МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління

ІВАНОВ Андрій Михайлович

Метод прогнозування обслуговування в системах моніторингу на базі IoT / Method for service forecasting in IoT-based monitoring systems

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

Кваліфікаційна робота

Виконав студент групи КНм-21
А.М. Іванов
_____

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

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

ТЕРНОПІЛЬ - 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «магістр»
спеціальність: 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Іванову Андрію Михайловичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи
Метод прогнозування обслуговування в системах моніторингу на базі IoT / Method for service forecasting in IoT-based monitoring systems
керівник роботи к.т.н., доцент, О.Р. Осолінський
затверджені наказом по університету від 20 грудня 2024 року № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 1 грудня 2025 р.

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

4. Основні питання, які потрібно розробити
- Проаналізувати інтелектуальні системи моніторингу на базі IoT;
- Проаналізуцвати роль сенсорних мереж, шлюзів і хмарних платформ
- Проаналізувати проблеми прогнозного обслуговування у розподілених IoT-системах;
- Вибрати перспективний шлях і постановка задачі дослідження;
- Розробити концепцію побудови аналітичного шару IoT-системи моніторингу;
- Розробити метод прогнозування обслуговування;
- Розробити математичну модель прогнозування та формалізації задачі оптимізації;
- Розробити модуль збору та попередньої обробки IoT-даних;
- Реалізувати прогнозну модель та аналітичний блок;
- Провести інтеграцію з платформою моніторингу та візуалізацію результатів;
- Провести оцінювання ефективності методу прогнозування.

5. Перелік графічного матеріалу у роботі
- загальна архітектура системи;
- архітектура гібридної моделі прогнозування на базі LSTM + ARIMA + XGBoost

6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

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

Студент _____ А. М. Іванов
підпис

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

РЕЗЮМЕ

Кваліфікаційна робота на тему «Метод прогнозування обслуговування в системах моніторингу на базі IoT» на здобуття освітнього ступеня «Магістр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 88 сторінки і містить 43 ілюстрації, 4 додатки та 45 використаних джерел.

Метою даної кваліфікаційної роботи є розробка методу прогнозування обслуговування в системах моніторингу на базі Інтернету речей.

Методи досліджень: методи, прогнозного обслуговування та аналізу часових рядів в Інтернеті речей, математичного моделювання процесів деградації обладнання, методів машинного навчання і глибоких нейронних мереж а також методи проектування та розроблення програмного забезпечення для розподілених IoT-систем.

Результати дослідження: удосконалено підхід до прогнозного технічного обслуговування в розподілених системах моніторингу на базі Інтернету речей шляхом розроблення методу прогнозування на основі індексу технічного стану та гібридної моделі часових рядів, що поєднує LSTM, ARIMA та XGBoost. Запропоновано математичну модель прийняття рішень щодо моменту технічного обслуговування.

Результати роботи можуть бути використані в системи моніторингу промислового обладнання на базі IoT та використовуватися як аналітичний модуль цифрових двійників.

Ключові слова: ІНТЕРНЕТ РЕЧЕЙ, ПРОГНОЗНЕ ТЕХНІЧНЕ ОБСЛУГОВУВАННЯ, ЧАСОВІ РЯДИ, СЕНСОРНІ ДАНІ; HEALTH INDEX; LSTM, ARIMA, XGBOOST, ЦИФРОВИЙ ДВІЙНИК, СИСТЕМИ МОНІТОРИНГУ ОБЛАДНАННЯ

ABSTRACT

Qualification work on the topic «Method for service forecasting in IoT-based monitoring systems» for Master's degree on speciality 122 «Computer Science» educational and professional program «Computer Science» is written on 88 pages and it contains 43 figures, 4 annexes and 45 references.

The aim of this qualification thesis is to develop a maintenance prediction method for monitoring systems based on the Internet of Things.

Research methods: methods of predictive maintenance and time series analysis in the Internet of Things, mathematical modelling of equipment degradation processes, machine learning and deep neural network methods, as well as methods for the design and development of software for distributed IoT systems.

Research results: the approach to predictive maintenance in distributed IoT-based monitoring systems has been improved by developing a prediction method based on a health index and a hybrid time-series model that combines LSTM, ARIMA and XGBoost. A mathematical decision-making model for determining the moment of maintenance has been proposed.

The results can be used in IoT-based monitoring systems for industrial equipment and employed as an analytical module of digital twins.

Keywords: INTERNET OF THINGS; PREDICTIVE MAINTENANCE, TIME SERIES, SENSOR DATA, HEALTH INDEX, LSTM, ARIMA, XGBOOST, DIGITAL TWIN, EQUIPMENT MONITORING SYSTEMS

ЗМІСТ

ВСТУП

Сучасна промислова інфраструктура швидко переходить до концепції розумного виробництва, де ключову роль відіграють системи моніторингу на базі Інтернету речей (IoT). На обладнанні встановлюються сотні сенсорів, які безперервно вимірюють температуру, вібрацію, струм, тиск, навантаження та інші параметри. Ці дані дають змогу оцінювати технічний стан вузлів в реальному часі, зменшувати простої та підвищувати безпечність експлуатації. Водночас зростає вартість простоїв і аварійних відмов, тому що зупинка одного критичного агрегату може призвести до значних фінансових втрат, порушення виробничого плану та ризиків для персоналу.

Традиційні підходи до технічного обслуговування вже не відповідають вимогам такої цифрової інфраструктури. Тому найбільш перспективним напрямом у цій сфері є прогнозне технічне обслуговування (PdM), яке використовує сенсорні дані, методи машинного навчання та статистичного аналізу для передбачення моменту виникнення відмови.

Крім того не всі існуючі підходи явно враховують економічні наслідки рішень щодо часу технічного обслуговування, тобто співвідношення витрат на ремонт і очікуваних втрат від можливої відмови.

Тому розроблення методу прогнозування технічного обслуговування, який працює з реальними потоками IoT-даних та підтримує прийняття рішень з врахуванням очікуваних витрат є актуальною задачею.

Метою роботи є розробка методу прогнозування обслуговування в системах моніторингу на базі Інтернету речей, який підвищує точність оцінювання технічного стану обладнання, зменшує ймовірність аварійних відмов і дозволяє оптимізувати витрати на технічне обслуговування за рахунок використання гібридних моделей аналізу часових рядів.

Об’єктом дослідження є процес прогнозного технічного обслуговування промислового обладнання в розподілених системах моніторингу на базі IoT.

Предметом дослідження є методи і моделі інтелектуального аналізу часових рядів сенсорних даних в розподілених IoT системах, правила прийняття рішень проведення технічного обслуговування з врахуванням витрат.

Методи дослідження включають аналіз наукових публікацій в галузі Інтернету речей, прогнозного обслуговування та аналізу часових рядів, математичного моделювання процесів деградації обладнання, методів машинного навчання і глибоких нейронних мереж а також методи проектування та розроблення програмного забезпечення для розподілених IoT-систем.

Практичне значення одержаних результатів полягає в тому, що розроблений метод може бути інтегрований у системи моніторингу промислового обладнання на базі IoT та використовуватися як аналітичний модуль цифрових двійників або систем управління технічним обслуговуванням.

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

Апробація результатів дослідження. Основні теоретичні положення роботи та практичні результати дослідження доповідалися і обговорювалися на IV Міжнародній науковій конференції X Міжнародній науковій конференції «Проблеми та перспективи реалізації та впровадження міждисциплінарних наукових досягнень», м. Чернігів, Україна та IX Міжнародній студентській конференції Пріоритетні напрямки та вектори розвитку світової науки (м. Суми, Україна).

1 АНАЛІЗ МЕТОДІВ ІНТЕЛЕКТУАЛЬНОГО МОНІТОРИНГУ НА БАЗІ ІОТ

1.1 Інтелектуальні системи моніторингу на базі IoT

Інтернет речей — це технологічна парадигма, що описує екосистему з'єднаних між собою фізичних та віртуальних об'єктів, здатних ідентифікуватися, обмінюватися даними, взаємодіяти з навколишнім середовищем та ініціювати керуючі дії [16-19]. На відміну від класичної парадигми machine-to-machine (M2M), де взаємодія обмежена вузькими вертикалями та специфічними протоколами, сучасний IoT передбачає масштабну інтероперабельність, уніфіковані моделі даних і сервіс-орієнтовані інтерфейси [17-19]. Ця еволюція стала можливою завдяки широкій доступності бездротових технологій зв'язку, здешевленню сенсорних технологій та мікроконтролерів, а також появі хмарних обчислень і машинного навчання [18,20,39].

Ключовою рисою IoT є здатність "речей" бути суб'єктами цифрового світу (рисунок 1.1). Вони мають унікальну ідентичність маючи IP – адрес або мітку, генерують телеметрію в реальному часі та можуть впливати на процеси через виконавчі механізми [16, 17]. Розумний пристрій або річ в цьому контексті це не тільки фізичний пристрій до якого може входити сенсор або лічильник енергії, а і віртуальна сутність, що представляє собою цифровий двійник, програмний агент або логічний сервіс, який представляє стан або поведінку реального об'єкта [12,18,42]. Тобто поєднання фізичних і віртуальних речей, зведених у спільну комунікаційно-обчислювальну інфраструктуру, робить IoT фундаментом інтелектуальних систем моніторингу [19,36,41].

Еволюційно IoT пройшов кілька хвиль розвитку. Початковий етап зосереджувався на ідентифікації об'єктів і простому зборі даних. Друга хвиля пов'язана з конвергенцією IP-мереж, появою хмар і великих даних де можна централізовано зберігати, агрегувати та аналізувати [17,19,39]. Сучасний етап — це перехід до розподіленої інтелектуальної обробки, стандартизованих моделей інтероперабельності та використання алгоритмів штучного інтелекту для прогнозування, оптимізації та автономного керування [18,22,30].

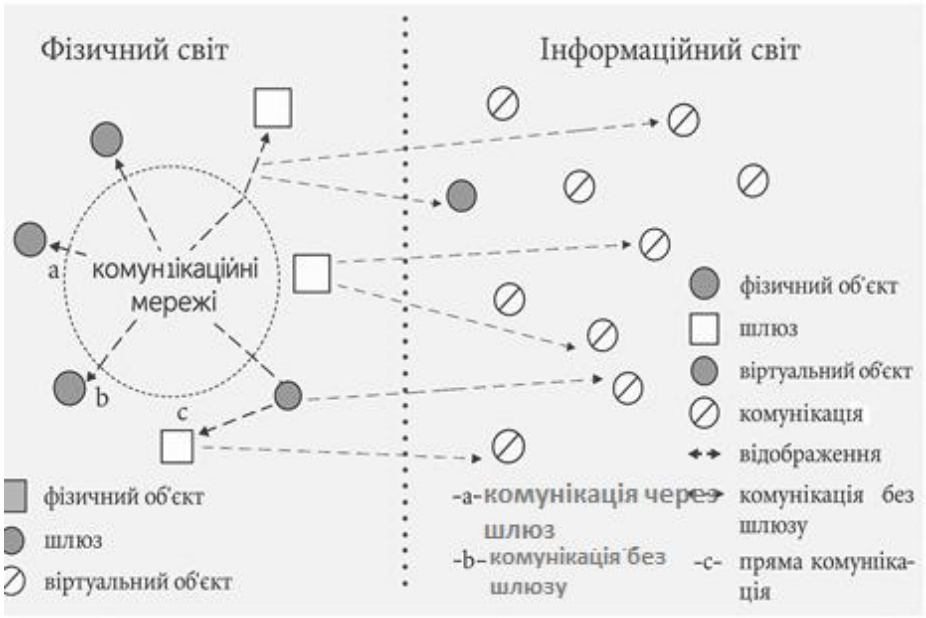


Рисунок 1.1 – Технічний огляд Інтернету речей

Для задач моніторингу технічного стану це означає можливість відстежувати зміни параметрів в режимі близькому до реального часу, пояснювати їхню причину та завчасно планувати обслуговування [20, 23, 25].

Крім того дана парадигма надає сервіс-орієнтований характер сучасного IoT тобто сенсорні спостереження трансформуються в цифрові сервіси — потоки подій, API для запитів стану, вебхуки для оповіщень, служби аналітики, моделі прогнозування [16, 19]. Це забезпечує повторне використання даних і компонентів в різних сценаріях, таких як диспетчеризація і прогнозне обслуговування та оптимізація виробничих циклів [18, 20, 36, 41].

1.1.1 Архітектура IoT-систем

Базова архітектура IoT в більшості це багатошарова структура. Найчастіше виділяють три або п’ять рівнів [16, 17, 36]. Трирівнева модель описує: (1) сенсорний рівень, (2) мережевий рівень, (3) прикладний рівень.

П’ятирівнева — деталізує середні шари, додаючи рівень абстракції і керування послугами та бізнес-рівень [17, 29, 36].

Сенсорний рівень містить сенсори, виконавчі механізми та вбудовані обчислення на мікроконтролерах (MCU) чи одноплатних комп’ютерах (SBC). На цьому рівні виконуються первинний збір, калібрування, видалення шумів,

агрегування та, за наявності ресурсу, локальна аналітика яка може базуватись на різних фільтрах, порогових правилах та базових класифікаторах [19, 20, 34, 41].

Мережевий рівень (рисунок 1.2) відповідає за транспортування телеметрії і команд керування між пристроями, шлюзами та сервісами. Він охоплює такі локальні та глобальні технології, як Wi-Fi, Ethernet, 4G/5G, LPWAN, маршрутизацію, шифрування та механізми забезпечення якості обслуговування (QoS). На цьому рівні часто реалізуються брокери повідомлень і шини подій, служби реєстрації пристроїв, каталогів тем і політик безпеки [20, 39, 41].

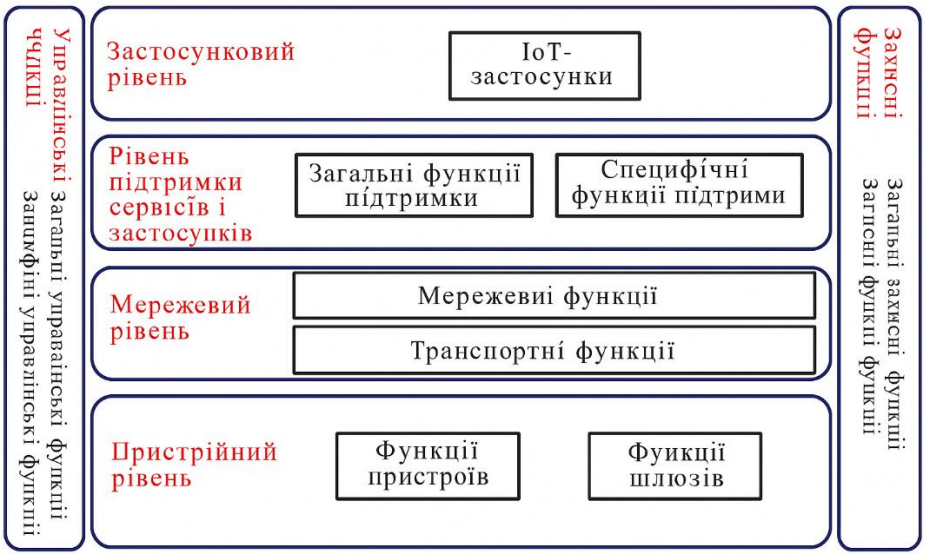


Рисунок 1.2 – Модель IoT

Прикладний рівень забезпечує аналітику, візуалізацію та інтеграцію з бізнес-процесами. Тут працюють сервіси зберігання часових рядів (TSDB), поточні обчислення, моделі машинного навчання для детекції аномалій та прогнозування, дашборди та API для зовнішніх систем [18, 19, 25, 39].

В п’ятирівневій моделі до мережевого та прикладного додають:

– Рівень абстракції та керування послугами (middleware) —це реєстри пристроїв, служби ідентифікації та аутентифікації, керування життєвим циклом, трансформації та нормалізація даних, моделі ресурсів і семантичні описи [16, 17, 29];

– Бізнес-рівень — це політики, KPI, правила відповідності, інтеграція аналітики з контуром прийняття рішень і планування обслуговування, побудова графіків ТО на основі оцінки ризику відмов [21, 28–30].

Частиною архітектурного проектування є вбудовані горизонтальні функції, такі як, шифрування, керування ключами, авторизація, керованість, моніторинг пристроїв і сервісів, ведення журналу подій, повідомлення, інтероперабельність, масштабованість та відмовостійкість [17, 19, 28, 29].

Крім того сучасна архітектура є не тільки орієнтованою на хмарні сервіси. Зараз наприклад домінує композиція edge–fog–cloud. Edge обробляє “гарячі” потоки на місці, що дає нижчу затримку, менший трафік у каналі, fog узгоджує локальні домени такі як агрегація, координація, cloud забезпечує довгострокове зберігання, складні аналітичні обчислення, MLOps і інтеграцію з корпоративними системами [18, 20, 22, 39, 41].

1.2 Роль сенсорних мереж, шлюзів і хмарних платформ

Сенсорні мережі призначені для безперервного спостереження за фізичними процесами, дотримуючись обмежень енергоспоживання, смуги пропускання та вартості [16, 20, 25].

Для моніторингу технічного стану характерні сенсори вібрацій, температури, струмів, тиску, акустичні/ультразвукові сенсори, енкодери обертів, лічильники циклів (рисунок 1.3) [20, 24, 34, 35]. Часто поєднують різні типи датчиків, щоб підвищити достовірність результатів, так званий сенсорний ансамбль. Також в таких мережах важливо враховувати основні метрологічні характеристики, такі як чутливість, похибка, дрейф і режим вибірки, тому що для PdM критичні висока дискретизація на швидких каналах та синхронізація [21, 25, 28].

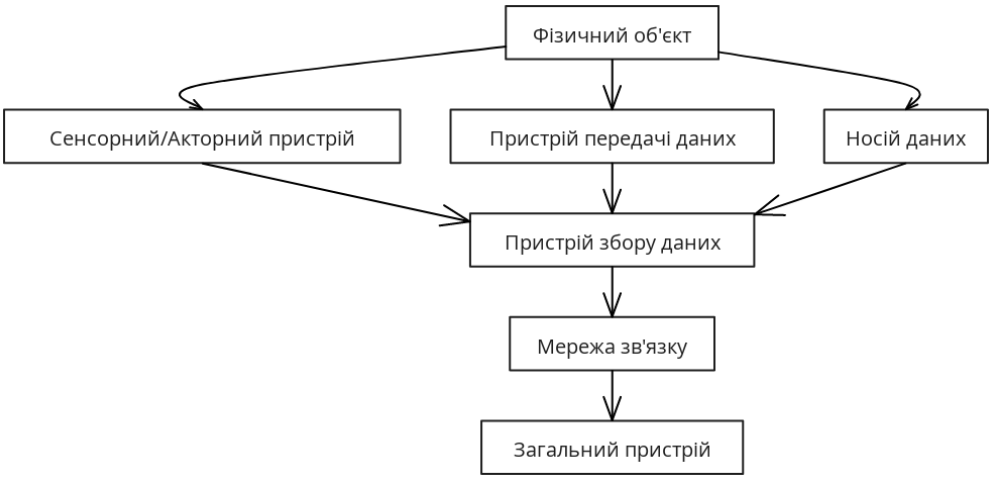


Рисунок 1.3 – Типи пристроїв та їх зв'язок з фізичними речами

Шлюзи виконують роль перекладачів та диспетчерів між пристроями та сервісами аналізу [19, 20, 36, 41]. Типові функції шлюзів:

–агрегація та мультиплексування каналів;

–трансляція протоколів і моделей;

–локальна аналітика і фільтрація;

–буферизація та попереднє збереження на вузлі при збоях зв’язку;

–безпека на краю;

–керування життєвим циклом пристроїв [20, 25, 39].

Для цехових і енергетичних сценаріїв шлюз часто має інтерфейси до промислових шин і протоколів. В розподілених системах шлюзи працюють на польових контролерах або SBC з підтримкою мобільного зв’язку [18, 20, 36].

Хмара забезпечує гнучкі ресурси: сховища часових рядів, обробку потоків, сервіси даних, моделі ML/AI та MLOps, візуалізацію і інтеграційні шари [18, 19, 25, 39].

До переваг можна віднести масштабування, управління версіями, ізоляція середовищ, резервування, глобальна доступність [19, 30, 31]. Недоліки — залежність від каналу, латентність, вартість трафіку. Тому разом із хмарою активно застосовують edge-аналітику, контейнеризацію, апаратні прискорювачі, локальні брокери і шини подій [18, 20, 22, 39]. Такий підхід дозволяє виконувати time-critical логіку на місці і відправляти в хмару лише релевантні підсумки або значущі фрейми для донавчання моделей [22, 25, 39].

1.2.1 Потоки даних у системах моніторингу технічного стану

Моніторинг технічного стану Condition Monitoring, (CM) і прогнозне обслуговування Predictive Maintenance, (PdM) висувають специфічні вимоги до конвеєрів даних, оскільки корисний сигнал часто “тонкий”, а часові закономірності — ключові [4, 5, 21, 23].

Типова траєкторія даних:

–сенсор формує сирі вимірювання;

–локальна попередня обробка на пристрої/шлюзі;

–доставлення у брокер/шину подій з QoS і політиками ретенції;

–потокова аналітика;

–збагачення метаданими;

–збереження в озері даних з індексацією за часом/обладнанням;

–інференс моделей прогнозування;

–формування робочих замовлень, корекція режимів [20,22,25,39].

Для польових вузлів і шлюзів широко застосовують MQTT з темами, ієрархіями топіків і рівнями QoS. Там, де потрібні високі обсяги та складні топології потоків, використовують Kafka [20, 22, 39].

Для PdM вирішальними є коректні часові мітки, синхронність каналів і сталість частоти вибірки. Потокові фреймворки працюють з подіями у “event time”, підтримуючи watermarks і затримку пізніх подій. Це критично для точного розрахунку агрегатів у “ковзних” вікнах, а також для побудови ознак, чутливих до фази процесу [22, 25].

InfluxDB використовують щільні часові індекси, стиснення, політики утримання для швидких запитів. Для CM корисні комбіновані схеми: “гарячі” дані з високою частотою — на локальному сервері, “теплі” — у TSDB для оперативної аналітики, “холодні” — у Data Lake для ретроспективного аналізу й донавчання моделей [23, 25, 39].

Моніторинг технічного стану спирається на доменно-специфічні ознаки та загальні ML-підходи [4, 5, 21, 40].

Для прийняття рішень щодо обслуговування використовують:

– RUL-оцінку прогноз часу до відмови;

– Прогноз ймовірності відмови в інтервалі часу;

– Станові класи, на які накладаються бізнес-правила для формування заявок ТО [3,7–11, 26, 27].

Крім того виробничі сценарії вимагають передбачуваних затримок і гарантованої доставки. В цих випадках можна використати підтвердження в MQTT, буферизацію на шлюзах, політики backpressure в потокових фреймворках, а також “circuit-breaker”-патерни в мікросервісах аналітики [22, 25, 39]. Для критично важливих вузлів передбачають незалежні канали зв’язку, локальні правила безпечної зупинки та “деградаційні” режими [28, 29, 32].

Grafana формує багатошарові дашборди до яких входять телеметрія в реальному часі, тренди та карти тепла для аномалій [25, 32, 39]. Інтеграції з CMMS автоматизують цикл прогноз, робоче замовлення, ресурсне планування, відповідь у модель”. Останній крок — замикання MLops-циклу до якого входять збір еталонів, онлайн-моніторинг якості, планове донавчання, контроль версій моделей і безпечне впровадження [22, 23, 31].

1.3 Методи прогнозування обслуговування обладнання

Технічне обслуговування є невід’ємною частиною управління життєвим циклом будь-якого технічного об’єкта, тому що воно забезпечує безперебійну роботу систем, запобігає аваріям і підвищує ефективність використання ресурсів [21, 28, 29]. Сучасна промисловість поступово переходить від реактивного підходу до прогнозного, використовуючи аналітику даних та Інтернет речей для своєчасного виявлення потенційних відмов [2–5, 23, 30]. Умовно всі типи технічного обслуговування поділяють на три основні категорії: реактивне, планове та прогнозне (рисунок 1.4) [21, 23, 40].

Рисунок 1.4 – Еволюція підходів до технічного обслуговування: від реактивного до прогнозного.

Реактивне обслуговування є найпростішим і найстарішим підходом, що ґрунтується на усуненні поломки після її виникнення. Воно не потребує попередніх витрат на діагностику, однак супроводжується значними ризиками тривалих простоїв, втратами продуктивності та додатковими витратами на екстрений ремонт [21, 30]. Цей підхід є допустимим для дешевих і не критичних елементів систем, але неприйнятним для складних або безперервних виробничих процесів.

Планове технічне обслуговування передбачає виконання регламентних робіт у визначені інтервали часу або після певної кількості робочих циклів. Його мета — мінімізувати ймовірність відмови шляхом заміни або перевірки деталей до того, як вони вийдуть з ладу. Планове обслуговування (рисунок 1.5) є більш ефективним за реактивне, однак має суттєвий недолік — воно базується на усереднених даних і не враховує реальний технічний стан обладнання [21, 23, 40].

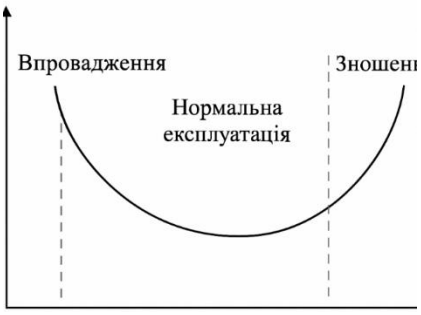


Рисунок 1.5 – Типова крива планового технічного обслуговування та імовірності відмов від часу експлуатації обладнання.

Через це ресурси часто витрачаються передчасно, або навпаки — поломка настає раніше, ніж заплановано.

Найсучаснішим підходом є прогнозне обслуговування, що базується на аналізі поточних сенсорних даних і застосуванні аналітичних моделей для передбачення моменту виникнення відмови [2–5, 21, 23, 24]. Основна ідея PdM полягає в тому, щоб виконувати технічне втручання саме тоді, коли воно дійсно необхідне. Для цього використовується велика кількість даних про температуру, вібрацію, тиск, струм, шум тощо, які надходять із сенсорних систем і дозволяють оцінити стан механізмів у реальному часі [3, 4, 20, 24, 34, 35]. Цей підхід знижує частоту аварійних ремонтів, мінімізує простої й підвищує термін служби обладнання.

В межах цієї концепції виник термін «залишковий ресурс» Remaining Useful Life, (RUL), який визначає, скільки часу залишилося до настання критичної відмови [7–10, 15, 26]. Для обчислення RUL застосовуються методи прогнозування, що належать до трьох основних груп — статистичних, методів машинного навчання та глибоких нейронних моделей [3–7, 21, 23].

Статистичні методи є найпростішими й найдавнішими у сфері прогнозного обслуговування. Вони базуються на обробці історичних даних і визначенні трендів деградації обладнання. До таких методів належать лінійна, логістична та поліноміальна регресія, яка дозволяє моделювати залежність між часом експлуатації та показниками стану; моделі часових рядів ARIMA, які прогнозують майбутні значення параметрів; а також аналіз надійності за допомогою розподілу Вейбулла чи функції виживання. Додатково застосовуються методи експоненціального згладжування, контрольні карти Шухарта, CUSUM і EWMA [4, 5, 21, 40], що дозволяють виявляти відхилення від нормального режиму роботи. Перевагою цих методів є простота, зрозумілість і можливість швидкого впровадження без потреби великих обчислювальних ресурсів. Недоліком є те, що вони погано працюють з нелінійними процесами, не здатні враховувати взаємодію великої кількості параметрів і вимагають припущень щодо розподілу даних. Тому з розвитком Інтернету речей і великих даних більшого поширення набули методи машинного навчання. Їхня перевага полягає у здатності виявляти складні закономірності у багатовимірних даних без потреби ручного проектування моделей.

Алгоритми машинного навчання можуть автоматично адаптуватися до нових умов і покращувати точність прогнозів з кожним новим циклом збору даних. Найбільш відомими алгоритмами ML у сфері PdM є дерева рішень, (Random Forest), градієнтне бустингування (XGBoost, CatBoost), метод опорних векторів (SVM), а також алгоритми кластеризації та пошуку найближчих сусідів (kNN). Вони дозволяють класифікувати стани обладнання як «нормальний», «попереджувальний» чи «критичний» та оцінювати ймовірність відмови.

Для зменшення розмірності даних часто застосовують метод головних компонент (PCA) або незалежного компонування (ICA). Більш складні підходи включають приховані марківські моделі (HMM), які описують ймовірні переходи між станами системи. Застосування ML значно підвищує точність прогнозів, проте вимагає якісної розмітки даних і підготовки вибірки, що є складним завданням у промислових умовах.

В транспортній галузі та енергетиці ML-моделі використовують для прогнозування зносу турбін, колісних пар, насосних систем. Наприклад, у турбінних установках алгоритми навчаються на історичних даних тиску, вібрації і температури для виявлення патернів, що передують поломці підшипників. У залізничному транспорті ансамблеві моделі оцінюють ступінь деградації візків і запобігають аваріям.

Подальший розвиток PdM пов’язаний із використанням глибоких нейронних мереж, які дозволяють автоматично вилучати інформативні ознаки з великих обсягів даних [3, 6, 7, 10, 11]. Глибоке навчання особливо ефективне для аналізу часових сигналів і складних просторово-часових залежностей. Найпоширенішими архітектурами є рекурентні нейронні мережі (RNN) та їхні модифікації LSTM і GRU, що зберігають пам’ять про попередні стани системи. З їхньою допомогою моделюють еволюцію технічного стану обладнання в часі [7, 10, 26]. Для аналізу сигналів або зображень застосовують згорткові нейронні мережі (CNN), які здатні розпізнавати візуальні або спектральні шаблони деградації [3, 6, 8, 11].

Автоенкодери використовуються для безнаглядного навчання — вони виявляють аномалії, порівнюючи поточні дані з відновленими на основі

нормальної поведінки системи. Гібридні архітектури CNN-LSTM або CNN-Transformer [3, 6, 8, 11] дозволяють одночасно враховувати як просторові, так і часові взаємозв'язки, що робить їх придатними для комплексних систем, таких як виробничі лінії або енергетичні установки.

У промисловості глибокі нейронні мережі дають змогу створювати цифрові двійники обладнання (Digital Twins), які відтворюють фізичні процеси у віртуальному середовищі. Цифрові двійники постійно оновлюються IoT-даними, дозволяючи моделювати майбутні сценарії експлуатації, прогнозувати відмови та планувати технічне обслуговування [12, 18, 22, 37, 42].

Такі рішення реалізовані у компаніях General Electric, Siemens, Bosch, Toyota, Airbus та інших лідерів індустрії 4.0. Їхні PdM-системи інтегрують дані з тисяч сенсорів, забезпечують хмарну аналітику та автоматизоване управління процесами обслуговування (рисунок 1.6).

Ефективність Predictive Maintenance безпосередньо залежить від якості IoT-даних [1,3–5].

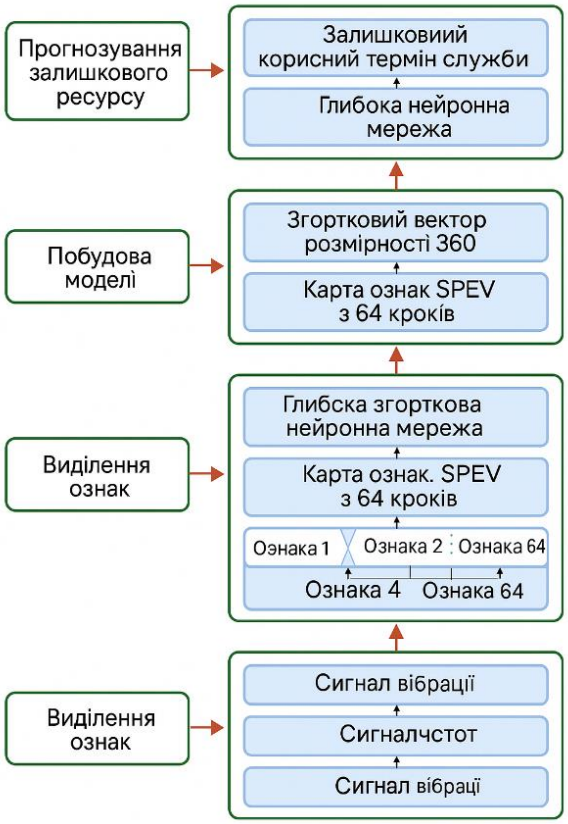


Рисунок 1.6 – Архітектура гібридної моделі CNN-LSTM для прогнозування відмов обладнання.

Потоки даних надходять через IoT-шлюзи, обробляються брокерами повідомлень і зберігаються у спеціалізованих базах даних часових рядів. Для аналізу застосовуються фреймворки Apache Spark, Flink або TensorFlow, які реалізують потокову обробку та навчання моделей у реальному часі. Типова архітектура системи PdM складається з п’яти рівнів: рівня збору даних, рівня передачі, рівня зберігання, рівня аналітики, рівня візуалізації керування, сповіщення та інтеграції.

Завдяки такій структурі PdM забезпечує замкнений цикл спостереження, аналізу, прогнозування та ідії. Отримані прогнози використовуються для автоматичного формування графіків технічного обслуговування, замовлення запчастин або повідомлення персоналу (рисунок 1.7).

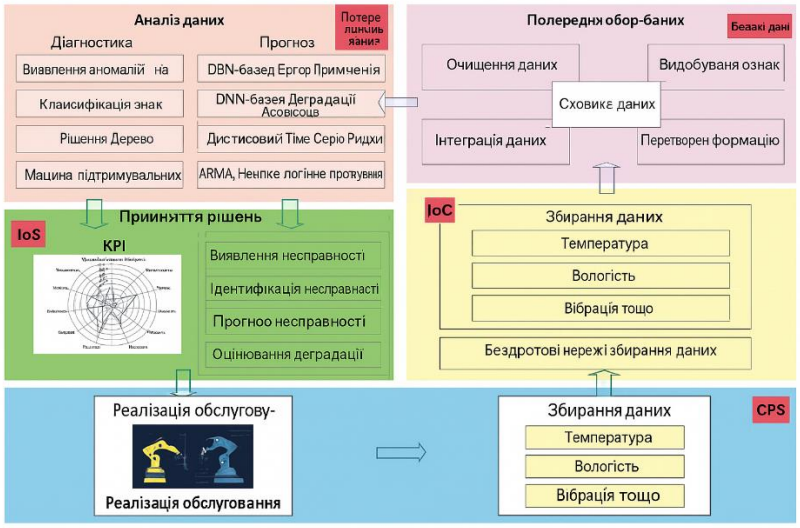


Рисунок 1.7– Архітектура системи прогнозного обслуговування на основі IoT.

Важливою тенденцією є використання методів онлайн-навчання, коли моделі постійно оновлюються під час роботи системи, що дозволяє адаптуватися до змін умов експлуатації [1, 6, 22, 23]. Інтеграція PdM з хмарними платформами робить ці рішення масштабованими, безпечними та доступними з будь-якої точки світу. Впровадження прогнозного обслуговування у виробництво знижує експлуатаційні витрати до 30 %, скорочує простої на 40 % і подовжує термін служби обладнання майже вдвічі. Таким чином, Predictive Maintenance є

ключовим компонентом концепції Smart та IoT, що забезпечує перехід від реактивного ремонту до інтелектуального управління технічним станом систем на основі даних і штучного інтелекту [2–5,18,22,24].

1.4 Проблеми прогнозного обслуговування у розподілених IoT-системах

Однією з ключових проблем, що перешкоджають ефективну реалізацію прогнозного обслуговування в розподілених системах Інтернету речей (IoT), є неповнота, зашумленість і нерівномірність потоків сенсорних даних (рисунок 1.8) [1, 20, 39].

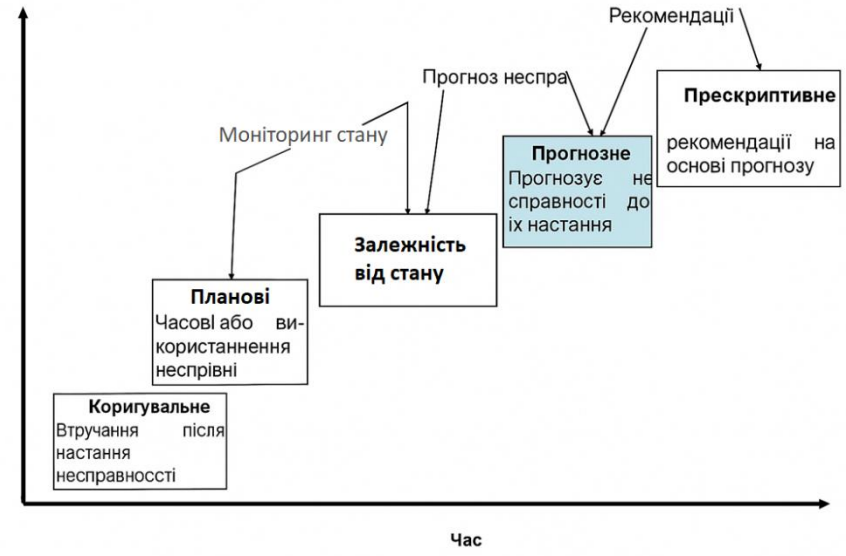


Рисунок 1.8– Еволюція стратегій технічного обслуговування

В промислових умовах дані часто надходять з різних джерел — сенсорів, контролерів, шлюзів і характеризуються різною частотою вибірки, затримками, помилками передавання або втратами пакетів. В реальних промислових середовищах значна частина даних є зашумленою або помилковою через несправність сенсорів, низький заряд батарей, вплив зовнішніх факторів чи екстремальні умови експлуатації [1, 23, 25] (рисунок 1.9).

Рисунок 1.9 – Схема виявлення аномалій

Наявність прогалин в потоках даних призводить до неможливості своєчасно виявляти деградацію обладнання, що суттєво знижує точність моделей прогнозування залишкового ресурсу. Крім того, неповнота даних порушує стабільність моделей глибокого навчання, які залежать від рівномірності часових рядів [1, 3, 6, 23, 27]. У випадках, коли сенсорна інформація надходить із різною частотою, відсутні вимірювання або спостерігаються «сплески» значень, відбувається викривлення статистичних характеристик і виникає помилка класифікації аномалій. Це особливо критично для систем реального часу, де рішення мають прийматися миттєво.

Нерівномірність потоків може бути спричинена не лише технічними обмеженнями сенсорів, а і архітектурними аспектами системи, зокрема затримками між рівнями edge–fog–cloud (рисунок 1.10) [18, 20, 22, 39]. Такі затримки змінюють ритм надходження даних, ускладнюючи узгодження часових рядів. Крім того, відсутність еталонних форматів та семантичної сумісності даних робить їхню агрегацію складною — різні пристрої використовують відмінні одиниці виміру, формати часу та протоколи передавання [17, 25, 36, 41]. В результаті це призводить до втрати цілісності аналітичних моделей та підвищує ризик хибних позитивних сигналів в процесі виявлення аномалій [1, 23, 27].

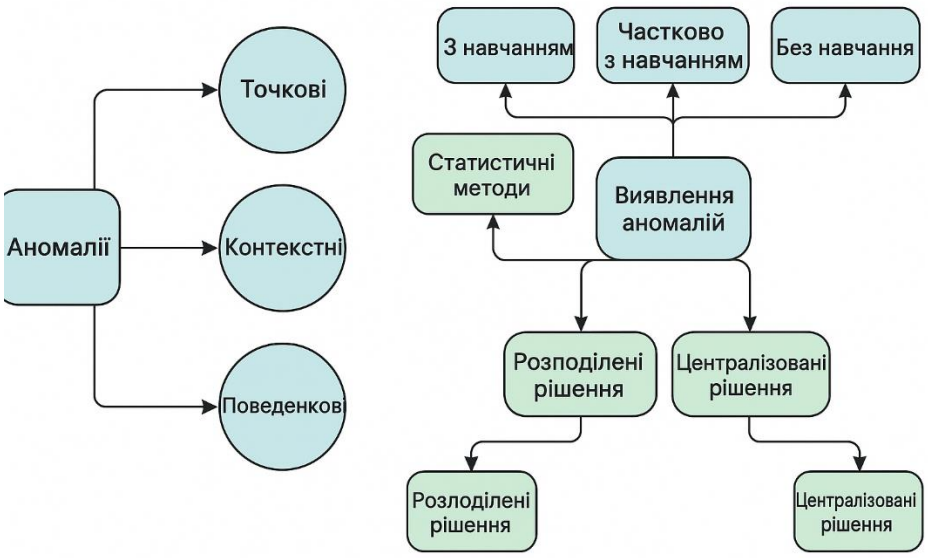


Рисунок 1.10 – Огляд типів та архітектур аномалій

Є декілька підходів до виявлення та корекції неповних або нерівномірних даних. Зокрема, методи аномалій на основі кореляцій між сенсорами дозволяють виявляти нетипові вимірювання та пропуски за рахунок аналізу взаємозв’язків між кількома сенсорними каналами. Використання кореляційних матриць, експоненційного ковзного середнього (EMA) чи Mahalanobis distance допомагає визначати відхилення в часових рядах. Крім того, застосовуються кластеризаційні методи, які дозволяють ідентифікувати локальні відхилення або розриви у потоках даних, що часто відповідають пропущеним або зашумленим вимірюванням.

Попри успіхи в моделюванні, більшість досліджень базуються на синтетичних наборах даних, тоді як реальні виробничі дані залишаються рідкісними та неповно позначеними [1, 23, 24, 35]. Це ускладнює перевірку точності алгоритмів. У розподілених IoT-системах значну складність становить синхронізація даних між вузлами та забезпечення стійкості алгоритмів до втрат або сплесків пакетів, особливо в бездротових середовищах [20, 25, 39, 41]. Крім того аномалії, спричинені несправностями сенсорів, можуть маскувати істинні деградаційні процеси, що вимагає впровадження багаторівневих моделей контролю якості даних [1, 23, 25].

Сучасні тенденції передбачають використання енергоефективних і розподілених методів обробки (рисунок 1.11), де попередня фільтрація та відновлення даних здійснюється безпосередньо на периферії [18, 20, 22, 39, 41]. Це знижує навантаження на хмарні сервери та мінімізує затримки, дозволяючи оперативно реагувати на неповноту даних. Перспективним напрямом є також поєднання методів виявлення аномалій з прогнозними моделями, що забезпечує взаємне підсилення, тобто очищені дані покращують точність прогнозів, а моделі прогнозування допомагають виявляти приховані пропуски у сенсорних потоках [1, 3, 6, 23, 27, 35].

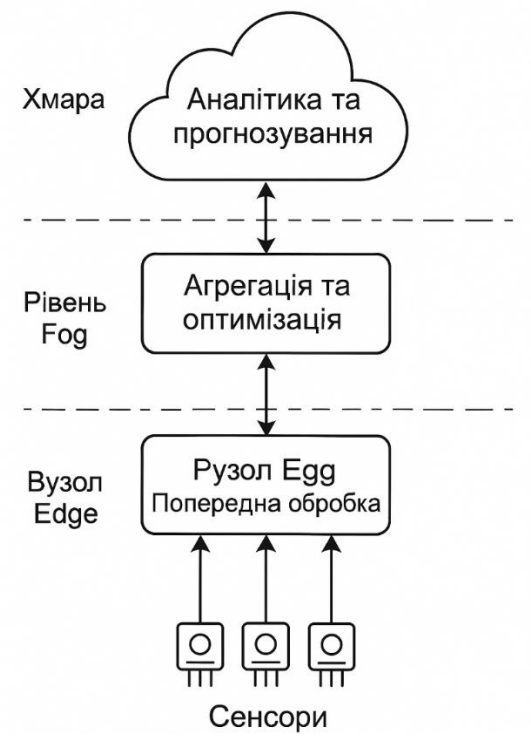


Рисунок 1.11 – Архітектура енергоефективного оброблення даних на периферії IoT

1.5 Вибір перспективного підходу та постановка задачі дослідження

При аналізі існуючих підходів до прогнозного обслуговування показано, що більшість рішень орієнтовані на відносно якісні, стаціонарні часові ряди і централізоване збирання даних у хмарі. Статистичні моделі (регресія, ARIMA, моделі надійності) забезпечують базовий прогноз тренду та індексу стану, але

погано адаптуються до нелінійних змін і багатовимірних сигналів. Методи машинного навчання та глибокі нейронні мережі значно підвищують точність прогнозу, проте чутливі до пропусків, зашумленості та нерівномірної дискретизації, а також часто не враховують розподілений характер IoT-інфраструктури та обмеження периферійних вузлів.

Додатковою проблемою розподілених IoT-систем є неповнота та нерівномірність потоків сенсорних даних, викликана перебоями зв’язку, різною частотою вибірки, відмовами сенсорів і затримками між рівнями edge–fog–cloud. Це призводить до порушення часової узгодженості, спотворення статистичних характеристик та погіршення роботи моделей прогнозування залишкового ресурсу і ймовірності відмови.

Проведений огляд також показав, що значна частина робіт зосереджується на покращенні статистичних або ML-метрик, тоді як задача вибору моменту обслуговування фактично є задачею оптимізації очікуваних витрат. З врахуванням виявлених обмежень перспективним напрямом розвитку є побудова методу прогнозування обслуговування, який одночасно:

– орієнтований на розподілену IoT-архітектуру;

– включає спеціалізований конвеєр підготовки даних;

– використовує гібридну прогностичну модель, де поєднуються глибока мережа типу LSTM, класична модель часових рядів та градієнтний бустинг для перетворення набору ознак і проміжних прогнозів у ймовірність відмови;

– інтегрує прогнозну модель в систему підтримки прийняття рішень.

Такий підхід дозволяє перейти від локальної оптимізації точності прогнозу до комплексної оптимізації експлуатаційних витрат, враховуючи специфіку сенсорних даних, обмеження розподіленої IoT-інфраструктури та вимоги до надійності виробничих процесів.

Метою роботи є розроблення методу прогнозування обслуговування в системах моніторингу на базі IoT, який забезпечує підвищену точність і стійкість прогнозу технічного стану обладнання в умовах неповних та нерівномірних сенсорних даних і дозволяє мінімізувати очікувані витрати за рахунок оптимального вибору моменту технічного обслуговування.

Для досягнення поставленої мети в роботі необхідно розв’язати такі задачі:

1. Проаналізувати існуючі підходи до побудови систем прогнозного обслуговування на базі IoT, виявити їхні переваги та обмеження щодо роботи з розподіленими сенсорними даними.

2. Проаналізувати особливості потоків даних у розподілених IoT-системах моніторингу технічного стану та сформулювати вимоги до конвеєра збору й попередньої обробки даних для задач прогнозного обслуговування.

3. Розробити модель конвеєра підготовки даних.

4. Розробити гібридний метод прогнозування технічного стану обладнання, що поєднує LSTM-мережу, модель ARIMA та XGBoost-модель.

5. Побудувати математичну модель інтеграції прогнозної підсистеми в контур прийняття рішень, сформулювати вартісний критерій вибору моменту технічного обслуговування та вивести правило ухвалення рішення на основі порівняння очікуваних витрат сценаріїв.

6. Програмно реалізувати запропонований метод в складі IoT-системи моніторингу де буде реалізовано модуль збору та попередньої обробки телеметрії, модуль формування ознак, модуль прогнозування та модуль інтерфейсу з системою управління технічним обслуговуванням.

7. Провести експериментальні дослідження на реальному або симульованому IoT-наборі даних, оцінити точність та стійкість запропонованої моделі.

8. Проаналізувати отримані результати, оцінити вплив якості попередньої обробки та вибору ознак на ефективність прогнозування.

Висновок до розділу 1

1. Проаналізовано сучасну парадигму Інтернету речей, як базу для інтелектуальних систем моніторингу. Показано, що перехід до сервіс-орієнтованої моделі створюють передумови для побудови гнучких систем моніторингу технічного стану обладнання.

2. Розглянуто багатошарову архітектуру IoT-систем які демонструють роль сенсорних мереж, шлюзів та хмарних платформ у забезпеченні безперервного збору і аналізу потоків сенсорних даних. Визначено актуальність гібридної моделі edge–fog–cloud, яка поєднує переваги локальної та хмарної обробки для задач моніторингу технічного стану.

3. Проведено огляд існуючих підходів до прогнозного обслуговування обладнання. Показано, що глибокі та гібридні моделі забезпечують більш високу точність і дають змогу реалізовувати цифрові двійники.

4. Виявлено ключові проблеми впровадження прогнозного обслуговування в розподілених IoT-системах.

5. На основі аналізу відомих рішень обґрунтовано доцільність переходу від модельно-орієнтованого покращення точності прогнозу до комплексного підходу, в якому прогнозна підсистема є частиною контуру підтримки прийняття рішень з врахуванням експлуатаційних витрат.

6. Сформульовано мету дослідження яка полягає в розробці методу прогнозування обслуговування в системах моніторингу на базі IoT, який буде стійкий до неповних і нерівномірних сенсорних даних та здатний мінімізувати очікувані витрати за рахунок оптимального планування технічного обслуговування.

2 МОДЕЛЬ ПРОГНОЗУВАННЯ ОБСЛУГОВУВАННЯ В СИСТЕМАХ IoT-МОНІТОРИНГУ

2.1 Концепція побудови аналітичного шару IoT-системи моніторингу

Із зростанням кількості IoT-пристроїв та обсягу сенсорних даних традиційна хмарна модель обчислень виявилася обмеженою через затримки, пропускну здатність каналів і високу вартість передачі даних. Сучасна концепція аналітичного шару IoT-системи базується на багаторівневій архітектурі Edge–Fog–Cloud, у якій обчислення та аналітика розподіляються ближче до джерел даних. Це дозволяє зменшити затримку, енергоспоживання і забезпечити стійку роботу в режимі реального часу.

Виходячи з обмежень та аналізу літератури аналітичний шар IoT-системи буде являти собою багаторівневу структуру (рисунок 2.1), яка буде включати чотири основні компоненти: рівень сенсорів, рівень шлюзів (edge/fog), рівень аналітики та рівень сервісів або керування.

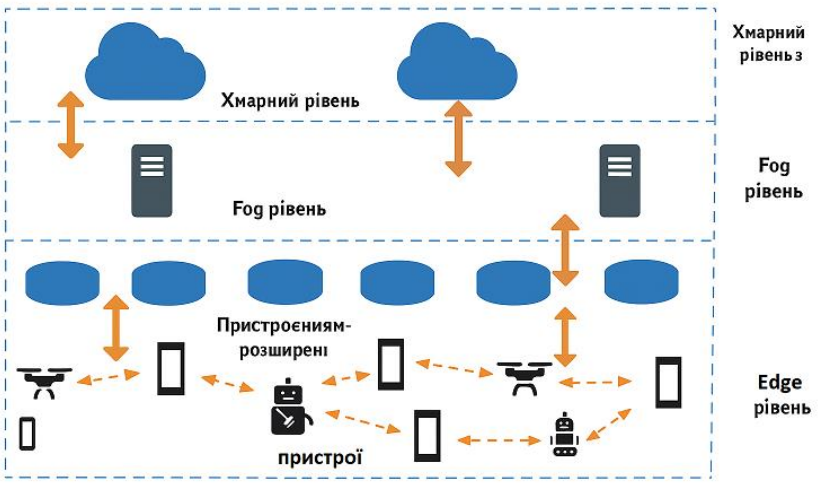


Рисунок 2.1 – Трирівнева архітектура для периферійних, туманних та хмарних обчислень з покращеними пристроями.

Рівень сенсорів охоплює фізичні пристрої – датчики температури, тиску, вібрацій, GPS-модулі, камери тощо. Ці пристрої генерують потоки даних із різною частотою і передають їх на найближчі вузли обробки.

Рівень шлюзів (edge та fog) відповідає за попередню фільтрацію, агрегацію та узгодження даних. Fog-рівень розташований між периферією та хмарою і виконує роль проміжного шару, що зменшує навантаження на центральну аналітику

Рівень аналітики реалізується в хмарному середовищі та виконує високорівневий аналіз — прогнозування, кореляцію потоків, побудову моделей деградації чи залишкового ресурсу (RUL). Тут розгортаються сервіси машинного та глибокого навчання.

Рівень сервісів забезпечує візуалізацію, сповіщення й інтеграцію з корпоративними системами (ERP/SCADA). Він дозволяє приймати рішення на основі прогнозів, формувати графіки обслуговування та замовляти запасні частини.

Дуже важливим є момент кооперації між рівнями (рисунок 2.2).

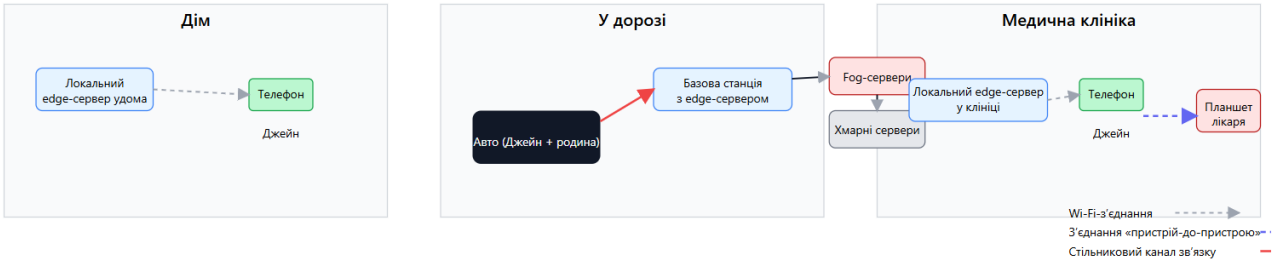


Рисунок 2.2 – Сценарій співпраці між вузлами ресурсів на периферійному, туманному та хмарному рівнях.

В такій моделі пристрої користувачів можуть виступати додатковими edge-ресурсами, утворюючи колективні вузли, які динамічно співпрацюють з fog та cloud-серверами. Така взаємодія забезпечує енергозбереження до 80 % і прискорення обчислень до 7,5 разів

2.1.1 Потоки даних і механізм збору подій

IoT-система моніторингу генерує безперервні потоки даних з сенсорів, які повинні бути своєчасно доставлені в аналітичний шар. Ключовими механізмами є потокова передача даних та моделі публікації-підписки.

На рівні edge та fog застосовується локальна обробка з використанням брокера повідомлень MQTT, який забезпечує надійне доставлення подій і керування чергами

Оптимальна передача досягається завдяки динамічному перенесенню завдань та адаптивному розподілу потоків між рівнями. Якщо з'єднання із хмарою нестабільне або дороге, більшість обчислень виконується на місцевих вузлах, а при наявності широкосмугового каналу навантаження переміщується на fog чи cloud.

Типовий конвеєр даних для аналітичного шару включає (рисунок 2.3):

– Збір подій коли дані надходять через брокери повідомлень.

– Буферизацію та агрегацію на fog-вузлах.

– Аналітику, яка виявляє аномалії та тренди в реальному часі.

– Зберігання в базах даних часових рядів та подальший високорівневий аналіз у хмарі.

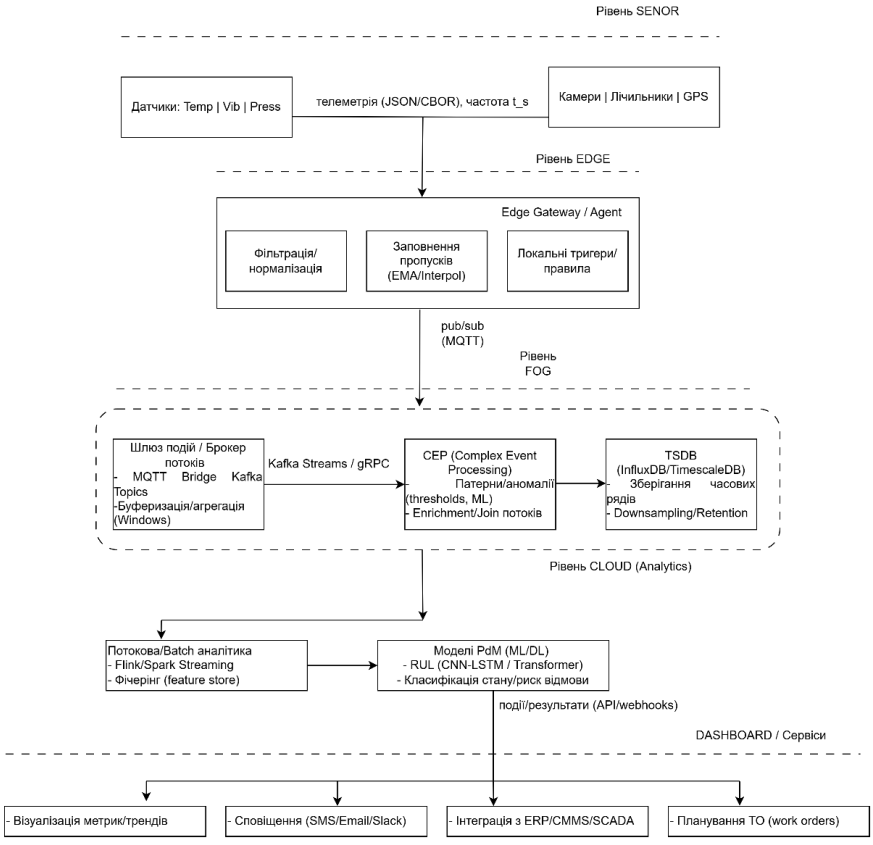


Рисунок 2.3 – Механізм збору та обробки потоків подій у багаторівневій IoT-архітектурі

Завдяки інтеграції механізмів edge-fog-cloud, аналітичний шар IoT-системи отримує здатність масштабуватися, працювати з реальним часом і мінімізувати витрати ресурсів. Використання описаних рівнів та потокових механізмів забезпечує базу для розроблення моделі прогнозного обслуговування.

2.2 Пропонований метод прогнозування обслуговування

Для вирішення даної задачі та згідно концепції була обрана комбінація кількох алгоритмів, де ядром буде нейронна мережа типу LSTM [44] (рисунок 2.4). Сенсорні дані є послідовними вимірюваннями в часі, тому важливо не просто детектувати окремі значення, а враховувати, як змінювався сигнал раніше. LSTM має особливість запам’ятовувати попередні стани, відсікати зайвий шум і виділяти довгострокові закономірності, що дозволяє детектувати повільну деградацію вузлів, зміну трендів та перехід з нормального режиму у проблемний.

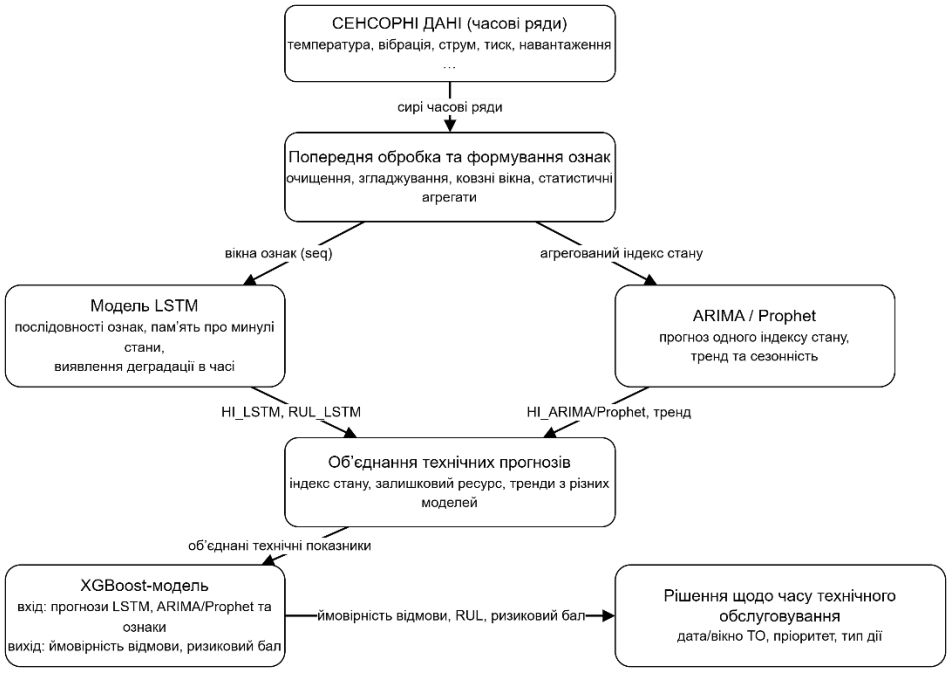


Рисунок 2.4 –Блок-схема запропонованого методу прогнозування обслуговування

Паралельно використовуються класичні моделі часових рядів, такі як ARIMA. Вони добре підходять для прогнозування одного узагальненого індексу стану, показують тренд, сезонність і дають інтерпретовані результати. Це необхідно для того, щоб визначати чому система детектує, що стан погіршується. Такі моделі виконують роль еталонної бази для порівняння з глибокою нейронною мережею.

На завершальному етапі використовується алгоритм XGBoost, як модель для оцінювання ризику відмови. На його вхід подаються прогнози LSTM, результати ARIMA та додаткові ознаки, такі як температура, вібрація, індекси завантаження. XGBoost перетворює ці дані на ймовірність відмови у заданому часовому горизонті та на ризиковий бал. З цього випливає, що поєднання LSTM, ARIMA та XGBoost дає змогу одночасно враховувати складну часову динаміку, мати інтерпретовані тренди і отримувати зрозумілу для прийняття рішень оцінку ризику.

Від того, наскільки правильно сформовані ці ознаки, напряму залежить якість моделі прогнозування.

Спочатку сирі вимірювання з різних сенсорів приводяться до єдиного часу, тобто вирівнюються часові мітки та отримується рівномірний часовий ряд. Потім дані очищуються від шуму за допомогою згладжування ковзним середнім. Якщо є послідовність значень $x1,x2,\ldots,xN$ у вікні, середнє значення обчислюється як

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i, \tag{2.1}$$

а дисперсія й стандартне відхилення – як

$$\sigma = \sqrt{\frac{1}{N}\sum_{i-1}^{N}(x_i - \mu)^2}, \tag{2.2}$$

Такі характеристики дають оцінку рівня та розкиду сигналу в кожному часовому фрагменті. Далі часовий ряд розбивається на ковзні вікна фіксованої тривалості. Для кожного вікна обчислюються такі статистичні ознаки, як

мінімум, максимум, медіана, квантилі Q_{10},Q_{90} , коефіцієнт варіації $CV=\sigma/\mu$. Крім цього визначається тренд в середині вікна. Для цього значення параметра апроксимується лінійною моделлю

$$x(t)\approx at+b, \tag{2.3}$$

де коефіцієнт a показує, чи зростає, чи спадає параметр в часі. Позитивні значення a можуть свідчити про поступове перегрівання або зростання вібрації.

Для вібраційних і акустичних сигналів додаються частотні ознаки. За допомогою швидкого перетворення Фур'є отримується спектр сигналу $X(f)$, і можна оцінити енергію в окремих діапазонах частот

$$E_{|f_1,f_2|} = \sum_{f-f_1}^{f_2} |X(f)|^2 \tag{2.4}$$

та виділяються домінантні частоти і їх гармоніки, пов'язані, з зношуванням підшипників.

Окремо формуються ознаки на основі технологічних параметрів процесу, до якого входять навантаження, витрати, продуктивності. На їх основі розраховуються інтегральні індекси, в даному випадку індекс завантаження

$$LI = \frac{P_{\text{поточне}}}{P_{\text{номінальне}}}, \tag{2.5}$$

індекс ефективності

$$EI = \frac{P_{\text{корисне}}}{P_{\text{спожите}}}, \tag{2.6}$$

а також індекс «здоров'я» агрегату, який може бути лінійною комбінацією нормалізованих параметрів:

$$HI=w_1z_1+w_2z_2+\cdots+w_kz_k,, \tag{2.7}$$

де zi – нормалізовані значення ключових показників, як температура, вібрація, w_i – їхні вагові коефіцієнти, струм.

Далі вибрано HI, який може виступати цільовою змінною для моделей типу ARIMA або додатковою ознакою для LSTM і XGBoost.

До цього вектора також додаються контекстні ознаки такі, як тип і модель обладнання, режим роботи який представляє собою номінальний, перехідний, тестовий режими, час від останнього ремонту, температуру, вологість та навколишнього середовища (рисунок 2.5). Вони дозволяють моделі відрізняти нормальну зміну параметрів через зміну режиму від справжньої деградації.

Частина всіх перетворень таких як агрегація, розрахунок простих статистик, індексів може виконуватись на периферійних вузлах (edge), що зменшує обсяг переданих в хмару даних і робить систему масштабованою та енергоефективною.

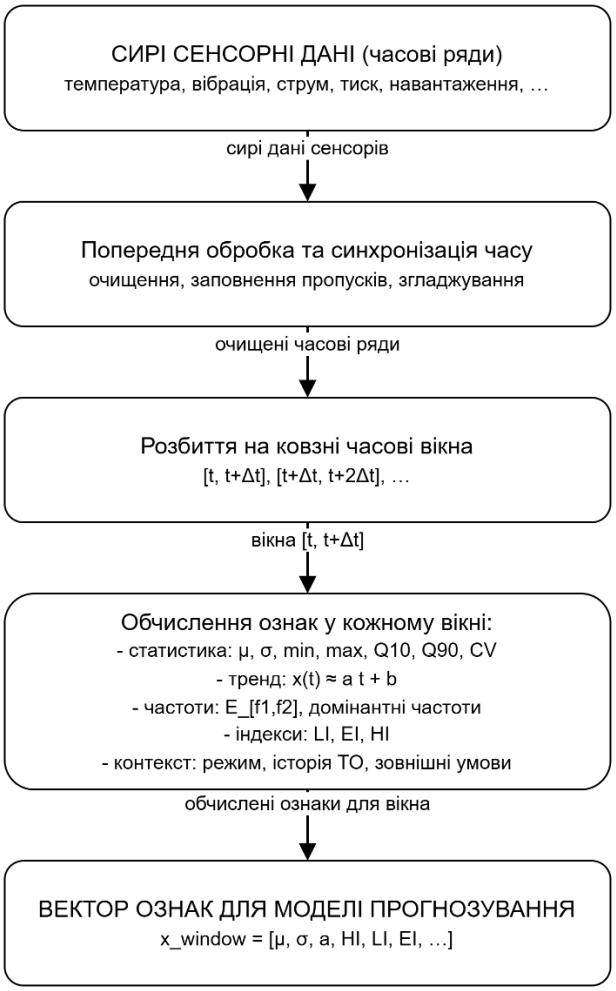


Рисунок 2.5 - Блок-діаграма потоку даних і прийняття рішення

Далі представлено опис основних функціональних блоків пропонованого методу.

У першому блоці визначається функція BuildFeatures, яка приймає два основні аргументи:

– raw_sensor_data – сирі вимірювання з сенсорів (часова мітка + параметри: температура, вібрація, струм, тиск, навантаження тощо);

– context_info – контекст: тип і модель обладнання, інформація про останнє обслуговування, температура й вологість навколишнього середовища, режим роботи.

Завдання функції – перетворити ці дані на набір векторів ознак, готових для подальшого прогнозування (рисунок 2.6).

```
data_sorted = SORT_BY_TIMESTAMP(raw_sensor_data)
data_resampled = RESAMPLE_TO_FIXED_INTERVAL(data_sorted, Δt)
    # наприклад, Δt = 1 сек чи 1 хв
```

Рисунок 2.6 – Сортування та синхронізація

На початку всі записи впорядковуються за часом (SORT_BY_TIMESTAMP). Потім робиться виконується повторна вибірка до фіксованого кроку часу Δt: наприклад, кожну секунду або кожну хвилину (рисунок 2.7).

```
data_filled = FILL_MISSING(data_resampled, method="interpolate_or_forward_fill")
```

Рисунок 2.7 – Повторна вибірка

Це потрібно, щоб мати рівномірний часовий ряд для подальших розрахунків, щоб не було прогалин чи нерівномірної дискретизації.

На цьому кроці заповнюються відсутні значення:

– інтерполяцією між сусідніми точками;

– або методом тиражування останнього відомого значення.

Кожен сенсорний канал згладжується або ковзним середнім, фільтром Баттерворта або іншим цифровим фільтром (рисунок 2.8).

```
FOR кожен канал sensor IN [temperature, vibration, current, pressure, load, ...]:
    data_smoothed[sensor] = SMOOTH(data_filled[sensor], method="moving_average" або "Butterworth")
```

Рисунок 2.8 –Згладження

Основною задачею є прибрати високочастотний шум, зберігши ключові тренди і зміни, які пов’язані з деградацією, а не з випадковими коливаннями.

Далі йде розбиття на ковзні вікна. (рисунок 2.9).

```
feature_vectors = ПУСТИЙ_СПИСОК
windows = SLIDING_WINDOWS(data_smoothed, window_size=WIN_SIZE, step=STEP)
```

Рисунок 2.9 – Розбиття на ковзні вікна

Тут формується набір ковзних вікон:

– WIN_SIZE – тривалість вікна 10–60 хвилин;

– STEP – крок зміщення вікна.

Кожне вікно – це часовий фрагмент функціонування обладнання, для якого далі обчислюються ознаки, такі як feature_vectors – список, куди складатимуться результати для кожного вікна.

Далі визначаються статистичні ознаки для кожного сенсора (рисунок 2.10).

```
ДЛЯ кожного вікна window У windows:
    features = ПУСТИЙ_СЛОВНИК

    ДЛЯ кожен канал sensor IN [temperature, vibration, current, pressure, load, ...]:
        values = GET_VALUES(window, sensor)

        μ = MEAN(values)
        σ = STD(values)
        min_val = MIN(values)
        max_val = MAX(values)
        median_val = MEDIAN(values)
        Q10 = QUANTILE(values, 0.10)
        Q90 = QUANTILE(values, 0.90)
        CV = σ / (μ + EPS)
```

Рисунок 2.10 – Визначення статистичних ознак для кожного сенсора

В цьому блоці для кожного сенсорного каналу в межах одного вікна обчислюються основні статистики:

– середнє, стандартне відхилення, мінімум, максимум;

– медіана;

– квантилі 10% та 90%;

– коефіцієнт варіації CV = σ / μ.

Ці ознаки характеризують рівень, розкид та розподіл значень параметра в часовому фрагменті.

Далі визначаються ознаки важливі для виявлення поступового відхилення параметра в небезпечний діапазон (рисунок 2.11).

```
    a, b = LINEAR_REGRESSION(window.time, values)
threshold = GET_THRESHOLD(sensor)
exceed_count = COUNT(values > threshold)
features[sensor + "_trend_a"] = a
features[sensor + "_exceed_count"] = exceed_count
```

Рисунок 2.11 – Виявлення поступового відхилення

– LINEAR_REGRESSION обчислює тренд у вікні: параметр *a* у моделі $x(t) \approx a\,t + b$ показує, чи зростає параметр (позитивний a), чи спадає;

– exceed_count рахує, скільки разів значення перевищували технологічний поріг, наприклад, граничну температуру або допустиму вібрацію.

На наступному кроці (рисунок 2.12) визначаються кореляції між парами каналів такими як струм і температура:

```
1  pairs = [(current, temperature), (current, vibration), (temperature, vibration)]
2  ДЛЯ (s1, s2) У pairs:
3      corr = CORRELATION(GET_VALUES(window, s1), GET_VALUES(window, s2))
4      features["corr_" + s1 + "_" + s2] = corr
```

Рисунок 2.12 –Визначення кореляції між парами каналів

– якщо струм зростає, а температура завжди росте – може бути нормальна поведінка;

– якщо кореляція змінюється – це може сигналізувати про аномалію.

Кореляції допомагають моделі відслідковувати взаємозв’язки між параметрами.

Далі можна визначити частотні та спектральні ознаки (рисунок 2.13).

```
    vib_values = GET_VALUES(window, "vibration")
    spectrum = FFT(vib_values)
    ДЛЯ кожен діапазон (f1, f2) У PREDEFINED_BANDS:
        energy = SPECTRAL_ENERGY(spectrum, f1, f2)
        features["vib_energy_" + STR(f1) + "_" + STR(f2)] = energy

    dominant_freq = DOMINANT_FREQUENCY(spectrum)
    spectral_entropy = SPECTRAL_ENTROPY(spectrum)
    features["vib_dominant_freq"] = dominant_freq
    features["vib_spectral_entropy"] = spectral_entropy
```

Рисунок 2.13 – Визначення частотних та спектральних ознак

Цей блок дає змогу обчислити вібраційні сигнали:

– FFT обчислює спектр;

– SPECTRAL_ENERGY дає енергію в заданих частотних діапазонах;

– DOMINANT_FREQUENCY – основна (найсильніша) частота;

– SPECTRAL_ENTROPY – «хаотичність» спектра.

На наступному кроці формується Індекси процесу: LI, EI, HI

Тут вводяться інтегральні індекси:

– LI – індекс завантаження, тобто наскільки поточне навантаження близьке до номінального;

– EI – індекс ефективності, який показує співвідношення корисної та спожитої потужності;

– HI – індекс здоров’я, побудований як зважена сума нормалізованих параметрів температури, вібрації та струму.

HI може використовуватися і як окрема ознака, і як цільова змінна для моделей прогнозування (рисунок 2.14).

```
    load_values = GET_VALUES(window, "load")
    power_input = ESTIMATE_INPUT_POWER(window)      # при потребі
    power_output = ESTIMATE_OUTPUT_POWER(window)    # при потребі

    LI = MEAN(load_values) / (NOMINAL_LOAD + EPS)
    EI = power_output / (power_input + EPS)
```

Рисунок 2.14 – Формування індексів процесу

Далі визначаються контекстні ознаки (рисунок 2.15).

```
95          features["equipment_type"] = ENCODE_CATEGORY(context_info.type)
96          features["equipment_model"] = ENCODE_CATEGORY(context_info.model)
97          features["mode"] = ENCODE_CATEGORY(context_info.mode)  # номінальний / перехідний / тестовий
98          features["time_since_last_maintenance"] =
99              TIME_DIFF(window.end_time, context_info.last_maintenance_time)
100         features["ambient_temperature"] = context_info.ambient_temperature
101         features["ambient_humidity"] = context_info.ambient_humidity
```

Рисунок 2.15 – Визначення контекстних ознак

Тут додаються не сенсорні, а технологічні ознаки, зазначені в документації виробників та зовнішні ознаки такі як тип і модель обладнання, режим роботи, час від останнього технічного обслуговування, температура і вологість середовища.

Це дозволяє моделі відрізнити нормальні зміни через зміну режиму від справжніх ознак зносу.

На останньому етапі визначається мітка часу вікна та збереження векторів (рисунок 2.16).

```
103         # 4.6. Додавання мітки часу для вікна (наприклад, середина)
104         features["window_timestamp"] = MIDPOINT(window.start_time, window.end_time)
105
106         # 4.7. Збереження вектора ознак
107         APPEND(feature_vectors, features)
108
109     ПОВЕРНУТИ feature_vectors
```

Рисунок 2.16 – Визначення мітки часу вікна та збереження векторів

Де:

– window_timestamp – центральний час для вікна, який використовується для подальшого вирівнювання з мітками відмов та планів технічного обслуговування;

– feature_vectors заповнюється по одному словнику features для кожного вікна;

У фіналі функція повертає повний набір векторів ознак для всього періоду.

2.3 Математична модель прогнозування та формалізація задачі оптимізації

Для побудови математичної моделі спочатку будується модель прогнозу на основі часових рядів та ознак, сформованих у попередньому розділі. Далі визначаються критерії якості прогнозу та метрики, за якими порівнюються різні

варіанти моделей. Після цього формалізується інтеграція прогнозної моделі в систему прийняття рішень з врахуванням ризику відмови та вартості рішень.

Нехай для кожного моменту часу t є вектор ознак який містить статистичні, спектральні, процесні та контекстні ознаки.

$$x_t \in R^d, \tag{2.8}$$

Цільова змінна може задаватися у двох варіантах:

– як неперервна величина – індекс стану або залишковий ресурс

$$y_t \in R(HI), \tag{2.9}$$

– або як ймовірність відмови в горизонті H:

$$y_t \in \{0,1\}, y_t = 1 \Rightarrow \text{відмова в інтервалі } [t, t+H]. \tag{2.10}$$

Модель прогнозування можна позначити, як функцію

$$y_t = f_\theta(x_{t-L+1} \cdots, x_t) \tag{2.11}$$

де θ – параметри моделі: ваги LSTM; коефіцієнти ARIMA; параметри XGBoost; а на вхід подається послідовність векторів ознак за останні L кроків. Для LSTM це послідовність в часі, для ARIMA– окремий індекс стану, для XGBoost – агрегований вектор ознак у момент часу t

Навчання моделі формулюється, як задача мінімізації середньої похибки прогнозу на історичних даних. Нехай N навчальних прикладів (X_i, y_i) де X_i– відповідна послідовність ознак. Тоді загальний функціонал похибки записується як

$$J(\theta) = \frac{1}{N} \sum_{i-1}^{N} \mathcal{L}\big(y_i f_\theta(X_i)\big), \tag{2.12}$$

де L– функція втрат - квадратична похибка або абсолютна похибка.

Для задач регресії та прогнозування індексу стану або RUL часто використовується квадратична функція втрат:

$$\mathcal{L}_{MSE}(y, \hat{y}) = (y - \hat{y})^2 \tag{2.13}$$

Для класифікації ризику відмови використано логістичну крос-ентропію:

$$\mathcal{L}_{CE}(y, \hat{p}) = -\big(ylog\hat{p} + (1 - y)log(1 - \hat{p})\big), \tag{2.14}$$

де $\hat{p} = f_\theta(X)$- прогнозована ймовірність відмови.

Задача навчання моделі формулюється як

$$\theta^* \arg\min_\theta J(\theta) \tag{2.15}$$

Далі при побудові та порівнянні моделей використовуються кілька метрик якості. Вони оцінюють, наскільки модель відтворює фактичні значення цільової змінної та допомагають обрати найкращу архітектуру і налаштування.

Для регресійних задач прогнозу, індексу стану HI використовуються:

Середньоквадратична помилка (MSE)

$$MSE = \frac{1}{N}\sum_{i-1}^{N}(y_i - \widehat{y_i})^2 \tag{2.16}$$

Корінь із середньоквадратичної помилки (RMSE)

$$RMSE = \sqrt{MSE} = \frac{1}{N}\sum_{i-1}^{N}|y_i - \widehat{y_i}|, \tag{2.17}$$

Вона менш чутлива до поодиноких великих помилок, тому корисна, коли важливі стабільні невеликі відхилення.

Середня абсолютна відносна помилка (MAPE)

$$MAPE = \frac{100\%}{N}\sum_{i-1}^{N}\left|\frac{y_i-\hat{y}_i}{y_i}\right|, \quad (2.18)$$

MAPE показує похибку в процентах і зручна для інтерпретації, але потребує обережності при малих y_i

Для класифікаційної задачі оцінки ризику відмови застосовуються:

– Accuracy, Precision, Recall, F1 – для загальної діагностики моделі;

– Precision@K – частка реальних відмов серед K об'єктів із найвищим прогнозованим ризиком.

Метрика Precision@K важлива для PdM, коли ресурси на обслуговування обмежені і потрібно правильно обрати групу найбільш ризикове обладнання:

$$Precision@K = \frac{\text{кількість справжніх відмов серед топ}-K\text{ об'єктів}}{K}, \quad (2.19)$$

У навчанні можна мінімізувати один функціонал, наприклад, MSE, але при виборі фінальної моделі враховувати також інші метрики, зокрема Precision@K для ризикових об'єктів. Це фактично перетворюється на багатокритеріальну оптимізацію, де баланс між RMSE, MAE, MAPE та показниками ризику визначається вимогами конкретного підприємства.

Для остаточного етапу потрібне прийняття оптимального рішення щодо часу технічного обслуговування. Для цього вводиться стохастична модель часу відмови та вартісний критерій.

Нехай T_f – випадковий момент відмови обладнання, а τ– момент, коли виконується планове технічне обслуговування.

Якщо обслуговування виконати в момент τ, витрати становлять які включають вартість робіт, запчастин і плановий простій.

$$C_{maint}(\tau), \quad (2.20)$$

Якщо обслуговування відкласти, то існує ризик, що відмова відбудеться в інтервалі $[t,t+H]$. Модель прогнозує ймовірність

$$P\left(T_f \leq t + H | X_t\right) = \hat{p}_t \tag{2.21}$$

а у випадку відмови понесені витрати дорівнюють C_{fail} куди входять аварійний простій, додаткові пошкодження, можливі штрафи та ризики безпеки.

Тоді очікувана вартість сценарію відкладення обслуговування на горизонті H можна записати як

$$\mathbb{E}\left[C_{delay}\right] \approx \hat{p}_i \cdot C_{\text{fail}} \tag{2.22}$$

Задача вибору оптимального моменту обслуговування формулюється, як мінімізація сумарних очікуваних витрат:

$$\tau^* = \arg\min_{\tau}(C_{maint}(\tau) + \mathbb{E}[C_{\text{fail}}(\tau)]), \tag{2.23}$$

де $E[C_{\text{fail}}(\tau)]$ залежить від прогнозованого ризику відмови у відповідні часові інтервали, який формує модель.

На практиці рішення приймається за нескладним правилом в момент часу t:

якщо

$$\hat{p}_i \cdot C_{\text{fail}} > C_{maint}(t), \tag{2.24}$$

то доцільно виконати обслуговування зараз. Інакше можна відкласти обслуговування та переглянути рішення на наступному кроці моніторингу.

Отже модель прогнозування, яка мінімізує функціонал похибки $J(\theta)$ на основі RMSE/MAE/MAPE, безпосередньо інтегрується у систему підтримки

прийняття рішень, де додатковим критерієм стає очікувана економічна вигода від правильного вибору моменту технічного обслуговування.

На рисунку 2.17 представлено узагальнену схему інтеграції моделі прогнозування технічного стану обладнання в контур прийняття рішень щодо технічного обслуговування.

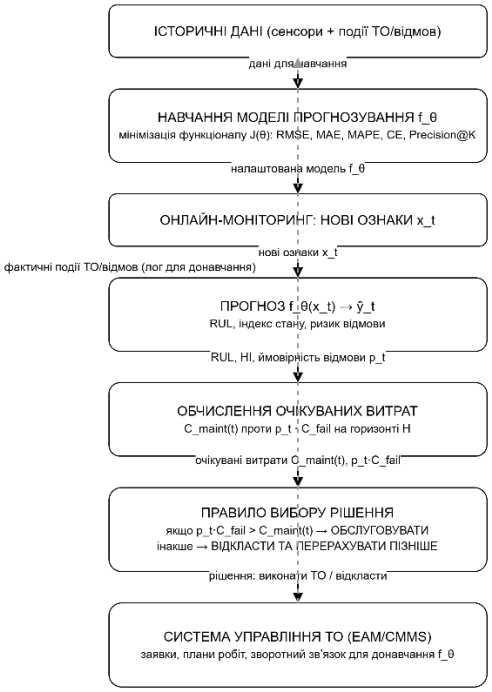


Рисунок 2.17 – Блок-схема інтеграції моделі прогнозування в контур прийняття рішень щодо технічного обслуговування на основі очікуваних витрат

Вона демонструє послідовний перехід від історичних сенсорних даних та подій обслуговування до навчання моделі, подальшого онлайн-прогнозування стану та розрахунку очікуваних витрат для різних сценаріїв.

Спочатку історичні дані використовуються для побудови та налаштування моделі шляхом мінімізації функціоналу похибки за метриками RMSE, MAE, MAPE та показниками якості класифікації ризику. Далі, в режимі онлайн-моніторингу, нові ознаки x_t подаються на вхід моделі, яка формує прогноз $\hat{y}_t$та оцінку ймовірності відмови. На основі цієї оцінки обчислюються очікувані витрати сценаріїв «обслуговувати зараз» та «відкласти».

Далі рішення передається до системи управління технічним обслуговуванням де автоматично формуються заявки, плануються роботи та

накопичується зворотний зв’язок для подальшого донавчання моделі. Тобто по суті це замкнений контур PdM: від збору даних і прогнозу – до економічно обґрунтованого управлінського рішення.

Висновки до розділу 2

1. Сформовано концепцію аналітичного шару IoT-системи моніторингу, що базується на архітектурі Edge–Fog–Cloud. Рівні сенсорів, шлюзів, хмарної аналітики та сервісів розмежовують функції збору, попередньої обробки, високорівневого аналізу та прийняття рішень, що дає змогу зменшити затримки, трафік у хмару та підвищити стійкість роботи системи.

2. Описано механізм потокового збору подій: використання брокера MQTT, буферизації та агрегації на fog-вузлах, зберігання в базах часових рядів і подальший аналіз у хмарі. Такий підхід забезпечує надійне доставлення телеметрії, масштабованість і можливість гнучко перерозподіляти обчислення залежно від якості каналу зв’язку та вартості ресурсів.

3. Запропоновано метод прогнозування обслуговування, який поєднує глибоку рекурентну мережу LSTM, класичні моделі часових рядів (ARIMA/Prophet) та градієнтний бустинг XGBoost для прийняття рішень у PdM-сценаріях.

4. Розроблено формалізований підхід до побудови ознак вирівнювання часових міток, повторної вибірки, згладжування, розбиття на ковзні вікна, розрахунку статистичних, спектральних, процесних та контекстних показників.

5. Сформульовано математичну модель прогнозування що використовує послідовність векторів ознак, та задачу навчання як мінімізацію функціоналу похибки з використанням метрик MSE, RMSE, MAE, MAPE для оцінки ризику відмови.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ ПРОГНОЗУВАННЯ ОБСЛУГОВУВАННЯ

3.1 Модуль збору та попередньої обробки IoT-даних

Модуль збору та попередньої обробки даних є основним початковим модулем запропонованого методу прогнозування обслуговування в системах моніторингу на базі IoT (рисунок 3.1). Його завдання – прийняти потоки телеметрії з сенсорів в реальному часі, привести їх до єдиного формату, очистити від шуму та пропусків, а далі зберегти у сховище часових рядів для подальшого аналізу та прогнозування.

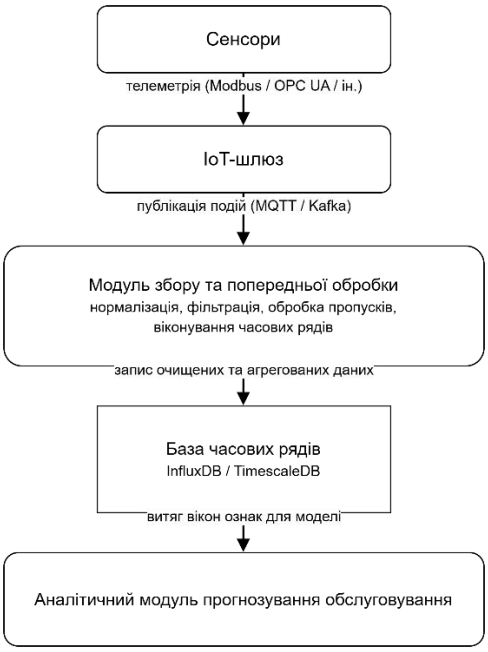


Рисунок 3.1 – Блок-схема потоку даних від сенсорів до аналітичного модуля прогнозування обслуговування в IoT-системі

На нижньому рівні сенсори такі як температура, вібрація, струм, тиск, навантаження передають дані на IoT-шлюз, використовуючи полегшені протоколи типу MQTT. На шлюзі дані перетворюються у єдиний формат JSON і публікуються в брокер повідомлень MQTT.

Нижче наведено приклад Python-клієнта (рисунок 3.2), який підписується на MQTT-тему з телеметрією та передає отримані повідомлення в модуль попередньої обробки.

```python
import json
import paho.mqtt.client as mqtt
from queue import Queue
raw_data_queue = Queue()
def on_message(client, userdata, msg):
    payload = json.loads(msg.payload.decode("utf-8"))
    raw_data_queue.put(payload)
client = mqtt.Client(client_id="preproc-service")
client.on_message = on_message
client.connect("mqtt-broker.local", port=1883)
client.subscribe("factory/line1/telemetry/#")
client.loop_start()
```

Рисунок 3.2 – Python-клієнт для підписки на MQTT-тему

Отримані повідомлення потрапляють в модуль попередньої обробки. Далі відбувається нормалізація, фільтрація та обробка пропусків. Основна ідея – перетворити нерівномірний, зашумлений потік показників на чистий, рівномірно дискретизований часовий ряд.

Спочатку дані сортуються за часовими мітками і приводяться до єдиного інтервалу дискретизації. Потім застосовується згладжування та заповнення пропусків. Приклад обробки у вигляді показано нижче на рисунку 3.3.

```python
import pandas as pd
def preprocess_timeseries(df: pd.DataFrame, freq="1s"):
    """
    df очікується у форматі:
    ts (datetime), device_id, temperature, vibration, current, ...
    """
    df = df.sort_values("ts")
    df = df.set_index("ts").resample(freq).mean()
    df = df.ffill().interpolate(method="time")
    return df.reset_index()
```

Рисунок 3.3 – Згладжування та заповнення пропусків

Після дискретизації дані все ще можуть містити випадкові коливання, які заважають моделі бачити реальні тенденції деградації. Тому застосовується фільтрація – ковзне середнє (рисунок 3.4).

```python
def smooth_column(series: pd.Series, window: int = 5) -> pd.Series:
    """
    Проста фільтрація ковзним середнім.
    window = 5 кожна точка замінюється
    середнім значенням по 5 сусідніх точках.
    """
    return series.rolling(window=window, min_periods=1, center=True).mean()

def smooth_dataframe(df: pd.DataFrame, sensor_cols):
    """
    Згладжуються лише сенсорні колонки (температура, вібрація, струм).
    """
    df_smoothed = df.copy()
    for col in sensor_cols:
        df_smoothed[col] = smooth_column(df[col])
    return df_smoothed
```

Рисунок 3.4 – Фільтрація

На наступному етапі оброблений часовий ряд потрібно зберегти в базі, що підтримує ефективні операції з часовими даними. Для цього застосовано InfluxDB, в ролі сховища документів з часовою ознакою. Нижче на рисунку 3.5) наведений фрагмент коду запису до InfluxDB в форматі line protocol.

```python
from influxdb_client import InfluxDBClient, Point, WriteOptions
client = InfluxDBClient(
    url="http://influxdb:8086",
    token="MY_TOKEN",
    org="my-org"
)
write_api = client.write_api(write_options=WriteOptions(batch_size=1000))
def write_to_influx(df, bucket: str = "iot_clean"):
    for _, row in df.iterrows():
        p = (
            Point("telemetry_clean")
            .tag("device_id", row["device_id"])
            .field("temperature", float(row["temperature"]))
            .field("vibration", float(row["vibration"]))
            .field("current", float(row["current"]))
            .time(row["ts"])  #
        )
        write_api.write(bucket=bucket, record=p)
```

Рисунок 3.5 – Запис до БД

Після збереження в очищеному вигляді модуль може формувати ковзні вікна, які будуть далі використані для формування ознак і навчання моделі. Спрощений приклад функції, яка генерує вікна, наведено нижче на рисунку 3.6.

```python
import numpy as np
def generate_windows(df: pd.DataFrame, window_size: int, step: int, sensor_cols):
    windows = []
    timestamps = df["ts"].values
    for start in range(0, len(df) - window_size + 1, step):
        end = start + window_size
        window_data = df.iloc[start:end]
        # забираємо тільки сенсорні колонки
        X = window_data[sensor_cols].to_numpy(dtype=float)
        ts_mid = timestamps[start + window_size // 2]  # час середини вікна
        windows.append({"ts_mid": ts_mid, "X": X})
    return windows
```

Рисунок 3.6 – Функція генерації вікна

Отже модуль збору та попередньої обробки IoT-даних реалізує повний цикл: від отримання сирих потоків телеметрії до формування багатовимірних, очищених та узгоджених в часі фрагментів часових рядів, які потім передаються в аналітичний модуль прогнозування та обслуговування.

3.2 Реалізація прогнозної моделі та аналітичного блоку

Аналітичний блок є основою системи прогнозування обслуговування в IoT-моніторингу. Якщо модуль збору та попередньої обробки IoT-даних відповідає за збір та підготовку часових рядів, то блок прогнозної моделі та аналітичного блоку перетворює ці дані на конкретний прогноз, наприклад скільки ще пропрацює обладнання, з яким ризиком може статися відмова та коли доцільно планувати ТО. Такий підхід відповідає сучасним PdM-системам, де модель навчається на історичних даних і далі використовується в онлайн режимі для підтримки рішень щодо обслуговування.

В пропонованому рішенні навчання здійснюється на історичних даних з датчиків, попередньо оброблених модулем збору попередніх даних. Для кожного ковзного вікна часових рядів сформовано вектор ознак, а цільова змінна задається у вигляді залишкового ресурсу (RUL) або індексу здоров'я обладнання. Для моделювання часових залежностей обрано рекурентну нейронну мережу типу LSTM, тому що такі моделі добре зарекомендували себе для оцінки RUL на основі послідовностей сенсорних вимірювань у PdM-задачах.

Навчальний конвеєр включає кілька кроків: завантаження історичних даних з бази часових рядів, формування вікон, розбиття на train/validation,

визначення архітектури LSTM-мережі та оптимізацію ваг за критерієм MSE/RMSE. Нижче наведено сфрагмент коду на, який ілюструє структуру моделі й тренувальний цикл.

На рисунку 3.7 представлено структуру моделі у вигляді класу HealthLSTM, який наслідує базовий клас nn.Module.

```
class HealthLSTM(nn.Module):
    def __init__(self, input_dim, hidden_dim=64, num_layers=2):
        super().__init__()
        self.lstm = nn.LSTM(input_dim, hidden_dim,
                            num_layers=num_layers,
                            batch_first=True)
        self.fc = nn.Linear(hidden_dim, 1)  # прогноз HI або RUL
```

Рисунок 3.7 – Архітектура LSTM-мережі

Конструктор приймає параметри розмірності вхідних ознак (input_dim), розміру прихованого шару (hidden_dim) та кількості шарів LSTM (num_layers). В середині конструктора створюється шар nn.LSTM, який обробляє послідовності векторів ознак в часі. Параметр batch_first=True означає, що перший вимір тензора відповідає розміру пакету (batch). Після LSTM визначається вихідний лінійний шар nn.Linear, який перетворює прихований стан в одне числове значення. Цей вихід інтерпретується як прогнозований індекс технічного стану (health index) або залишковий ресурс (RUL).

Навчання моделі організоване у вигляді циклу по епохах. Історичні дані, перетворені в ковзні вікна, подаються в модель за допомогою об’єкта train_loader. Для кожного пакету спочатку обнуляються попередні градієнти, потім виконується прямий прохід моделі та обчислюється функція втрат на основі середньоквадратичної помилки (MSE). Далі виконується зворотний прохід, під час якого обчислюються градієнти по вагам, та крок оптимізації методом Adam, так модель поступово навчається мінімізувати різницю між прогнозованими значеннями індексу стану та відомими істинними значеннями з історичних даних. Після завершення навчання отримані ваги зберігаються у файл і використовуються в режимі отримання прогнозу навченою моделлю в окремому сервісі.

Після завершення навчання найкраща за валідаційною вибіркою модель зберігається у вигляді «артефакту» (файл .pt або .onnx). Цей артефакт завантажується окремим сервісом прогнозування, який працює як мікросервіс у хмарі або на edge-сервері. Сервіс отримує поточні вікна ознак (наприклад, через REST-API), проганяє їх через модель і повертає прогнозований індекс стану, очікуваний RUL та категорію ризику (низький/середній/високий).

У фрагменті коду який зображено на рисунку 3.8 представлено, як навчена модель інтегрується у сервіс прогнозування на базі FastAPI. На етапі ініціалізації створюється екземпляр тієї ж моделі HealthLSTM з відповідною розмірністю входу, після чого в неї завантажуються збережені ваги. Модель переводиться в режим оцінювання, що вимикає механізми, характерні для навчання і забезпечує стабільну роботу під час прогнозування. Далі оголошується HTTP-ендпоінт /predict, який приймає POST-запити з JSON-повідомленням. В середині запиту очікується масив з часовою послідовністю ознак для одного вікна.

```
4  model = HealthLSTM(input_dim=num_features)
5  model.load_state_dict(torch.load("health_lstm.pt"))
6  model.eval()
```

Рисунок 3.8 – Інтеграція моделі в сервіс прогнозування

В цьому фрагменті створюється екземпляр моделі HealthLSTM з тією ж архітектурою, яка використовувалась під час навчання (кількість вхідних ознак задається як num_features). Потім у модель завантажуються збережені ваги з файлу "health_lstm.pt". Цей файл є результатом попереднього етапу навчання.

Виклик model.eval() переводить модель у режим оцінювання (інференсу). У цьому режимі вимикаються механізми, характерні для навчання (наприклад, dropout), і модель стабільно працює для отримання прогнозів на нових даних.

Тут визначається HTTP-ендпоінт POST /predict. Це означає, що зовнішні системи можуть надіслати POST-запит на адресу /predict з даними для прогнозування.

Функція predict (рисунок 3.9) очікує, що в тілі запиту (об'єкт payload) буде поле "X", яке містить масив розміром [seq_len, num_features]. Це одне часове вікно з послідовністю векторів ознак.

```
@app.post("/predict")
def predict(payload: dict):
    import numpy as np
    X = np.array(payload["X"], dtype=float)
    X_tensor = torch.tensor(X).unsqueeze(0)  # [1, seq_len, num_features]
```

Рисунок 3.9 – Функція predict

Отриманий масив спочатку перетворюється в масив NumPy (np.array), а потім у тензор PyTorch (torch.tensor). Виклик unsqueeze(0) додає перший вимір, щоб сформувати тензор форми [1, seq_len, num_features], де 1 –розмір пакету. Такий формат відповідає тому, як модель очікує отримати дані.

Далі виконується прогноз (рисунок 3.10). Контекстний менеджер with torch.no_grad(): вимикає обчислення градієнтів, оскільки при інференсі вони не потрібні.

```
    with torch.no_grad():
        hi_pred = model(X_tensor).item()
    # правило ризику
    if hi_pred < 0.3:
        risk = "high"
    elif hi_pred < 0.6:
        risk = "medium"
    else:
        risk = "low"
    return {"health_index": hi_pred, "risk_level": risk}
```

Рисунок 3.10 – Прогнозування

Це зменшує витрати пам'яті і пришвидшує обчислення. Модель отримує на вхід тензор X_tensor і повертає числове значення, яке інтерпретується, як індекс технічного стану (health_index). Виклик .item() витягує це значення як звичайне число типу float.

Далі застосовується порогове правило для визначення рівня ризику:

– якщо hi_pred < 0.3, стан вважається поганим, а ризик – високим ("high");

– якщо hi_pred знаходиться між 0.3 і 0.6, обладнання відноситься до середнього ризику ("medium");

– якщо hi_pred ≥ 0.6, вважається, що стан задовільний, і ризик відмови низький ("low").

В результаті функція повертає словник, який FastAPI автоматично перетворює в JSON-відповідь. Ця відповідь містить два поля: числовий health_index та текстовий risk_level. Такий формат зручний для подальшого використання в системах моніторингу, дашбордах або у логіці прийняття рішень щодо технічного обслуговування.

Такий підхід дозволяє безпосередньо вбудовувати модель у IoT-платформу або систему моніторингу та отримувати прогнози в режимі близькому до реального часу для підтримки рішень щодо технічного обслуговування.

3.3 Інтеграція з платформою моніторингу та візуалізація результатів

В запропонованій системі [43] прогнозування обслуговування на базі IoT важливу роль відіграє платформа моніторингу, яка об’єднує збір даних, аналітику та візуалізацію для оператора або системного адміністратора. Найчастіше використовується зв’язка сенсори + шлюзи +хмара/edge + графічний інтерфейс, де окремий модуль відповідає за відображення стану обладнання та сповіщення про ризики.

Для візуалізації часових рядів сенсорних даних і результатів прогнозування доцільно застосовувати готові open-source інструменти, такі як Grafana, а також використовувати Node-RED для оркестрації потоків подій. У хмарній або edge-інфраструктурі сенсорні дані та результати моделей зберігаються в сховищах часових рядів (InfluxDB, TimescaleDB). Grafana підключається до цих джерел як «data source» і будує інтерактивні дашборди.

На рисунку 3.11 подано дашборд Grafana, який використовується для візуалізації результатів прогнозування обслуговування в IoT-системі. На верхній частині дашборду відображаються часові графіки сирих та згладжених сенсорних сигналів де представлені показники температури, вібрації та струму,

що дозволяє оператору співставляти аномалії в даних із конкретними режимами роботи обладнання.

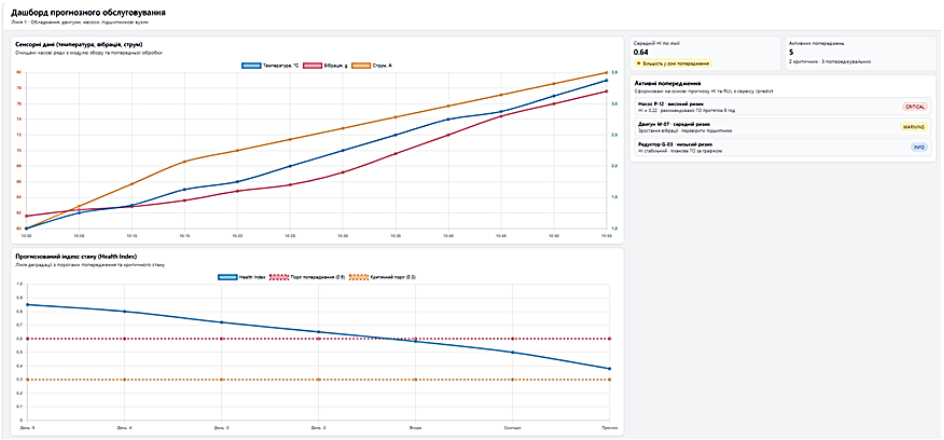


Рисунок 3.11 – Дашборд для візуалізації результатів прогнозування обслуговування в IoT-системі

Нижче наведено графік прогнозованого індексу стану (Health Index, HI) з пороговими лініями: нормальна, попереджувальна та критична зона. В правій частині дашборду розташований віджет з переліком активних попереджень: для кожної машини показується поточний HI, рівень ризику («low», «medium», «high»), час спрацювання тривоги та коротка рекомендація. Додаткові панелі можуть відображати кількість спрацювань за період, розподіл обладнання за рівнями ризику та середні значення HI по цехах або лініях.

На рисунку 3.12 показано демонстраційний потік Node-RED, який реалізує інтеграцію між потоком IoT-даних, сервісом прогнозування та системою сповіщень. У лівій частині схеми знаходиться вузол MQTT in, що підписується на тему з телеметрією (наприклад, factory/line1/telemetry).

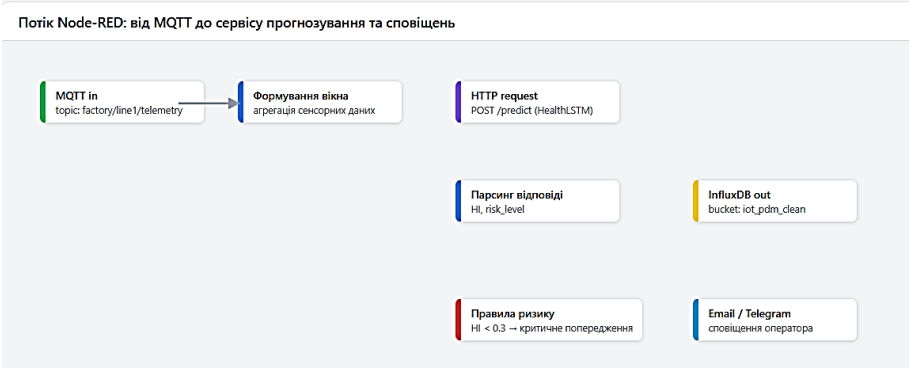


Рисунок 3.12 – Потік Node-RED

Далі йдуть функціональні вузли function, які виконують агрегацію, формування ковзних вікон і підготовку даних до запиту в сервіс /predict. Наступним елементом є вузол HTTP request, що надсилає вікно ознак до REST-API та отримує у відповідь індекс стану і рівень ризику. Результати записуються у базу часових рядів через вузол InfluxDB out, а також потрапляють на вузол switch, який реалізує прості правила сповіщення, наприклад, при високому ризику створюється окреме попередження. У правій частині потоку можуть бути налаштовані вузли email, telegram або dashboard, які відповідають за відправку повідомлень та онлайн-відображення статусу.

Окремо реалізується веб-інтерфейс адміністратора, де можна переглядати список обладнання, поточні рівні ризику, прогнозований залишковий ресурс, рекомендовані дати обслуговування, а також налаштовувати пороги сповіщень і правила.

На рисунку 3.13 наведено приклад веб-інтерфейсу, орієнтованого на інженера або системного адміністратора. Центральне місце займає таблиця з переліком обладнання: ідентифікатор, назва, індекс стану, рівень ризику (кольоровий індикатор), орієнтовний RUL та рекомендований статус («норма», «планувати ТО», «невідкладна перевірка»).

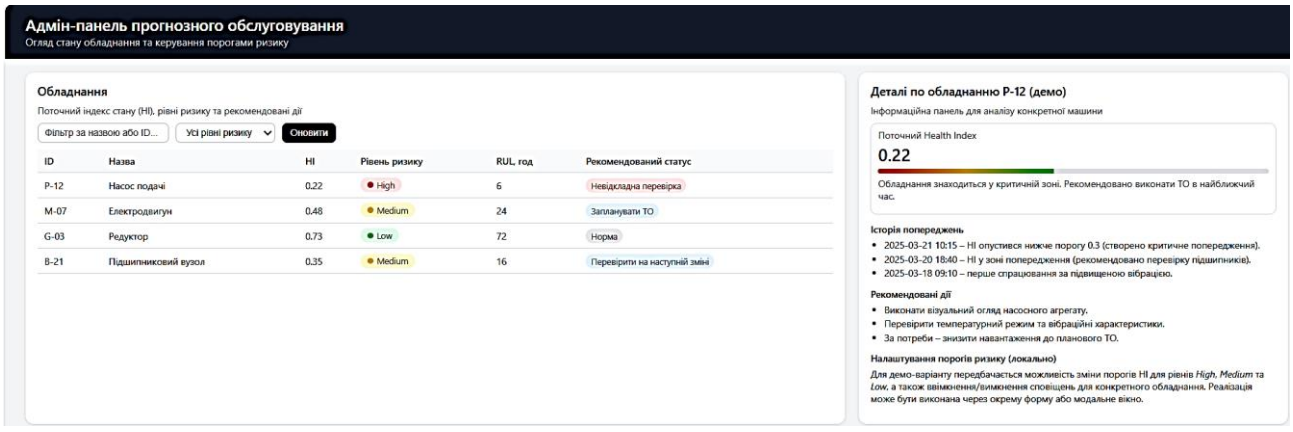


Рисунок 3.13 – Веб-інтерфейс адміністратора

У верхній частині є фільтри за цехом, типом обладнання та рівнем ризику, що дозволяє швидко знаходити критичні об’єкти. У правій панелі відображаються детальні дані по вибраному обладнанню: графіки HI та RUL,

історія попереджень, а також блок з рекомендаціями. Інтерфейс також передбачає зміну порогових значень, керування правилами сповіщень і фіксацію фактів виконаного обслуговування.

Алгоритм сповіщення працює послідовно. Спочатку сервіс прогнозування з певною періодичністю обробляє нові вікна сенсорних даних і обчислює для кожного агрегату індекс стану HI та/або оцінку залишкового ресурсу RUL. Далі застосовуються правила ризику: якщо HI < HI_critical або RUL < RUL_critical, формується критичне попередження; якщо HI менший за попереджувальний поріг, але ще не досяг критичного, створюється попередження «середній ризик».

Події попереджень записуються в таблицю alerts в базі даних і додаються як анотації до відповідних графіків. Після цього через Node-RED або вбудовані канали сповіщення Grafana попередження дублюються в зовнішні сервіси: електронну пошту, месенджери або CMMS. Щоб уникнути надмірної кількості тривог, можуть використовуватись прості механізми згладжування, наприклад, вимога кількох послідовних спрацювань.

На завершальному етапі оператор після фактичного виконання обслуговування позначає подію, як закриту та фіксує реальний стан обладнання. Ця інформація повертається в контур навчання моделі й використовується як додаткові мітки для подальшого покращення якості прогнозів.

3.4 Оцінювання ефективності методу прогнозування

Оцінювання ефективності методу прогнозування обслуговування в системах моніторингу на базі IoT виконується в декілька етапів. По-перше, модель порівнюється з базовими машинними підходами, які вже використовуються в задачах аналізу часових рядів. Далі аналізуються числові метрики точності прогнозу та часові характеристики роботи моделі. На наступному кроці метод тестується на реальному або симульованому IoT-наборі даних з максимально наближеними до виробництва сценаріями роботи обладнання.

Для об’єктивного оцінювання запропонованого методу на основі індексу стану Health Index, (HI) та LSTM-моделі формується набір базових моделей, з якими виконується порівняння. До таких базових підходів належать:

– прості моделі, наприклад, «останнє значення» або ковзне середнє, які не враховують складну динаміку процесу;

– класичні статистичні моделі часових рядів, такі як ARIMA/ARIMAX, що добре працюють для приблизно стаціонарних сигналів;

– моделі градієнтного бустингу (XGBoost), які використовують набір агрегованих і спектральних ознак;

– прості багатошарові перцептрони (MLP) без явного врахування послідовної структури даних.

Запропонована LSTM-модель використовує послідовності попередньо сформованих вікон ознак (температура, вібрація, струм, індекси стану та контекстні параметри) і здатна враховувати довгострокові залежності та патерни деградації обладнання. В процесі порівняння всі моделі навчаються на тих самих історичних даних з датчиків та оцінюються на відкладеній тестовій вибірці, що містить приклади як нормальної роботи, так і передвідмовних станів.

Точність прогнозування оцінюється за класичними метриками регресії для HI/RUL, такими, як середньоквадратична помилка (RMSE), середня абсолютна помилка (MAE) та відносна помилка (MAPE). Для задачі класифікації рівнів ризику («low», «medium», «high») додатково можуть використовуватися F1-міра, Precision та Recall для кожного класу. Окремо аналізується, наскільки рано модель здатна сигналізувати про наближення відмови.

Крім якості прогнозу, враховується продуктивність системи де включено час навчання моделі на історичних даних, час формування прогнозу для одного вікна сенсорних даних, використання оперативної пам’яті та можливість виконання навченої моделі на edge-вузлах з обмеженими ресурсами. Це використовується для практичного впровадження в реальних IoT-системах, де кількість об’єктів може бути великою, а затримки в обробці – критичними.

На рисунку 3.14 наведено WEB-інтерфейс для оцінювання моделей. В лівій частині сторінки показана таблиця з порівнянням основних метрик (RMSE, MAE, MAPE) для кількох моделей: простої, ARIMA, XGBoost і LSTM.

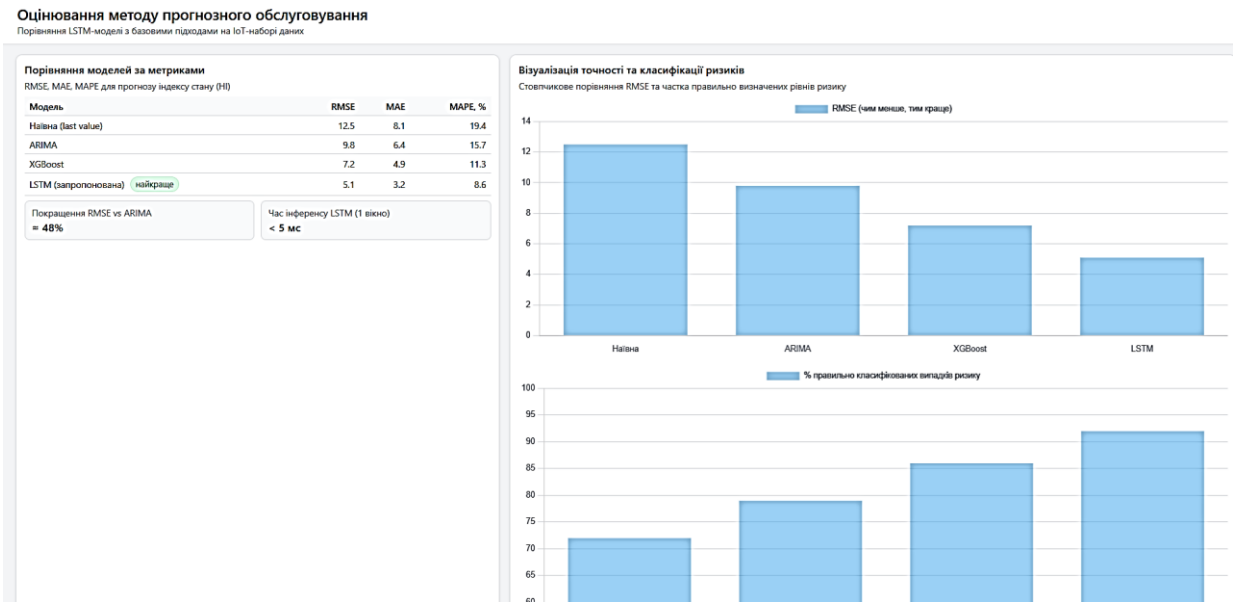


Рисунок 3.14 – Оцінка моделей

Кольорові індикатори допомагають швидко побачити, яка модель дає найменшу помилку. В правій частині розміщено стовпчикову діаграму, що візуально порівнює значення RMSE між моделями, та додатковий блок, де може відображатися частка правильно класифікованих випадків за класами ризику. Такий формат подачі результатів дає змогу інженеру швидко інтерпретувати, наскільки запропонований метод кращий за альтернативи.

Для перевірки запропонованого методу прогнозування обслуговування система тестувалась на наборі даних, який імітує реальну роботу обладнання. В цих даних присутні періоди нормальної роботи, повільної деградації, а також моменти фактичного обслуговування або відмови.

Під час тестування потоки сенсорних даних проходять через модуль передобробки, де формуються ковзні вікна і обчислюються ознаки, а далі подаються на входи LSTM-моделі. Модель повертає прогнозовані значення індексу стану (Health Index, HI_pred), які порівнюються з «еталонними»

значеннями HI_true або з моментами реальних відмов. Це дозволяє оцінити, наскільки рано і стабільно модель «бачить» деградацію.

В тестовому режимі система працює так само, як і в експлуатації: результати прогнозу пишуться в базу часових рядів, на дашборді оновлюються графіки, а правила ризику генерують події попереджень. Оператор бачить одночасно дві криві — фактичного стану й прогнозованого — і може візуально оцінити, де модель «випереджає» появу критичних ситуацій.

На рисунку 3.15 наведено приклад демонстраційного екрану, який ілюструє тестування методу на симульованому IoT-наборі даних.

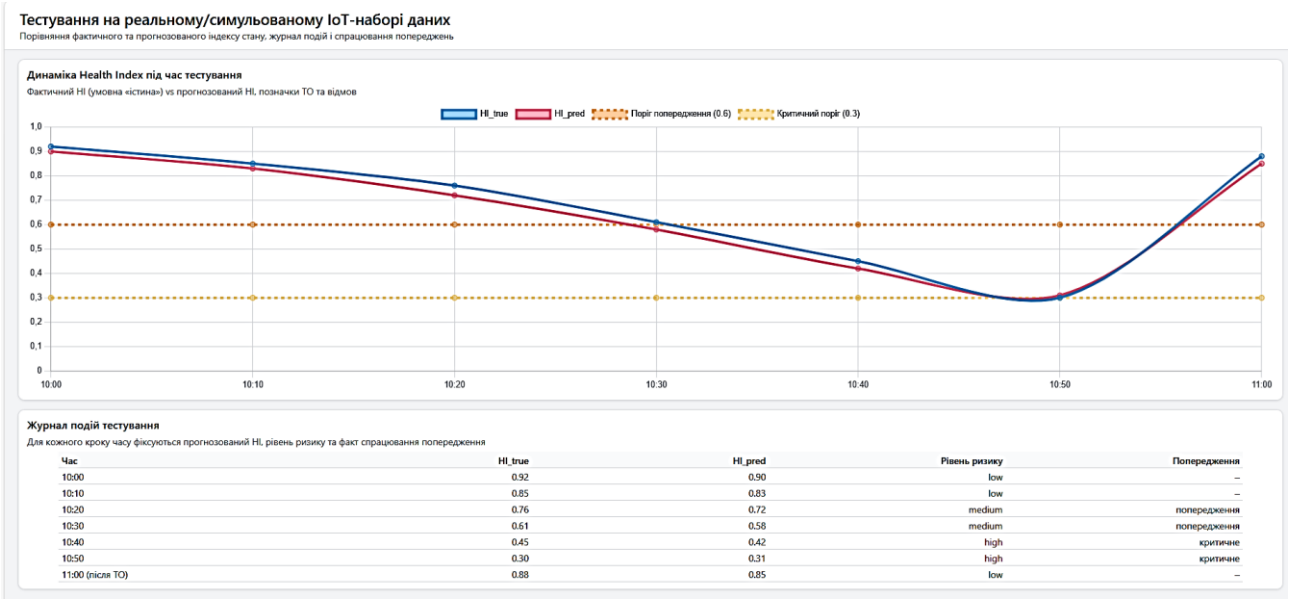


Рисунок 3.15 –Тестування методу

В верхній частині показано графік «фактичний HI» та «прогнозований HI». На графік нанесено вертикальні маркери: зелені позначають моменти планового технічного обслуговування, червоні — моменти відмови. Це дозволяє побачити, за скільки часу до критичної події модель знижує прогнозований індекс стану нижче порогу.

В нижній частині інтерфейсу розміщений журнал подій тестування. Для кожного кроку часу фіксуються дата і час, прогнозований HI, рівень ризику («low», «medium», «high») та факт спрацювання попередження. Журнал дає

змогу порахувати, скільки попереджень було «вчасними», скільки — хибними, а також підтвердити, що модель не створює надмірної кількості тривог.

Висновки до розділу 3

1. Реалізовано повний програмний прототип який реалізує метод прогнозування обслуговування в системах моніторингу на базі IoT – від отримання сирих потоків телеметрії до видачі рішень щодо технічного обслуговування. Побудовано модуль збору та попередньої обробки даних, виконує нормалізацію, згладжування, заповнення пропусків і формування ковзних вікон, після чого зберігає очищені часові ряди до InfluxDB як сховища часових даних.

2. Розроблено блок прогнозування, реалізований на LSTM-моделі індексу технічного стану та залишкового ресурсу (RUL). Запропоновано навчальний конвеєр, який включає завантаження історичних даних з БД, формування вікон ознак, розбиття на train/validation, налаштування архітектури мережі та оптимізацію ваг за критерієм MSE/RMSE.

3. Інтегровано модель в мікросервіс на базі FastAPI. Створено HTTP-ендпоінт /predict, який приймає вікно ознак, виконує використання навченої моделі для отримання прогнозу LSTM-мережі та повертає числовий індекс стану та текстовий рівень ризику («low», «medium», «high»).

4. Реалізовано інтеграцію з платформою моніторингу та візуалізацією результатів, побудовано панель керування з часовими графіками сенсорних сигналів, кривими прогнозованого HI з порогами та панеллю попереджень. Розроблено веб-інтерфейс адміністратора з таблицею обладнання, індикаторами ризику, графіками HI/RUL та можливістю керування порогами та правилами сповіщення.

5. Запропоновано алгоритм сповіщення про потребу обслуговування, який базується на порівнянні прогнозованого HI/RUL з критичними та попереджувальними порогами.

6. Проведено оцінювання ефективності методу на основі порівняння з базовими моделями. Тестування на симульованому IoT-наборі даних, яке продемонструвало здатність системи працювати в режимі, наближеному до реальних умов експлуатації.

7. Отримані результати підтверджують практичну придатність розробленого методу для побудови інтелектуальних систем прогнозного обслуговування на базі IoT.

ВИСНОВКИ

1. Проведено аналіз сучасних підходів до інтелектуального моніторингу на базі IoT та систем прогнозного обслуговування (PdM). Виявлено, що ключовими проблемами є неповнота та нерівномірність потоків сенсорних даних, відсутність уніфікованих форматів телеметрії та потреба в розподіленому аналізі на рівнях edge–fog–cloud.

2. Сформульовано мету дослідження, яка полягає в розробці методу прогнозування обслуговування в системах моніторингу на базі IoT, який стійкий до неповних і нерівномірних сенсорних даних та здатний мінімізувати очікувані витрати за рахунок оптимального планування технічного обслуговування.

3. Запропоновано концепцію багаторівневого аналітичного шару IoT-системи моніторингу, що включає рівні сенсорів, шлюзів, аналітики та сервісів. Для цього шару побудовано математичну модель методу PdM: описано формування послідовностей векторів ознак з часових рядів, задано функціонал похибки прогнозу та введено критерії оцінки ризику відмови та залишкового ресурсу.

4. Розроблено замкнутий конвеєр обробки IoT-даних сенсори - шлюз - брокер повідомлень - модуль попередньої обробки - сховище часових рядів - аналітичний модуль, де реалізовано нормалізацію, згладжування, заповнення пропусків, вирівнювання часової шкали та формування ковзних вікон ознак.

5. Створено прогнозну підсистему на базі LSTM-мережі для оцінювання індексу технічного стану (Health Index) та залишкового ресурсу обладнання. LSTM-модель навчається на векторах ознак, сформованих у ковзних вікнах, з використанням вищезазначених метрик похибки..

6. Сформовано модель прийняття рішень для технічного обслуговування, де порівнюються очікувані витрати сценаріїв. На основі оціненої ймовірності відмови сформульовано правило вибору оптимального моменту ТО, яке забезпечує мінімум очікуваних сумарних витрат.

7. Реалізовано повний програмний прототип системи, який включає модуль збору та попередньої обробки телеметрії, блок LSTM-прогнозування,

мікросервіс FastAPI з REST-ендпоінтом, а також інтеграцію з платформою моніторингу на базі Grafana і Node-RED. Створено веб-інтерфейси для адміністратора де представлено дашборди з часовими графіками сенсорних сигналів і HI, панель попереджень, інтерфейс оцінювання якості моделей та екран тестування.

8. Проведено експериментальне тестування на IoT-наборі даних, що містить періоди нормальної роботи, деградації та моменти фактичного обслуговування та відмов. Результати показали, що запропонований метод забезпечує більш стабільне прогнозування індексу стану та своєчасне формування попереджень.

9. Новизна роботи полягає у поєднанні в одному методі PdM обробки динамічно неповних і нерівномірних потоків сенсорних даних архітектурою Edge–Fog–Cloud та вартісного критерію прийняття рішень щодо обслуговування. Практична цінність полягає в інтеграції розробленого методу в IoT-платформи для моніторингу обладнання з метою зменшення кількості позапланових ремонтів і підвищення ефективності управління технічними ресурсами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Martínez-Heredia A.M., Ventura S. Weak Supervision: A Survey on Predictive Maintenance. *WIREs Data Mining and Knowledge Discovery*. 2025. Vol. 15, No. 2. e70022.

2. Patil S., Kudale P. Predictive Maintenance: Evolution, Purposes and Approaches for Industrial Systems – A Comprehensive Review. *International Journal of Scientific Research in Engineering and Management*. 2025. Vol. 09, No. 01. Pp. 1–10.

3. Zhang Y., Yang G., Wang H., et al. Predictive Maintenance for Industrial Equipment: A Deep Learning Approach. *IEEE Transactions on Industrial Informatics*. 2023. Vol. 19, No. 4. Pp. 5123–5134.

4. Carvalho T.P., Soares F.A.A.M.N., Vita R., et al. A Systematic Literature Review of Machine Learning Methods for Predictive Maintenance. *Computers & Industrial Engineering*. 2019. Vol. 137. 106024.

5. Zhang W., Yang D., Wang H. Data-driven Methods for Predictive Maintenance of Industrial Equipment: A Survey. *IEEE Systems Journal*. 2020. Vol. 14, No. 3. Pp. 3136–3150.

6. Nguyen K.T., Medjaher K. A New Dynamic Predictive Maintenance Framework Using Deep Learning for Smart Manufacturing. Journal of Manufacturing Systems. 2021. Vol. 58. Pp. 369–384.

7. Zhang C., Lim P., Qin A.K., Tan K.C. Multiobjective Deep Learning for Remaining Useful Life Prediction in Asset Maintenance. IEEE Transactions on Neural Networks and Learning Systems. 2020. Vol. 31, No. 7. Pp. 2264–2277.

8. Babu G.S., Zhao P., Li X.L. Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. Applied Intelligence. 2020. Vol. 50. Pp. 3593–3618.

9. Li X., Ding Q., Sun J.Q. Remaining Useful Life Estimation in Prognostics Using Deep Convolution Neural Networks. Reliability Engineering & System Safety. 2020. Vol. 185. 109700.

10. Zhang S., Luo M., Yang S., et al. LSTM-based Remaining Useful Life Prediction for Industrial Systems. Neurocomputing. 2021. Vol. 453. Pp. 380–393.

11. Chen Y., Li C., Qian C., et al. A Hybrid CNN–LSTM Model for Predictive Maintenance of Rotating Machinery. Mechanical Systems and Signal Processing. 2022. Vol. 165. 108382.

12. Aivaliotis P., Georgoulias K., Chryssolouris G. The Use of Digital Twin for Predictive Maintenance in Manufacturing. Procedia Manufacturing. 2019. Vol. 39. Pp. 194–201.

13. Lee J., Lapira E., Bagheri B., Kao H.-A. Recent Advances and Trends in Predictive Manufacturing Systems in Big Data Environment. Manufacturing Letters. 2020. Vol. 26. Pp. 34–40.

14. Susto G.A., Schirru A., Pampuri S., et al. Machine Learning for Predictive Maintenance: A Multiple Classifier Approach. IEEE Transactions on Industrial Informatics. 2020. Vol. 11, No. 3. Pp. 812–820.

15. Zhang B., Qian C., Li X. Physics-Informed Neural Networks for Remaining Useful Life Prediction of Degrading Systems. Reliability Engineering & System Safety. 2024. Vol. 236. 109321.

16. IoT Architecture and Application: A Survey. PeerJ Computer Science. 2016. Vol. 2. e150.

17. Al-Fuqaha A., Guizani M., Mohammadi M., Aledhari M., Ayyash M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials. 2019. Vol. 21, No. 4. Pp. 2692–2772.

18. Wang S., Wan J., Li D., Zhang C. Implementing Smart Factory of Industrie 4.0: An Outlook. International Journal of Distributed Sensor Networks. 2020. Vol. 16, No. 1. 380.

19. Lee I., Lee K. The Internet of Things (IoT): Applications, Investments, and Challenges for Enterprises. Business Horizons. 2019. Vol. 58, No. 4. Pp. 431–440.

20. Amjad M., Shah M.A., Afzal M.K., Kim K. IoT-based Smart Monitoring of Industrial Equipment for Predictive Maintenance. IEEE Access. 2020. Vol. 8. Pp. 68379–68391.

21. Bousdekis A., Magoutas B., Apostolou D., Mentzas G. A Review of Data-driven Decision Support Methods for Maintenance Scheduling. Computers & Industrial Engineering. 2019. Vol. 137. 106024.

22. Xiong R., Tian J., Mu H., et al. A Deep Learning-based Framework for Predictive Maintenance in Cyber–Physical Production Systems. Robotics and Computer-Integrated Manufacturing. 2021. Vol. 68. 102075.

23. Sutrisno E., Oh H., Vasan A.S.S., et al. A Review of Predictive Maintenance: Methods, Systems, and Tools. Journal of Intelligent Manufacturing. 2022. Vol. 33. Pp. 1161–1185.

24. Kliestik T., Misankova M., Valaskova K., et al. AI-based Predictive Maintenance in Automotive Industry: A Survey. Sustainability. 2023. Vol. 15, No. 4. 3456.

25. Domingues P., Cardoso A., Sá da Costa J. An Overview on Remote and Distributed Condition Monitoring of Industrial Systems. Measurement. 2020. Vol. 165. 108257.

26. Peng C., Dong M., Li C. An Integrated LSTM-based Remaining Useful Life Prediction Framework for Complex Industrial Systems. ISA Transactions. 2021. Vol. 112. Pp. 72–81.

27. Wen C., Fan J., Li X., et al. A Multi-sensor Fusion Approach for Predictive Maintenance Using Attention-based LSTM. Sensors. 2023. Vol. 23, No. 21. 5970.

28. ISO 13374-1:2019. Condition monitoring and diagnostics of machines – Data processing, communication and presentation – Part 1: General guidelines. International Organization for Standardization, 2019.

29. IEC 62890:2020. Industrial-process measurement, control and automation – Life-cycle management for systems and components. International Electrotechnical Commission, 2020.

30. McKinsey & Company. The Future of Predictive Maintenance in Manufacturing. McKinsey Report. 2021. 28 p.

31. IBM Corporation. Predictive Maintenance for Industry 4.0: An IBM Point of View. White Paper. 2020. 24 p.

32. Bosch Rexroth. Condition Monitoring and Predictive Maintenance in Industrial Hydraulics. Technical Report. 2022. 20 p.

33. Колдун М. М., Грудзинський Ю. Є. Побудова алгоритму прогностичної моделі при створенні предиктивного модуля передбачення нештатних ситуацій на виробництві // Вісник Національного технічного університету «Харківський політехнічний інститут». Серія: Інформатика та моделювання. – 2020.

34. Фінькова О. В. Інформаційна система аналізу показників сенсорів технологічних ліній : магістерська кваліфікаційна робота. – Запоріжжя : Національний університет «Запорізька політехніка», 2024.

35. Гнатів А. Р. Інформаційна система моніторингу метеопараметрів з використанням концепції Інтернету речей : магістерська кваліфікаційна робота. – Львів : Національний університет «Львівська політехніка», 2024.

36. Констанченко О. Є. Інтелектуальна система моніторингу ґрунту на основі технологій Інтернету речей // Збірник наукових праць з прикладної інформатики. – 2023.

37. Костюк І. С. Комп'ютерна система для діагностування дефектів сонячних панелей у середовищі IoT : кваліфікаційна робота магістра : 123 Комп'ютерна інженерія / І. С. Костюк ; Хмельниц. нац. ун-т. – Хмельницький, 2025. – 116 с.

38. Худяков Д. О. Удосконалення методів оперативного моніторингу та прогнозування технічного стану транспортних засобів : дис. … д-ра філософії. – 2023.

39. Боковий П. О. Розробка архітектури та програмного забезпечення системи Інтернету речей для моніторингу параметрів технічних об'єктів : магістерська кваліфікаційна робота. – Київ : НТУУ «КПІ ім. І. Сікорського», 2019.

40. Микитишин А. Г. Методи та засоби оптимізації IoT систем для мікрокліматного контролю вирощувальних систем : кваліфікаційна робота на здобуття освітнього ступеня магістр за спеціальністю „123 — комп'ютерна інженерія“ / А. Г. Микитишин. — Тернопіль: ТНТУ, 2023. — 69 с.

41. Боричевський В. В., ін. Комп’ютеризована система моніторингу та керування штучним досвічуванням рослин у теплицях на базі сенсорних мереж та нечіткої логіки // Збірник наукових праць з автоматизації та електротехнічних систем. – 2019–2021.

42. Цифрові двійники для технологічних ліній в «розумному місті», Дипломна робота освітнього рівня «Магістр», Мадзей Петро Андрійович, Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кафедра комп’ютерних наук, група СFм-61, Тернопіль, 2019, С.106.

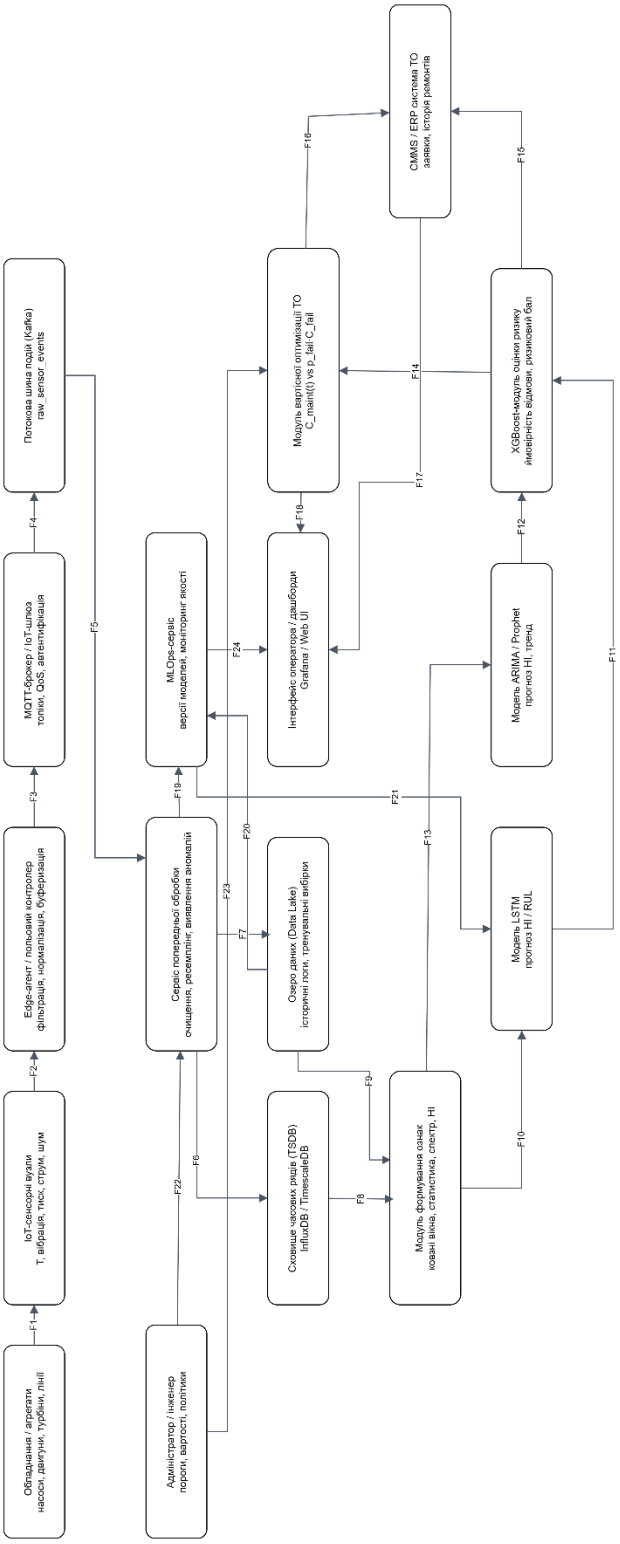
43. Іванов А. М., Осолінський О. Р., Система моніторингу на базі IoT для прогнозування обслуговування, IX Міжнародна мультидисциплінарна студентська наукова конференція «Пріоритетні напрямки та вектори розвитку світової науки», 5 грудня 2025 р. м. Суми, Україна., С.547-549.

44. Іванов А. М., Осолінський О. Р., Метод прогнозування обслуговування в системах IoT-моніторингу, X Міжнародна наукова конференція «Проблеми та перспективи реалізації та впровадження міждисциплінарних наукових досягнень», 5 грудня 2025 р., м. Чернігів., С. 292-294.

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

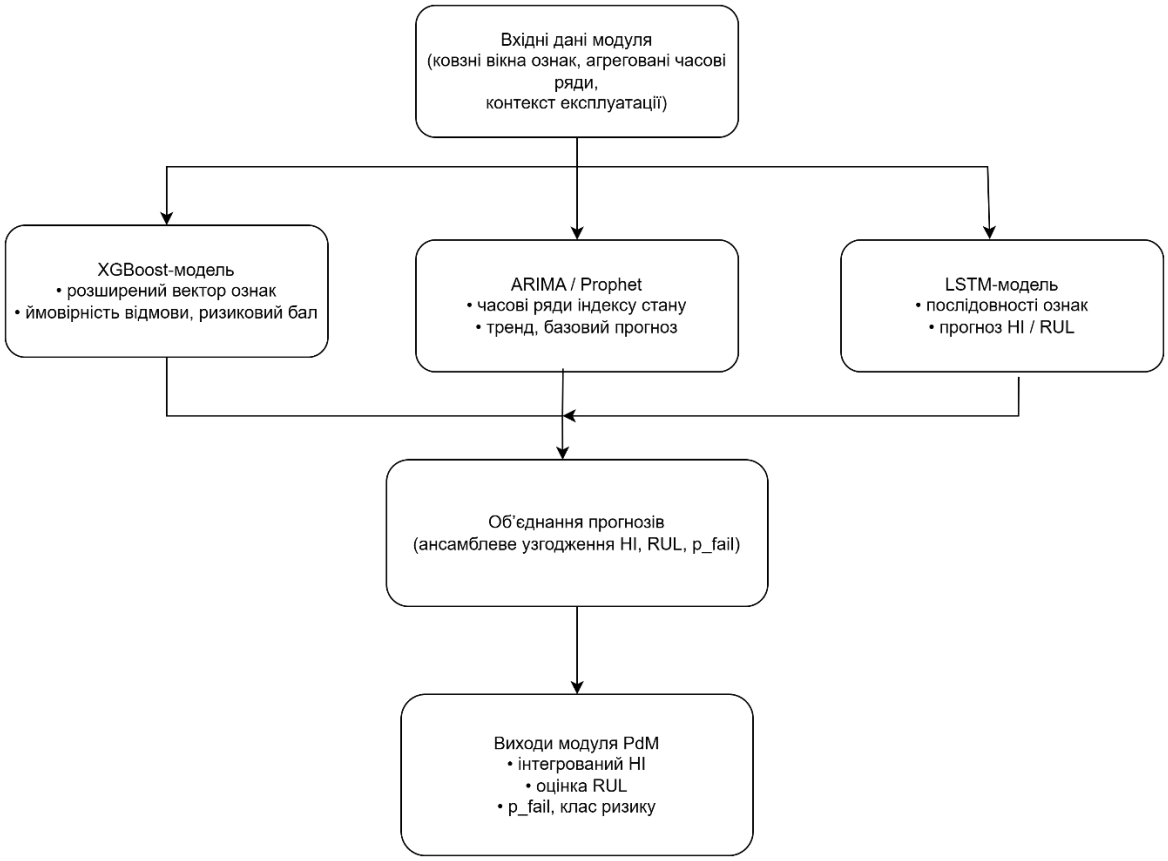
Додаток А

Загальна архітектура системи



Додаток Б

Архітектура гібридної моделі прогнозування на базі LSTM + ARIMA + XGBoost



Додаток В

Програмна реалізація гібридного модуля прогнозного обслуговування на базі LSTM, ARIMA та XGBoost

```python
import numpy as np
import pandas as pd

from sklearn.metrics import (
    roc_auc_score,
    f1_score,
    accuracy_score,
    mean_squared_error,
    r2_score
)
from xgboost import XGBClassifier, XGBRegressor
from statsmodels.tsa.arima.model import ARIMA

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.optimizers import Adam


# ------------------------------------------------------------
# 1. Допоміжна функція: формування LSTM-датасету
#    з features_df (послідовності ковзних вікон)
# ------------------------------------------------------------

def make_lstm_dataset(
    features_df: pd.DataFrame,
    target: pd.Series,
    seq_len: int = 10
):
    """
    Формує вибірку для LSTM з табличних ознак.

    Вхід:
      features_df : DataFrame з ознаками (результат build_features)
                    індекс – час (window_timestamp).
      target      : Series з цільовими значеннями (0/1 або RUL/HI),
                    індекс має збігатися з features_df.index.
      seq_len     : довжина послідовності (кількість вікон в одному LSTM-зразку).

    Вихід:
      X_seq : np.ndarray [n_samples, seq_len, n_features]
      y_seq : np.ndarray [n_samples]
      ts    : Index – часові мітки останнього вікна (у вирівняному наборі)
    """
    # Вирівнюємо ціль по індексу
    target = target.reindex(features_df.index)
    target = target.astype(float).fillna(method="ffill").fillna(method="b

    values = features_df.values
    y_vals = target.values
    idx = features_df.index.to_numpy()

    X_seq, y_seq, ts = [], [], []

    for i in range(seq_len, len(features_df)):
        X_seq.append(values[i - seq_len:i])
        y_seq.append(y_vals[i])
        ts.append(idx[i])

    X_seq = np.array(X_seq, dtype=float)
    y_seq = np.array(y_seq, dtype=float)
    ts = pd.Index(ts)

    return X_seq, y_seq, ts


# ------------------------------------------------------------
# 2. Основний пайплайн: гібридний модуль PdM
#    (LSTM + ARIMA + XGBoost)
# ------------------------------------------------------------

def train_hybrid_pdm_pipeline(
    features_df: pd.DataFrame,
    target: pd.Series,
    lstm_seq_len: int = 10,
    arima_order=(2, 0, 2),
    task: str = "classification",  # "classification" або "regression"
    test_ratio: float = 0.2,
    val_ratio: float = 0.1,
    random_state: int = 42
):
    """
    Тренує гібридну модель:
      - LSTM по послідовності векторів ознак;
      - ARIMA по часовому ряду target;
      - XGBoost як метамодель, яка бере:
        * табличні ознаки;
```

```
        * прогноз LSTM;
        * прогноз ARIMA.

    Вхід:
      features_df : DataFrame з ознаками (результат build_features)
      target      : Series з цільовими значеннями (0/1 або неперервні)
      lstm_seq_len: довжина послідовності для LSTM
      arima_order : параметри ARIMA(p,d,q)
      task        : "classification" або "regression"
      test_ratio  : частка тестової вибірки (по часу)
      val_ratio   : частка валідаційної всередині train

    Вихід (dict):
      {
        "lstm_model": keras.Model,
        "arima_model": statsmodels.ARIMAResults,
        "xgb_model": XGBClassifier/XGBRegressor,
        "metrics": {...},
        "meta_train_index": Index,
        "meta_test_index": Index
      }
    """

    # --- 2.1 Формуємо LSTM-датасет ---
    X_seq, y_seq, ts_seq = make_lstm_dataset(
        features_df=features_df,
        target=target,
        seq_len=lstm_seq_len
    )

    n_samples = len(ts_seq)
    if n_samples < 20:
        raise ValueError("Замало вікон для навчання LSTM/ARIMA/XGBoost")

    # Розбиття по часу (без shuffle)
    # train : val : test ≈ (1 - test_ratio) ділимо ще на train/val
    test_size = int(np.floor(n_samples * test_ratio))
    val_size = int(np.floor(n_samples * val_ratio))
    train_size = n_samples - test_size - val_size

    if train_size <= 0:
        raise ValueError("Надто великий test_ratio/val_ratio для малого датасету.")
```

```
    train_slice = slice(0, train_size)
    val_slice = slice(train_size, train_size + val_size)
    test_slice = slice(train_size + val_size, n_samples)

    X_train, y_train = X_seq[train_slice], y_seq[train_slice]
    X_val, y_val = X_seq[val_slice], y_seq[val_slice]
    X_test, y_test = X_seq[test_slice], y_seq[test_slice]

    ts_train = ts_seq[train_slice]
    ts_val = ts_seq[val_slice]
    ts_test = ts_seq[test_slice]

    n_features = X_train.shape[-1]

    # --- 2.2 LSTM-модель ---
    lstm_model = Sequential()
    lstm_model.add(LSTM(64, input_shape=(lstm_seq_len, n_features), return_sequences=False))
    lstm_model.add(Dropout(0.2))
    if task == "classification":
        lstm_model.add(Dense(1, activation="sigmoid"))
        lstm_model.compile(
            loss="binary_crossentropy",
            optimizer=Adam(learning_rate=1e-3),
            metrics=["accuracy"]
        )
    else:  # regression
        lstm_model.add(Dense(1, activation="linear"))
        lstm_model.compile(
            loss="mse",
            optimizer=Adam(learning_rate=1e-3),
            metrics=["mse"]
        )

    # Навчання LSTM
    lstm_model.fit(
        X_train, y_train,
        validation_data=(X_val, y_val) if len(X_val) > 0 else None,
        epochs=20,
        batch_size=32,
        verbose=1
    )
```

```python

    # Отримуємо LSTM-прогнози для ВСІХ зразків (train+val+test)
    y_lstm_all = lstm_model.predict(X_seq, verbose=0).flatten()

    # Вирівнюємо їх за ts_seq (індекс)
    lstm_preds_series = pd.Series(y_lstm_all, index=ts_seq, name="y_hat_lstm")

    # --- 2.3 ARIMA-модель (по всьому часовому ряду target) ---
    # Для простоти беремо target у тому ж порядку, що й features_df
    target_full = target.reindex(features_df.index).astype(float)
    target_full = target_full.interpolate().ffill().bfill()

    arima_model = ARIMA(target_full.values, order=arima_order).fit()

    # Прогнози ARIMA у тій самій області, де є LSTM-вікна:
    # Починаємо з індексу lstm_seq_len (бо до цього вікна ще неповні)
    start_idx = lstm_seq_len
    end_idx = lstm_seq_len + n_samples - 1

    arima_pred_all = arima_model.predict(start=start_idx, end=end_idx)
    arima_pred_series = pd.Series(
        arima_pred_all,
        index=ts_seq,
        name="y_hat_arima"
    )

    # --- 2.4 Формуємо мета-датасет для XGBoost ---
    # Беремо лише ті рядки features_df, для яких є ts_seq
    base_df_for_meta = features_df.loc[ts_seq]

    # X_meta = [усі базові ознаки] + [y_hat_lstm] + [y_hat_arima]
    X_meta = base_df_for_meta.copy()
    X_meta["y_hat_lstm"] = lstm_preds_series
    X_meta["y_hat_arima"] = arima_pred_series

    # Ціль:
    y_meta = target.reindex(ts_seq).astype(float)
    y_meta = y_meta.fillna(method="ffill").fillna(method="bfill")

    # Розбиваємо X_meta/y_meta за тим самим train/val/test
    X_meta_train = X_meta.iloc[train_slice]
    y_meta_train = y_meta.iloc[train_slice]
    --    _       -_              _

    X_meta_val = X_meta.iloc[val_slice]
    y_meta_val = y_meta.iloc[val_slice]

    X_meta_test = X_meta.iloc[test_slice]
    y_meta_test = y_meta.iloc[test_slice]

    # --- 2.5 Навчання XGBoost як метамоделі ---
    if task == "classification":
        xgb_model = XGBClassifier(
            n_estimators=200,
            max_depth=4,
            learning_rate=0.05,
            subsample=0.8,
            colsample_bytree=0.8,
            random_state=random_state,
            n_jobs=-1
        )
    else:  # regression
        xgb_model = XGBRegressor(
            n_estimators=300,
            max_depth=4,
            learning_rate=0.05,
            subsample=0.8,
            colsample_bytree=0.8,
            random_state=random_state,
            n_jobs=-1
        )

    xgb_model.fit(
        X_meta_train.values,
        y_meta_train.values
    )

    # --- 2.6 Оцінювання (тільки XGBoost як кінцевий прогноз PdM) ---
    y_hat_test = xgb_model.predict(X_meta_test.values)

    metrics = {}

    if task == "classification":
        # Якщо XGBoost повертає ймовірності (для binary), краще використовувати predict_proba
        try:
```

```
            y_proba_test = xgb_model.predict_proba(X_meta_test.values)[:, 1]
        except Exception:
            y_proba_test = y_hat_test

        y_pred_labels = (y_proba_test >= 0.5).astype(int)
        y_true_labels = y_meta_test.values.astype(int)

        metrics["test_accuracy"] = float(accuracy_score(y_true_labels, y_pred_labels))
        metrics["test_f1"] = float(f1_score(y_true_labels, y_pred_labels))
        metrics["test_roc_auc"] = float(roc_auc_score(y_true_labels, y_proba_test))
    else:
        mse = mean_squared_error(y_meta_test.values, y_hat_test)
        r2 = r2_score(y_meta_test.values, y_hat_test)
        metrics["test_mse"] = float(mse)
        metrics["test_rmse"] = float(np.sqrt(mse))
        metrics["test_r2"] = float(r2)

    # Також можна порахувати метрики для самого LSTM чи ARIMA окремо
    # (для порівняння з XGBoost як метамоделлю)

    return {
        "lstm_model": lstm_model,
        "arima_model": arima_model,
        "xgb_model": xgb_model,
        "metrics": metrics,
        "meta_train_index": ts_train,
        "meta_val_index": ts_val,
        "meta_test_index": ts_test
    }
```

Додаток Г

Апробація отриманих результатів

МОЛОДІЖНА НАУКОВА ЛІГА

МАТЕРІАЛИ IX МІЖНАРОДНОЇ СТУДЕНТСЬКОЇ НАУКОВОЇ

КОНФЕРЕНЦІЇ

ПРІОРИТЕТНІ НАПРЯМКИ ТА ВЕКТОРИ РОЗВИТКУ СВІТОВОЇ НАУКИ

м. Суми, Україна
5 грудня 2025 рік

Вінниця, Україна
«UKRLOGOS Group»
2025

УДК 082:001
П 75

Голова оргкомітету: Коренюк І.О.
Верстка: Білоус Т.В.
Дизайн: Бондаренко І.В.

Рекомендовано до видання Вченою Радою Інституту науково-технічної інтеграції та співпраці. Протокол № 48 від 04.12.2025 року.

Конференцію зареєстровано Державною науковою установою «УкрІНТЕІ» в базі даних науково-технічних заходів України та бюлетені «План проведення наукових, науково-технічних заходів в Україні» (Посвідчення № 459 від 10.06.2025).

П 75 **Пріоритетні напрямки та вектори розвитку світової науки**: матеріали IX Міжнародної студентської наукової конференції, м. Суми, 5 грудня, 2025 рік / ГО «Молодіжна наукова ліга». — Вінниця: ТОВ «УКРЛОГОС Груп», 2025. — 814 с.

ISBN 978-617-8582-06-7
DOI 10.62732/liga-inter-05.12.2025

Викладено матеріали учасників IX Міжнародної мультидисциплінарної студентської наукової конференції «Пріоритетні напрямки та вектори розвитку світової науки», яка відбулася 5 грудня 2025 року у місті Суми, Україна.

УДК 082:001

ISBN 978-617-8582-06-7

Іванов Андрій Михайлович, магістр спеціальності "Комп'ютерні науки"
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович, канд.техн.наук, доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

СИСТЕМА МОНІТОРИНГУ НА БАЗІ IoT ДЛЯ ПРОГНОЗУВАННЯ ОБСЛУГОВУВАННЯ

Вступ

В сучасних інфраструктурних комплексах витрати на підтримання працездатності обладнання займають значну частку собівартості продукції і послуг, а простої через аварійні відмови безпосередньо впливають на безпеку та конкурентоспроможність підприємства. Традиційні підходи до технічного обслуговування, тобто реагування по факту відмови або регламентні роботи за календарем чи напрацюванням виявляються економічно неефективними, тому що частина ресурсів витрачається на зайві профілактичні втручання, тоді як реальні дефекти виявляються із запізненням. Тому зараз прогнозне обслуговування, яке усуває недоліки класичних підходів розглядається як стратегія, що дозволяє планувати ремонтні дії на основі поточного стану та очікуваної деградації вузлів, мінімізуючи ризик критичних відмов і оптимізуючи витрати на обслуговування.

Ключовою передумовою для впровадження PdM є наявність розгалуженої інфраструктури моніторингу. Інтернет речей (IoT) надає для цього необхідні засоби, що включає сенсорні вузли, вбудовані контролери, бездротові канали зв'язку та хмарні сервіси, які дозволяють формувати єдиний інформаційний простір. Використання такого підходу дає можливість поєднати онлайн-моніторинг, аналітику великих масивів історичних даних та інтеграції з системами управління технічним обслуговуванням і ремонтами.

В таких умовах зростає потреба в цілісних рішеннях, які одночасно враховують особливості IoT-інфраструктури та системи які дозволяють прогнозувати планове обслуговування.

Реалізація системи прогнозування обслуговування

В запропонованій системі прогнозування обслуговування на базі IoT важливу роль відіграє платформа моніторингу, яка об'єднує збір даних, аналітику та візуалізацію для оператора або системного адміністратора. Найчастіше використовується зв'язка сенсори + шлюзи +хмара/edge + графічний інтерфейс, де окремий модуль відповідає за відображення стану обладнання та сповіщення про ризики.

На рисунку 1 подано інтерфейс адміністратора, який використовується для візуалізації результатів прогнозування обслуговування в IoT-системі який дозволяє оператору співставляти аномалії в даних із конкретними режимами роботи обладнання.

На рисунку 2 наведено приклад веб-інтерфейсу, орієнтованого на системного адміністратора. Центральне місце займає таблиця з переліком обладнання, куди входить ідентифікатор, назва, індекс стану, рівень ризику, орієнтовний RUL та рекомендований статус, наприклад норма, планувати ТО або невідкладна перевірка.

Рис. 1. **Інтерфейс адміністратора для візуалізації результатів прогнозування обслуговування в IoT-системі**

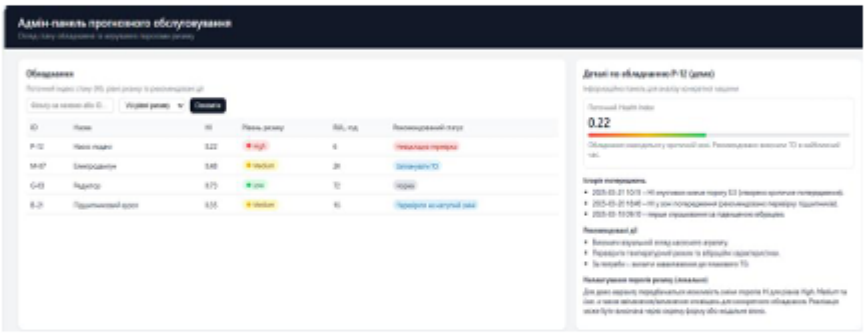


Рис. 2. **Веб-інтерфейс адміністратора**

Алгоритм сповіщення працює послідовно. Спочатку сервіс прогнозування з певною періодичністю обробляє нові вікна сенсорних даних і обчислює для кожного агрегату індекс стану HI та оцінку залишкового ресурсу RUL. Далі застосовуються правила ризику: якщо HI < HI_critical або RUL < RUL_critical, формується критичне попередження; якщо HI менший за попереджувальний поріг, але ще не досяг критичного, створюється попередження «середній ризик».

На завершальному етапі оператор після фактичного виконання обслуговування позначає подію, як закриту та фіксує реальний стан обладнання. Ця інформація повертається в контур навчання моделі й використовується як додаткові мітки для подальшого покращення якості прогнозів.

На рисунку 3 наведено приклад демонстраційного екрану, який ілюструє тестування методу на симульованому IoT-наборі даних.

В нижній частині інтерфейсу розміщений журнал подій тестування. Для кожного кроку часу фіксуються дата і час, прогнозований HI, рівень ризику («low», «medium», «high») та факт спрацювання попередження. Журнал дає змогу порахувати, скільки попереджень було «вчасними», скільки — хибними, а також підтвердження, що модель не створює надмірної кількості тривог.

Рис. 3. **Тестування методу**

Висновки. Реалізовано повний програмний прототип який реалізує метод прогнозування обслуговування в системах моніторингу на базі IoT – від отримання сирих потоків телеметрії до видачі рішень щодо технічного обслуговування.

Список використаних джерел:

1. Martínez-Heredia A.M., Ventura S. Weak Supervision: A Survey on Predictive Maintenance. Wiley Interdisciplinary Reviews: Data Mining and Knowledge *Discovery*. 2025. Vol. 15, No. 2. e70022.
2. Patil S., Kudale P. Predictive Maintenance: Evolution, Purposes and Approaches for Industrial Systems – A Comprehensive Review. *International Journal of Scientific Research in Engineering and Management*. 2025. Vol. 09, No. 01. P. 1–10.
3. Zhang Y., Yang G., Wang H., et al. Predictive Maintenance for Industrial Equipment: A Deep Learning Approach. *IEEE Transactions on Industrial Informatics*. 2023. Vol. 19, No. 4. P. 5123–5134.
4. Carvalho T.P., Soares F.A.A.M.N., Vita R., et al. A Systematic Literature Review of Machine Learning Methods for Predictive Maintenance. *Computers & Industrial Engineering*. 2019. Vol. 137. P. 106024.

ЗБІРНИК НАУКОВИХ ПРАЦЬ

З МАТЕРІАЛАМИ X МІЖНАРОДНОЇ НАУКОВОЇ КОНФЕРЕНЦІЇ

5 ГРУДНЯ 2025 РІК

М. ЧЕРНІГІВ, УКРАЇНА

«ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ РЕАЛІЗАЦІЇ ТА ВПРОВАДЖЕННЯ МІЖДИСЦИПЛІНАРНИХ НАУКОВИХ ДОСЯГНЕНЬ»

ЗБІРНИК НАУКОВИХ
ПРАЦЬ З МАТЕРІАЛАМИ
X МІЖНАРОДНОЇ
НАУКОВОЇ КОНФЕРЕНЦІЇ

ПРОБЛЕМИ ТА ПЕРСПЕКТИВИ РЕАЛІЗАЦІЇ ТА ВПРОВАДЖЕННЯ МІЖДИСЦИПЛІНАРНИХ НАУКОВИХ ДОСЯГНЕНЬ

5 грудня 2025 рік
м. Чернігів, Україна

Вінниця, Україна
«UKRLOGOS Group»
2025

УДК 082:001
П 78

Організація, від імені якої випущено видання:
ГО «Міжнародний центр наукових досліджень»
Номер запису організації в Єдиному реєстрі громадських об'єднань: 1499141.

Голова оргкомітету: Сотник С.Г.

Верстка: Білоус Т.В.
Дизайн: Бондаренко І.В.

Рекомендовано до видання Вченою Радою Інституту науково-технічної інтеграції та співпраці. Протокол № 48 від 04.12.2025 року.

*Конференцію зареєстровано Державною науковою установою у сфері управління Міністерства освіти і науки «Український інститут науково-технічної експертизи та інформації» в базі даних науково-технічних заходів України на поточний рік та бюлетені «План проведення наукових науково-технічних заходів в Україні» (**Посвідчення № 499 від 10.06.2025**).*

*Збірник наукових праць з матеріалами конференції видано офіційно суб'єктом видавничої справи зі **Свідоцтвом ДК № 7860 від 22.06.2023**.*

Матеріали конференції знаходяться у відкритому доступі на умовах ліцензії Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

П 78 **Проблеми та перспективи реалізації та впровадження міждисциплінарних наукових досягнень**: збірник наукових праць з матеріалами X Міжнародної наукової конференції, м. Чернігів, 5 грудня, 2025 р. / Міжнародний центр наукових досліджень. — Вінниця: ТОВ «УКРЛОГОС Груп, 2025. — 626 с.

ISBN 978-617-8582-05-0
DOI 10.62731/mcnd-05.12.2025

Викладено матеріали учасників X Міжнародної наукової конференції «Проблеми та перспективи реалізації та впровадження міждисциплінарних наукових досягнень», яка відбулася 5 грудня 2025 року у місті Чернігів.

УДК 082:001

ISBN 978-617-8582-05-0

СЕКЦІЯ XIII.
ЕКОЛОГІЯ ТА ТЕХНОЛОГІЇ ЗАХИСТУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА

СЕКЦІЯ XIV.
КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ

СЕКЦІЯ XV.
ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ

СЕКЦІЯ XVI.
ТРАНСПОРТ ТА ТРАНСПОРТНІ ТЕХНОЛОГІЇ

МЕТОД ПРОГНОЗУВАННЯ ОБСЛУГОВУВАННЯ В СИСТЕМАХ IOT-МОНІТОРИНГУ

Іванов Андрій Михайлович
Магістр спеціальності "Комп'ютерні науки"
Західноукраїнський національний університет, Україна

Науковий керівник: Осолінський Олександр Романович
ORCID ID: 0000-0002-0136-395X
канд.техн.наук, доцент кафедри інформаційно-обчислювальних систем і управління
Західноукраїнський національний університет, Україна

Вступ

Розвиток концепцій «Індустрія 4.0» та «розумне виробництво» супроводжується широким впровадженням систем Інтернету речей (IoT) для безперервного моніторингу технічного стану обладнання [5]. Сенсорні вузли, вбудовані контролери та хмарні сервіси формують єдину цифрову інфраструктуру, в якій телеметрія з фізичних об'єктів збирається і використовується для аналітики та підтримки прийняття рішень. Однією з ключових задач в таких системах є прогнозне технічне обслуговування (PdM), яке дає змогу виконувати ремонтні дії саме в момент, коли ризик відмови стає економічно доцільним [1].

Але практичне впровадження таких систем в розподілених IoT-інфраструктурах супроводжується деякими проблемами, саме неповнотою та нерівномірністю потоків даних, шумами, асинхронністтю часових рядів між сенсорами, обмеженими ресурсами периферійних вузлів, та необхідністю врахування економічних критеріїв під час вибору часу обслуговування [2].

Для цього запропоновано багаторівневу архітектуру аналітичної системи та гібридний прогностичний модуль, що поєднує LSTM-мережу, модель ARIMA/Prophet та XGBoost-класіфікатор ризику [1-5].

Пропонований метод прогнозування обслуговування

Для вирішення даної задачі була обрана комбінація кількох алгоритмів, де ядром є нейронна мережа типу LSTM [6]. Сенсорні дані є послідовними вимірюваннями в часі, тому потрібно не просто детектувати окремі значення, а враховувати, як змінювався сигнал раніше. LSTM має особливість запам'ятовувати попередні стани,

відсікати зайвий шум і виділяти довгострокові закономірності, що дозволяє виявляти повільну деградацію вузлів, зміну трендів та перехід з нормального режиму у проблемний.

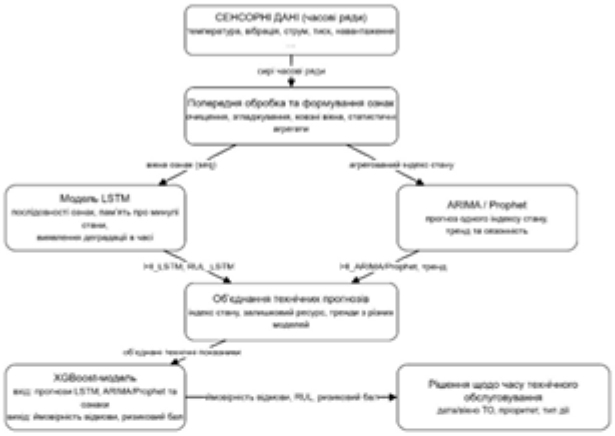


Рис. 1. **Блок-схема запропонованого методу прогнозування обслуговування**

Паралельно використовуються класичні моделі часових рядів, такі як ARIMA. Вони добре підходять для прогнозування одного узагальненого індексу стану, показують тренд, сезонність і дають інтерпретовані результати. Це необхідно для того, щоб визначати чому система детектує, що стан погіршується. Такі моделі виконують роль еталонної бази для порівняння з глибокою нейронною мережею.

На завершальному етапі використовується алгоритм XGBoost, як модель для оцінювання ризику відмови. На його вхід подаються прогнози LSTM, результати ARIMA та додаткові ознаки, такі як температура, вібрація, індекси завантаження. XGBoost перетворює ці дані на ймовірність відмови в заданому часовому горизонті та на ризиковий бал. Тобто виходить, що поєднання LSTM, ARIMA та XGBoost дає можливість одночасно враховувати складну часову динаміку, мати інтерпретовані тренди і отримувати зрозумілу для прийняття рішень оцінку ризику.

Висновки.

Запропоновано підхід прогнозування обслуговування в системах IoT-моніторингу, який інтегрує сенсорні мережі, периферійну та хмарну аналітику, модуль формування ознак, гібридний прогностичний модуль і підсистему вартісної оптимізації. Даний метод може бути реалізований на базі відкритих технологій, таких як MQTT, InfluxDB, Python, TensorFlow, XGBoost і який можна адаптувати до різних галузей

Список використаних джерел:

4. Serradilla O., González A., Triguero I., García S. Deep learning models for predictive maintenance: A survey. *Applied Intelligence*. 2022. Vol. 52. P. 11037–11074.
5. Canizo J. M., Onieva E., Conde A., Charramendieta S., Trujillo S. Real-time predictive maintenance for wind turbines using Big Data frameworks. *IEEE International Conference on Big Data*. 2019. P. 3744–3753.
6. Zhao S., Wang D., Yan R., Mao K. Remaining useful life prediction of rolling bearings using deep CNN. *Neurocomputing*. 2020. Vol. 407. P. 556–566.
7. Yoon J., Kim S., Choi J. Cost-sensitive maintenance planning using prognostics information. *Reliability Engineering & System Safety*. 2019. Vol. 188. P. 90–99.
8. Bousdekis A., Magoutas B., Apostolou D., Mentzas G. Review, analysis and synthesis of prognostic-based decision support methods for condition-based maintenance. *Journal of Intelligent Manufacturing*. 2019. Vol. 30. P. 2803–2825.
9. Ragab A. M., Elattar H. M., Aly A. A. LSTM-based remaining useful life prediction for industrial systems. *International Journal of Prognostics and Health Management*. 2023. Vol. 14. No. 1. P. 1–12.