масштабного фармацевтичного комплексу. У процесі такого тестування ретельно фіксуються показники пропускної здатності затримки обробки пакетів та стабільності роботи бази даних. Отримані результати підтверджують здатність платформи ефективно масштабуватися та безперебійно функціонувати навіть в умовах пікових навантажень.

Гнучкість налаштувань емулятора дозволяє розробникам легко змінювати параметри симуляції задавати різні рівні шуму в показниках та моделювати нестандартні ситуації. Це створює ідеальне середовище для калібрування алгоритмів машинного навчання та перевірки надійності механізмів виявлення аномалій. Забезпечення високої якості моніторингу вимагає глибокого розуміння процесів які відбуваються на рівні збору та передачі інформації. Програмний комплекс генерує безперервний потік даних який повністю відповідає специфікаціям реальної апаратури. Завдяки використанню сучасних мов

програмування вдалося досягти високої продуктивності генератора який здатний формувати величезну кількість повідомлень щосекунди. Важливим аспектом є врахування мережевих затримок та можливих втрат пакетів що робить симуляцію ще більш наближеною до реальних умов експлуатації.

Під час таких випробувань штучно створюються пікові навантаження які перевіряють здатність обробляти масиви інформації без зниження швидкодії. Отримана статистика допомагає виявити вузькі місця в архітектурі та оптимізувати алгоритми запису. Процес навантажувального тестування також включає перевірку стабільності роботи сховища яке повинно швидко фіксувати нові значення та одночасно обслуговувати запити на читання від панелі оператора. Успішне проходження цих тестів гарантує що система зможе ефективно працювати на підприємстві де ціна помилки або затримки може бути надзвичайно високою. Крім того емулятор дозволяє перевірити коректність роботи механізмів відновлення зв'язку після його тимчасової втрати. Віртуальні контролери запрограмовані таким чином щоб автоматично намагатися відновити з'єднання з брокером повідомлень при виникненні проблем у мережі що забезпечує безперервність процесу моніторингу та мінімізує ризик втрати важливих технологічних даних.

3.5 Результати перевірки працездатності

Для перевірки системи створено модульні та API-тести. Модульні тести перевіряють генерацію валідних повідомлень емулятором, прийом старого MQTT-повідомлення без метаданих, збереження controller_id, sensor_id, protocol і quality, а також роботу контуру операторського керування.

API-тести перевіряють /sensors/latest, /sensors/history, /analytics/summary, /analytics/topology, /analytics/control/state і /analytics/recommendations. Регресійні тести підтверджують, що додаткові поля метаданих не порушують роботу виявлення аномалій, класифікації стану і прогнозування.

Під час контрольного запуску тестового набору отримано результат: 42 тести пройдено успішно. Це підтверджує працездатність основних компонентів системи після додавання промислових метаданих топології та операторського контуру.

Локальна демонстрація виконується запуском брокера MQTT, серверного рівня FastAPI і промислового шлюзу. Панель оператора доступна за адресою http://127.0.0.1:8000. Серверний рівень повертає HTTP 200 для головної сторінки та /analytics/summary, а точка /analytics/control/state повертає поточний симульований стан припливно-витяжної установки.

Для гарантування стабільності та коректності роботи всіх цих компонентів у системі розгорнуто комплексне тестове покриття, розділене за функціональними модулями. Валідація структури даних здійснюється у файлі test_controller_metadata.py, який перевіряє коректність MQTT-метаданих та правильність побудови мережевої топології. Модуль test_industrial_gateway.py фокусується на процесах імітації обладнання, тестуючи алгоритми генерації повідомлень для ПЛК, STM32-вузлів та промислового шлюзу. Контур взаємодії з користувачем та безпеки покривається тестом test_operator_control.py, який перевіряє механізми формування рекомендацій, логування дій у журнал та роботу симульованого керування устаткуванням. Доступність та коректність відповідей серверної частини контролюється через test_api.py, де тестуються всі ключові REST-ендпоінти. Нарешті, аналітичний блок перевіряється двома виділеними модулями: test_anomaly_detector.py оцінює точність і якість поведінки алгоритму Isolation Forest, а test_state_classifier.py верифікує роботу детермінованих правил та класифікатора Random Forest.

Модульні тести потрібні для перевірки локальної логіки без запуску всієї системи. Наприклад, генерація повідомлення промислового шлюзу може бути перевірена окремо від MQTT-брокера, а класифікатор стану – окремо від панелі оператора.

API-тести перевіряють поведінку серверного рівня з погляду зовнішнього користувача. Якщо точка /sensors/latest повертає актуальні показники, а /analytics/topology повертає активні контролери, це підтверджує, що база даних, маршрути і формат відповіді узгоджені між собою.

Регресійні тести важливі після додавання промислових метаданих. Раніше система могла працювати лише з sensor_type, value і timestamp, а після розширення повідомлення з'явилися controller_id, sensor_id, protocol і quality. Тести підтверджують, що старий формат не зламався.

Окрема перевірка injected_anomalies потрібна для оцінювання якості детектора аномалій. Якщо емулятор штучно створює відхилення, система повинна записати факт такої події, щоб потім можна було порівняти його з результатами алгоритму.

Тестування операторського контуру перевіряє, що рекомендація може бути отримана, дія може бути виконана як симуляція, а результат потрапляє в журнал. Це підтверджує не лише аналітичну, а й управлінську частину прототипу.

Аналіз результатів перевірки працездатності системи підтверджує правильність обраних архітектурних рішень та ефективність застосованих алгоритмів. Усі компоненти програмного комплексу продемонстрували високий рівень інтеграції та здатність безперебійно обмінюватися даними у режимі реального часу. Серверна частина успішно приймає нормалізує та зберігає вхідні повідомлення забезпечуючи швидкий доступ до історії вимірювань.

Робота конвеєра штучного інтелекту також показала відмінні результати у виявленні нестандартних ситуацій та класифікації станів системи. Алгоритми машинного навчання здатні швидко адаптуватися до змін у потоці даних та своєчасно сигналізувати про потенційні загрози для технологічного процесу. Панель оператора миттєво відображає оновлену інформацію дозволяючи персоналу контролювати ситуацію без жодних затримок. Особливої уваги заслуговує стабільність роботи модуля прогнозування який з високою точністю

передбачає зміни температури на заданий горизонт часу. Це дає можливість операторам приймати превентивні заходи та уникати критичних відхилень у параметрах мікроклімату чистої кімнати.

Результати багаторазових запусків тестових сценаріїв підтверджують відсутність витоків пам'яті та інших критичних помилок у програмному коді. База даних ефективно справляється з постійним потоком нових записів не виявляючи ознак деградації продуктивності навіть після тривалого часу безперервної роботи. Контур безпечного операторського підтвердження також пройшов повну перевірку довівши неможливість несанкціонованого втручання у технологічний процес. Кожна дія оператора надійно фіксується у захищеному журналі що забезпечує повну простежуваність та відповідає суворим вимогам стандартів якості.

Отримані результати дають всі підстави стверджувати що розроблений прототип повністю готовий до подальшого масштабування та впровадження у виробниче середовище. Окремо варто відзначити високу якість візуалізації даних на панелі яка дозволяє персоналу швидко оцінювати поточний стан системи. Графіки трендів плавно оновлюються відображаючи найменші коливання технологічних параметрів. Користувацький інтерфейс інтуїтивно зрозумілий та не вимагає тривалого навчання для початку експлуатації. Усі ці фактори в сукупності забезпечують високу ефективність процесу моніторингу та дозволяють мінімізувати вплив людського фактора на прийняття управлінських рішень покращуючи загальну надійність автоматизації.

3.6 Порядок тестування роботи системи

Для захисту роботи передбачено декілька тестових сценаріїв.. Запускається брокер MQTT, серверний рівень і промисловий шлюз. Панель оператора показує активний RPI-GW-01, PLC-01, STM32-NODE-* та поточні значення параметрів. Стан системи класифікується як Normal або Warning залежно від згенерованих даних.

Далі – відхилення технологічного параметра. Емулятор може сформувати інжектовану аномалію або природне відхилення, наприклад підвищення температури, CO2 чи падіння каскаду тиску. Серверний рівень зберігає вимірювання, конвеєр ШІ обчислює anomaly_score, панель оператора підсвічує відповідний параметр і показує джерело: PLC-01 або STM32-NODE-*.

Оператор переходить у розділ Аналітика і натискає кнопку «Запитати ШІ». Серверний рівень формує контекст аналізу, запускає мовну модель або локальні правила і повертає структурований висновок українською мовою: оцінка стану, виявлені відхилення та рекомендації оператору.

У розділі Керування оператор бачить рекомендовані дії. Він може відхилити рекомендацію або виконати симуляцію. У разі виконання оновлюється control_state, наприклад fan_speed_percent або pressure_setpoint_pa, а в operator_actions створюється запис зі статусом executed.

Запускається scripts/simulate_10k_sensors.py із параметрами --sensors 10000 --controllers 64 --batch-size 500. Скрипт генерує батчі телеметрії від багатьох контролерів і виводить показники пропускної здатності, затримки та кількість некоректних або відкинутих вимірювань. Цей сценарій демонструє, що архітектура може бути масштабована за межі невеликої демонстрації панелі оператора.

Для всебічної перевірки розробленого програмного комплексу передбачено серію базових демонстраційних сценаріїв, кожен з яких фокусується на конкретному аспекті роботи платформи. У межах нормального

режиму демонструється стабільне надходження телеметрії у реальному часі та автоматична побудова мережевої топології, де ключовим результатом є відображення активних контролерів і сенсорів. Сценарій моделювання аномалії призначений для перевірки конвеєра ШІ та візуального підсвічування потенційних ризиків на основі розрахованих міток is_anomaly та індексу anomaly_score. Режим ручного ШІ-аналізу дозволяє оцінити якість експертної діагностики за прямим запитом оператора, видаючи розгорнуту текстову рекомендацію українською мовою. Безпека та простежуваність дій відпрацьовуються у сценарії операторського рішення, який ілюструє обов'язкове підтвердження будь-якої керівної дії людиною з подальшим внесенням відповідного запису до захищеного журналу. Замикає цей ланцюжок сценарій симуляції 10 тисяч сенсорів, орієнтований на стрес-тестування та отримання підсумкових метрик пропускної здатності системи за екстремальних інформаційних навантажень.

Процедура тестування охоплює всі ключові аспекти функціонування розробленого програмного комплексу починаючи від збору сирих даних і закінчуючи формуванням аналітичних звітів. Спочатку перевіряється здатність системи автоматично розпізнавати нові джерела та коректно відображати їх у загальній топології мережі. Далі ініціюється процес генерації нормального технологічного фону під час якого всі параметри знаходяться в межах допустимих норм. На цьому етапі оцінюється стабільність роботи графічних інтерфейсів та коректність запису інформації у локальну базу даних.

Після успішного проходження базового тесту здійснюється перехід до перевірки реакції програми на виникнення аномальних ситуацій. Для цього штучно вводяться критичні відхилення у показники температури або тиску імітуючи поломку обладнання або порушення герметичності чистої кімнати. Алгоритм повинен миттєво зафіксувати ці зміни підвищити рівень тривоги та сформувати відповідне попередження на панелі оператора. Одночасно запускається механізм генерації рекомендацій який аналізує поточний стан та

пропонує оптимальні шляхи вирішення проблеми. Людина уважно вивчає запропоновані кроки та приймає рішення про їх виконання або відхилення. Важливо переконатися що всі ці дії правильно реєструються у журналі подій створюючи надійний аудиторський слід.

Окремим етапом тестування є перевірка роботи модуля прогнозування який адаптує свої розрахунки відповідно до нових умов середовища. Успішне завершення всіх передбачених кроків підтверджує високу надійність архітектури та її готовність до роботи в умовах реального виробництва. Комплексний підхід до тестування гарантує виявлення більшості потенційних проблем ще на етапі розробки прототипу що суттєво знижує витрати на їх усунення у майбутньому.

Ретельна перевірка кожного модуля ізольовано та у взаємодії з іншими компонентами створює міцний фундамент для побудови безвідмовної промислової платформи. Результати проведених перевірок ретельно документуються та слугують базою для подальшого вдосконалення програмного забезпечення. Особлива увага приділяється правильності обробки мережевих затримок які можуть виникати при передачі показників від віддалених датчиків до центрального сервера. Платформа вміє правильно синхронізувати інформацію опираючись на часові мітки створені безпосередньо у момент фізичного вимірювання.

3.7 Розгортання на Windows

Оскільки розроблення і демонстрація виконуються на Windows, у README наведено зручну для Windows інструкцію запуску. Рекомендований порядок: створити віртуальне середовище Python, встановити requirements.txt, скопіювати .env.example у .env, запустити брокер MQTT, серверний рівень і промисловий шлюз.

Для брокера MQTT рекомендовано Mosquitto, але під час локальної демонстрації можливе використання amqtt, якщо Mosquitto не встановлений у

системі. Це не змінює логіку проєкту, оскільки серверний рівень взаємодіє з брокером через стандартний протокол MQTT.

Серверний рівень запускається командою python -m uvicorn backend.main:app --host 127.0.0.1 --port 8000. Після запуску панель оператора доступна у браузері за адресою http://127.0.0.1:8000. Промисловий шлюз запускається окремим процесом командою python -m sensor_emulator.industrial_gateway.

Файл .env.example не містить реального ключа програмного інтерфейсу. Для OpenRouter використовується заповнювач your_openrouter_api_key_here. Якщо ключ не заданий, система не падає, а застосовує локальні правила для ШІ-аналізу. Це важливо для повторюваності демонстрації на іншому комп'ютері.

Для розгортання та тестування системи у локальному середовищі передбачено чіткий набір основних команд запуску. Ініціалізація серверного рівня здійснюється за допомогою вебсервера Uvicorn шляхом виконання команди python -m uvicorn backend.main:app --host 127.0.0.1 --port 8000. Одночасно з цим запуск промислового шлюзу для генерації стандартних потоків телеметрії виконується через модуль python -m sensor_emulator.industrial_gateway. Якщо необхідно провести стрес-тестування платформи під високим навантаженням, застосовується спеціалізований сценарій симуляції 10 тисяч сенсорів, який викликається командою python scripts\simulate_10k_sensors.py --sensors 10000 --controllers 64 --batch-size 500. Нарешті, верифікація працездатності всього набору модулів та перевірка кодової бази реалізується запуском тестового фреймворку за допомогою команди python -m pytest.

Процес розгортання програмного комплексу на операційній системі вимагає чіткого дотримання послідовності дій для забезпечення безперебійної роботи всіх компонентів. Спочатку необхідно підготувати ізольоване середовище яке дозволить уникнути конфліктів між залежностями різних проектів. Після створення такого середовища виконується завантаження та

встановлення всіх необхідних бібліотек пакетів та фреймворків які забезпечують функціонування серверної частини алгоритмів машинного навчання та інтерфейсу користувача.

Наступним кроком є налаштування конфігураційних файлів де вказуються параметри підключення до сховища даних адреси брокера повідомлень та ключі доступу до зовнішніх аналітичних сервісів. Дуже важливо правильно вказати всі шляхи та порти щоб компоненти системи могли безперешкодно обмінюватися даними між собою. Після завершення базових налаштувань ініціюється запуск служби яка відповідає за обробку вхідних повідомлень та маршрутизацію трафіку. Паралельно стартує процес емуляції промислового обладнання який починає наповнювати систему тестовими даними.

Серверна частина постійно прослуховує визначені канали зв'язку перехоплюючи інформацію та направляючи її до конвеєра штучного інтелекту. Графічний інтерфейс завантажується у браузері користувача автоматично встановлюючи постійне з'єднання з сервером для отримання оновлень у режимі реального часу. Весь процес запуску супроводжується виведенням детальної інформації у консоль що дозволяє розробнику контролювати кожен етап та оперативно реагувати на можливі помилки.

Правильно організоване розгортання гарантує стабільну роботу системи навіть при тривалому використанні. Ретельна підготовка робочого середовища є запорукою відсутності проблем сумісності програмного забезпечення у майбутньому. Важливо також забезпечити наявність достатнього обсягу дискового простору для збереження історичних записів які постійно накопичуються у процесі роботи платформи моніторингу. Налаштування автоматичного запуску всіх служб після перезавантаження операційної системи суттєво спрощує експлуатацію комплексу та зменшує час простою обладнання. Крім того регулярне оновлення встановлених бібліотек дозволяє підтримувати високий рівень безпеки та отримувати доступ до нових функціональних можливостей. Детально описаний процес розгортання робить проект зрозумілим

для широкого кола спеціалістів дозволяючи їм швидко налаштувати власну копію програми для детального вивчення або подальшої модифікації.

3.8 Безпека, обмеження та напрями розвитку

Розроблена система є демонстраційним прототипом. Вона не повинна підключатися до реального обладнання без додаткових рівнів кібербезпеки, автентифікації, авторизації, журналювання доступу, валідації команд і промислової сертифікації. У межах дипломної роботи керування реалізовано лише як симуляцію, що є правильним рішенням з погляду безпеки.

Потенційним обмеженням є використання SQLite для зберігання даних. Для дипломного прототипу це достатньо, але для реального підприємства бажано використовувати PostgreSQL, TimescaleDB або InfluxDB. Також доцільно відокремити підписник MQTT, обробники ШІ і сервіс програмного інтерфейсу в окремі процеси або контейнери.

Для реального впровадження необхідно додати автентифікацію користувачів, ролі оператора, інженера і адміністратора, цифровий підпис операторських рішень, аудиторський слід, резервне копіювання бази даних і політику зберігання історичних показників.

Окремим напрямом розвитку є інтеграція з реальними ПЛК через OPC UA або промисловий шлюз, який вже підтримує PROFINET/Modbus TCP на польовому рівні. У такому випадку розроблена система може залишитися серверним, ШІ- та операторським рівнем, а джерелом даних стане реальний промисловий шлюз.

Після підтвердження оператором правильності або хибності рекомендації система може накопичувати валідаційні дані та використовувати їх для покращення моделей класифікації або ранжування рекомендованих дій.

Аналіз поточної архітектури дозволяє виділити чіткі технічні обмеження та напрями розвитку розробленого прототипу. По-перше, використання SQLite

для збереження даних у промисловому впровадженні доцільно замінити на потужніші СКБД на кшталт PostgreSQL або її тайм-серійного розширення TimescaleDB. По-друге, симульовану роботу протоколу PROFINET у майбутньому варто перевести на реальний фізичний шлюз або виконати повноцінну інтеграцію через стандарт OPC UA. По-третє, поточний відкритий програмний інтерфейс, що функціонує без автентифікації, вимагає впровадження сучасних протоколів безпеки JWT чи OAuth2 із розмежуванням ролей користувачів. Четвертим напрямом є еволюція локальних правил рекомендацій, які можна покращити шляхом безпосереднього машинного навчання на історично підтверджених оператором діях. Нарешті, архітектуру з одним процесом серверного рівня для підвищення відмовостійкості та продуктивності слід масштабувати через впровадження черг задач, додаткових робочих процесів та загальної контейнеризації компонентів системи.

Обмеження прототипу не зменшують його демонстраційної цінності, але їх потрібно явно зазначити. Система показує архітектурний і програмний механізм моніторингу, однак для реального виробництва необхідні додаткові процедури валідації, кібербезпеки та інтеграції з наявною автоматикою.

Для промислового впровадження SQLite доцільно замінити на серверну базу даних. Це дозволить працювати з більшими обсягами історії, розмежовувати доступ користувачів, виконувати резервне копіювання і підключати зовнішні засоби аналітики.

Окремим напрямом є автентифікація та авторизація. У дипломному прототипі інтерфейс відкритий для спрощення демонстрації, але на реальному підприємстві повинні бути ролі оператора, інженера, адміністратора і переглядача, а також журналювання входів і змін.

Для реального підключення до ПЛК доцільно використовувати промисловий шлюз або OPC UA. У такій схемі розроблена система залишиться рівнем моніторингу та аналітики, а низькорівнева взаємодія з обладнанням буде виконуватися засобами, придатними для промислової експлуатації.

Подальший розвиток ШІ-компонента може включати накопичення підтверджених оператором дій. Якщо оператор регулярно приймає або відхиляє певні рекомендації, ці дані можуть бути використані для покращення правил, навчання моделі ранжування або формування персоналізованих підказок.

Забезпечення високого рівня безпеки є критично важливим завданням при проектуванні сучасних систем моніторингу. Незважаючи на те що розроблений комплекс є демонстраційним прототипом у ньому закладені базові принципи захисту інформації які можуть бути масштабовані у майбутньому. Використання ізольованих контурів управління гарантує що жодна команда не може бути відправлена на виконавчі механізми без явного підтвердження з боку уповноваженого персоналу. Кожна дія оператора ретельно документується створюючи детальну історію взаємодії з платформою яка використовується для розслідування інцидентів або оптимізації робочих процесів.

Одним з основних обмежень поточної версії є відсутність повноцінної системи розмежування прав доступу що вимагає впровадження надійних механізмів автентифікації користувачів перед початком роботи з реальними промисловими даними. Також існує потреба у переході на більш потужні рішення для зберігання інформації які здатні працювати з колосальними обсягами історичної телеметрії та забезпечувати високу швидкість виконання складних аналітичних запитів.

Напрями подальшого розвитку проекту включають розширення можливостей конвеєра штучного інтелекту шляхом впровадження алгоритмів глибокого навчання здатних самостійно знаходити приховані закономірності у поведінці складного технологічного обладнання. Важливим кроком стане створення мобільного застосунку який дозволить інженерам та керівникам отримувати оперативні сповіщення про стан чистої кімнати знаходячись за межами робочого місця. Розроблений прототип має величезний потенціал для модернізації та адаптації під специфічні потреби будь якого фармацевтичного підприємства.

Поступове впровадження нових модулів безпеки шифрування каналів зв'язку та резервного копіювання даних перетворить цю дослідницьку роботу на повноцінний комерційний продукт готовий до суворої сертифікації. Безперервний процес вдосконалення архітектури гарантує актуальність розробленого програмного забезпечення протягом багатьох років дозволяючи компаніям підтримувати найвищі стандарти якості виготовлення медичних препаратів та мінімізувати виробничі ризики пов'язані з людським фактором або технічними несправностями.

ВИСНОВКИ

У кваліфікаційній роботі розроблено інтелектуальну систему моніторингу технологічних даних на прикладі чистої кімнати GMP класу C. Запропонований приклад дозволяє показати не лише збір даних із датчиків, а повний промисловий шлях телеметрії від контролерів до ШІ-аналітики та панелі оператора.

Проаналізовано предметну область інтелектуального моніторингу технологічного обладнання, включно з підходами SCADA, IoT, MES та машинного навчання. На основі аналізу обґрунтовано потребу у багаторівневій архітектурі з ПЛК, STM32, шлюзом Raspberry Pi, MQTT і серверним рівнем.

Розроблено структуру передачі даних від сенсорів через ПЛК/STM32 та шлюз до серверного рівня FastAPI. Уніфіковане MQTT-повідомлення містить controller_id, controller_type, zone, sensor_id, protocol, quality, timestamp, sensor_type, value, unit та ознаки аномалій.

Реалізовано конвеєр ШІ, який включає виявлення аномалій Isolation Forest, класифікацію стану правилами та Random Forest, короткострокове прогнозування температури на +30 хв та формування текстових рекомендацій через мовну модель або локальні правила.

Створено україномовну вебпанель оператора з розділами огляду, контролерів, датчиків, аналітики та керування. Додано ручне звернення до ШІ, рекомендовані дії, симульоване керування припливно-витяжною установкою і журнал рішень оператора.

Реалізовано режим демонстрації масштабу понад 10 тисяч сенсорів, що дозволяє оцінити пропускну здатність, затримку батча і кількість некоректних або відкинутих вимірювань без перевантаження основної панелі оператора.

Працездатність системи підтверджено тестами: 42 автоматизовані тести пройдено успішно. Отриманий прототип може бути використаний як основа для подальшого розширення, інтеграції з реальними контролерами та підключенням до промислової SCADA/MES-інфраструктури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Пальчевський Б. О. Інтелектуальний моніторинг параметрів технологічного обладнання. Наукові нотатки. 2022. №73. DOI: 10.36910/775.24153966.2022.73.33. URL: https://eforum.lntu.edu.ua/index.php/naukovi_notatky/en/article/view/818

2. Лагойда А. І., Лагойда Л. І., Чигур І. І., Чигур Л. Я. Інтелектуальна система збору та обробки даних на базі мікроконтролера з інтеграцією у SCADA. Методи та прилади контролю якості. 2025. №2(55). С. 140–147. DOI: 10.31471/1993-9981-2025-2(55)-140-147. URL: https://mpky.nung.edu.ua/index.php/mpky/article/view/679

3. Павлов С. Г., Лисенко В. П., Лендєл Т. І., Наконечна К. В. Інтелектуальна система автоматизованого моніторингу якості біоматеріалу для зброджування в біогазовій установці. Bulletin of NTUU KPI. Chemical Engineering, Ecology and Resource Saving. 2025. №3(24). С. 41–51. DOI: 10.20535/2617-9741.3.2025.340376. URL: https://chemengine.kpi.ua/article/view/340376

4. Галлямов К. С., Красніков І. Л., Левадна О. С. Інтеграція SCADA-систем із реляційними базами даних у системах промислової автоматизації. Інформаційні технології і автоматизація – 2025 : матеріали 18-ї Міжнародної науково-практичної конференції, Одеса, 2025. С. 389–392. URL: https://repository.kpi.kharkov.ua/entities/publication/19036f9b-3e24-46ca-bf33-d1507574718d

5. ISO 14644-3:2019. Cleanrooms and associated controlled environments — Part 3: Test methods. URL: https://www.iso.org/obp/ui/en/#iso:std:iso:14644:-3:ed-2:v2:en

6. ISO 14644-4:2022. Cleanrooms and associated controlled environments — Part 4: Design, construction and start-up. URL: https://www.iso.org/standard/72379.html

7. ISO 14644-16:2019. Cleanrooms and associated controlled environments — Part 16: Energy efficiency in cleanrooms and separative devices. URL: https://www.iso.org/standard/66331.html

8. ISO 14644-17:2021. Cleanrooms and associated controlled environments — Part 17: Particle deposition rate applications. URL: https://www.iso.org/obp/ui/en/#iso:std:iso:14644:-17:ed-1:v1:en

9. ISO 21501-4:2018. Determination of particle size distribution — Single particle light interaction methods — Part 4: Light scattering airborne particle counter for clean spaces. URL: https://www.iso.org/standard/67843.html

10. МОЗ України. Настанова «Лікарські засоби. Належна виробнича практика. СТ-Н МОЗУ 42-4.0:2020». URL: https://www.dls.gov.ua/wp-content/uploads/2020/05/Настанова-СТ-Н-МОЗУ-42-4.0_2020.pdf

11. МОЗ України. Про внесення змін до Додатка 1 до настанови «Лікарські засоби. Належна виробнича практика. СТ-Н МОЗУ 42-4.0:2020». 2023. URL: https://zakon.rada.gov.ua/go/v1604282-23

12. МОЗ України. Наказ від 08.11.2024 №1883 «Про затвердження Зміни до настанови “Лікарські засоби. Належна виробнича практика. СТ-Н МОЗУ 42-4.0:2020”». URL: https://moz.gov.ua/uk/decrees/nakaz-moz-ukrayini-vid-08-11-2024-1883-pro-zatverdzhennya-zmini-do-nastanovi-likarski-zasobi-nalezhna-virobnicha-praktika-st-n-mozu-42-4-0-2020-sho-rekomendovano-do-zastosuvannya-sub-yektami-gospodaryuvannya

13. European Commission. EudraLex Volume 4. Annex 1: Manufacture of Sterile Medicinal Products. Revised version, 2022. URL: https://health.ec.europa.eu/system/files/2022-08/20220825_gmp-an1_en_0.pdf

14. European Commission. EudraLex Volume 4. Good Manufacturing Practice guidelines. URL: https://health.ec.europa.eu/medicinal-products/eudralex/eudralex-volume-4_en

15. World Health Organization. TRS 1044: WHO Expert Committee on Specifications for Pharmaceutical Preparations. 2022. URL: https://www.who.int/publications/i/item/9789240063822

16. PIC/S. PE 009-17: Guide to Good Manufacturing Practice for Medicinal Products. 2023. URL: https://picscheme.org/en/publications

17. ISPE. GAMP 5: A Risk-Based Approach to Compliant GxP Computerized Systems. 2nd ed. 2022. URL: https://guidance-docs.ispe.org/doi/book/10.1002/9781946964571

18. IEC 62443-4-1:2018. Security for industrial automation and control systems — Part 4-1: Secure product development lifecycle requirements. URL: https://webstore.iec.ch/publication/33615

19. IEC 62443-4-2:2019. Security for industrial automation and control systems — Part 4-2: Technical security requirements for IACS components. URL: https://webstore.iec.ch/publication/34421

20. OPC Foundation. OPC Unified Architecture Specification. Version 1.05.03. 2023. URL: https://reference.opcfoundation.org/

21. PROFIBUS & PROFINET International. PROFINET System Description. URL: https://www.profibus.com/technology/profinet

22. OASIS Standard. MQTT Version 5.0 Specification. 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

23. Eclipse Foundation. Sparkplug Specification Version 3.0.0. 2022. URL: https://sparkplug.eclipse.org/specification/

24. FastAPI Documentation. URL: https://fastapi.tiangolo.com/

25. Eclipse Paho MQTT Python Client Documentation. URL: https://eclipse.dev/paho/files/paho.mqtt.python/html/

26. SQLite Documentation. URL: https://www.sqlite.org/docs.html

27. Python Software Foundation. Python 3.12 Documentation. 2024. URL: https://docs.python.org/3.12/

28. Raspberry Pi Ltd. Raspberry Pi Documentation. URL: https://www.raspberrypi.com/documentation/

29. STMicroelectronics. STM32Cube MCU Package Documentation. URL: https://www.st.com/en/embedded-software/stm32cube-mcu-packages.html

30. Eclipse Foundation. Mosquitto MQTT Broker Documentation. URL: https://mosquitto.org/documentation/

31. scikit-learn developers. scikit-learn User Guide. Version 1.5. 2024. URL: https://scikit-learn.org/stable/user_guide.html

32. Taylor S. J., Letham B. Forecasting at Scale. The American Statistician. 2018. Vol. 72(1). P. 37–45. DOI: 10.1080/00031305.2017.1380080. URL: https://doi.org/10.1080/00031305.2017.1380080

33. Hyndman R. J., Athanasopoulos G. Forecasting: Principles and Practice. 3rd ed. OTexts, 2021. URL: https://otexts.com/fpp3/

34. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 3rd ed. O'Reilly Media, 2022. URL: https://www.oreilly.com/library/view/hands-on-machine-learning/9781098125967/

35. Montgomery D. C. Introduction to Statistical Quality Control. 8th ed. Wiley, 2019. URL: https://www.wiley.com/en-us/Introduction+to+Statistical+Quality+Control%2C+8th+Edition-p-9781119399308

36. NIST. Guide to Operational Technology Security. Special Publication 800-82 Revision 3. 2024. URL: https://csrc.nist.gov/pubs/sp/800/82/r3/final

37. National Institute of Standards and Technology. Artificial Intelligence Risk Management Framework (AI RMF 1.0). 2023. URL: https://www.nist.gov/itl/ai-risk-management-framework

38. MDN Web Docs. WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

39. Chart.js Documentation. URL: https://www.chartjs.org/docs/latest/

40. OpenRouter Documentation. URL: https://openrouter.ai/docs/api-reference/overview

ДОДАТОК А

ПРИКЛАД MQTT-ПОВІДОМЛЕННЯ ПРОМИСЛОВОГО ШЛЮЗУ


```
{
  "gateway_id": "RPI-GW-01",
  "controller_id": "PLC-01",
  "controller_type": "PLC",
  "zone": "grade-c",
  "sensor_id": "PLC-01-TT-001",
  "sensor_type": "temperature",
  "value": 22.4,
  "unit": "C",
  "protocol": "PROFINET",
  "quality": "GOOD",
  "anomaly": false,
  "anomaly_type": null,
  "ts": 1710000000.0
}
```

ДОДАТОК Б

ОСНОВНІ ФАЙЛИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

– backend/database.py – схема SQLite, збереження телеметрії, топологія, control_state, operator_actions.
– backend/mqtt_subscriber.py – підписник MQTT, розбір старого і нового формату повідомлень, трансляція WebSocket.
– backend/routers/analytics.py – підсумок, топологія, ШІ-аналіз, рекомендації, дії, стан керування.
– sensor_emulator/industrial_gateway.py – емулятор ПЛК/STM32/шлюзу Raspberry Pi.
– scripts/simulate_10k_sensors.py – симуляція масштабу понад 10 тисяч сенсорів.
– dashboard/index.html, dashboard/app.js, dashboard/styles.css – україномовна вебпанель оператора.
– ai_pipeline/pipeline.py – центральний конвеєр ШІ.
– ai_pipeline/llm_analyst.py – аналітик на основі OpenRouter-сумісної мовної моделі.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Перейма Артем Юрійович

Тема: Інтелектуальна система моніторингу технологічних даних

Спеціальність: 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Обсяг кваліфікаційної роботи:

Кількість сторінок записки 65

1. Короткий зміст роботи та прийнятих рішень: Метою роботи є розробка інтелектуальної системи моніторингу технологічних даних на прикладі чистої кімнати GMP класу С

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

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі виконано аналіз предметної області інтелектуального моніторингу технологічних даних, розглянуто особливості чистої кімнати GMP класу С, передачу даних від контролерів та існуючі підходи до побудови систем моніторингу.
У другому розділі спроєктовано архітектуру інтелектуальної системи моніторингу, визначено модель даних, структуру MQTT-повідомлень, методи штучного інтелекту, контур рекомендацій та операторського керування, а також функціональні й нефункціональні вимоги до системи.
У третьому розділі розроблено програмний прототип системи, реалізовано серверний рівень, базу даних, вебпанель оператора, емуляцію контролерів, навантажувальне тестування та перевірку працездатності системи. У роботі використано сучасні програмні засоби та методи: Python, FastAPI, MQTT, SQLite, WebSocket, Isolation Forest, Random Forest та засоби прогнозування.

4. Позитивні сторони роботи: практична спрямованість, актуальність теми, використання сучасних технологій збору, передавання й аналізу технологічних даних, наявність працюючого програмного прототипу, реалізація вебпанелі оператора, підтримка промислових метаданих

5. Негативні сторони роботи: у роботі недостатньо уваги приділяється аналізу існуючих технічних рішень

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

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

8. Інші зауваження: відсутні

9. Оцінка дипломної роботи: добре (С)

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

Стецюк Михайло Васильович, ст. викладач, PhD, к-ф „Кібербезпеки"

"08" червня 2026 р. (підпис)

Завідувачу кафедри автоматизації,
комп'ютерно-інтегрованих технологій та
робототехніки Людмилі КОРЕЦЬКІЙ
здобувача вищої освіти
Перейми Артема Юрійовича
факультет ІТ, курс ІІІ, група АКІТРс-23-1

ЗАЯВА

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

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

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

02.06.2026
дата

підпис

Протокол аналізу звіту подібності експертом

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

Автор: Артем ПЕРЕЙМА

Співавтор:

Назва: КвР_Перейма_Артем_інтелектуальна_система_моніторингу_оновлено

Експерт: Микола ФЕДУЛА

Підрозділ: Кафедра автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

Коефіцієнт подібності 1:0.08%

Коефіцієнт подібності 2:0%

Мікропробіли: 16

Заміна букв: 6

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-06-17 08:38:36.0

Після аналізу Звіту подібності констатую наступне:

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

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

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

Обгрунтування:

2026-06-17 Доцент Микола Федула

Дата експерт

Anti-Plagiarism v-15.258 (global version)

The maximum coincidence with one document 3.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 14%**

ID: 275696 Title: БКР Інтелектуальна система моніторингу технологічних даних Added in a DB: 2026-06-17 Authors: Артем ПЕРЕЙМА Heads: Микола ФЕДУЛА Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	49507	507	3317 (7%)	43 (8%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ АВТОМАТИЗАЦІЇ, КОМП'ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи: «Інтелектуальна система моніторингу технологічних даних»
Автор: Перейма Артем Юрійович
Освітня програма: Освітньо-професійна програма «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Рівень вищої освіти: перший (бакалаврський)
Спеціальність: 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Науковий керівник: Федула Микола Васильович, кандидат технічних наук, доцент

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

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

Підтвердження:
Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:
1) у тексті роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках, у структурі змісту, назвах розділів/підрозділів, у назвах публікацій у переліку джерел посилання;
2) усі запозичення є фрагментарними або мають належним чином оформленні посилання;
3) виявлені модифікації тексту не впливають на відсоток схожості.
Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 5,01% і адресується до 10 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 19.06.2026

Завідувач кафедри _____ Людмила КОРЕЦЬКА
Гарант освітньої програми _____ Юрій ФОРКУН
Керівник кваліфікаційної роботи _____ Микола ФЕДУЛА