Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно–інтегрованих технологій, автоматизації та робототехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
Розроблення програмного модуля оперативно-диспетчерського контролю виробництва на базі кібер-фізичних систем керування
(тема)

Виконав:
здобувач 2 року навчання,
групи КІТПВм-24-1

Дмитро РЯБОВОЛ
Спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Тип програми Освітньо–професійна
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва
Керівник доц. Наталія ДЕМСЬКА

Допускається до захисту
Зав. кафедри КІТАР _____ Ігор НЕВЛЮДОВ
(підпис)

2025

Я, Рябовол Дмитро Андрійович, як здобувач вищої освіти ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Я не використовував штучний інтелект для підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

18 грудня 2025 р.  Дмитро РЯБОВОЛ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет АКТ
Кафедра КІТАР
Рівень вищої освіти другий (магістерський)
Спеціальність 174 Автоматизація, комп'ютерно–інтегровані технології та робототехніка
Тип програми Освітньо-професійна
Освітня програма Комп'ютерно-інтегровані технологічні процеси і виробництва

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР _____
(підпис)
« ___»_____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Рябоволу Дмитру Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програмного модуля оперативно-диспетчерського контролю виробництва на базі кібер-фізичних систем керування

Затверджена наказом по університету від 10.11.2025 р. № 1029 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 24.12.2025 р.

3. Вихідні дані до роботи 3.1 Використане програмне забезпечення: Node-RED, Python, LLM API; 3.2 Для розробки використовується Visual Studio Code, Node-RED Editor, веб-браузер Chrome; 3.3 Системні вимоги: ОЗП 8-16 ГБ, процесор 2-4 ядра, вільне місце 5 ГБ, підключення до інтернету. Обладнання використовується для розробки та експериментів.

4. Перелік питань, що потрібно опрацювати в роботі 4.1 Вступ; 4.2 Аналіз існуючих підходів до інтелектуальної підтримки операторів; 4.3 Обґрунтування технологічного стеку та архітектури; 4.4 Визначення параметрів моніторингу та сценаріїв; 4.5 Розроблення алгоритмів детекції та генерації рекомендацій; 4.6 Реалізація програмного модуля; 4.7 Створення інтерфейсу оператора та інтеграція з ШІ-сервісом; 4.8 Експериментальна оцінка ефективності та статистичний аналіз; 4.9 Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій
Графічний матеріал у вигляді презентації – 12 арк. ф. А 4

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз існуючих підходів до автоматизації оперативно-диспетчерського контролю	01.09.25-23.09.25	виконано
2	Постановка задачі та вибір об'єкта дослідження	24.09.25-18.10.25	виконано
3	Розроблення архітектури та алгоритмів програмного модуля	19.10.25-17.11.25	виконано
4	Реалізація та експериментальна оцінка ефективності системи	17.11.25-22.11.25	виконано
5	Тестування функціональності, оптимізація та підготовка висновків / звітів	23.11.25-30.11.25	виконано
6	Оформлення пояснювальної записки	23.11.25-01.12.25	виконано
7	Подання роботи на нормоконтроль	19.12.25	виконано
8	Подання роботи для перевірки роботи на академічну доброчесність		
9	Подання роботи на рецензію		
10	Подання роботи на підпис зав. кафедри		
11	Подання кваліфікаційної роботи в ЕК		

Дата видачі завдання 01.09.2025 р.

Здобувач _____ Дмитро РЯБОВОЛ
(підпис)

Керівник роботи _____ доц. Наталія ДЕМСЬКА
(підпис)

РЕФЕРАТ

Пояснювальна записка: 82 с., 2 табл., 11 рис., 4 дод., 24 джерела.

КІБЕР-ФІЗИЧНА СИСТЕМА, ОПЕРАТИВНО-ДИСПЕТЧЕРСЬКИЙ КОНТРОЛЬ, ШТУЧНИЙ ІНТЕЛЕКТ, ГЕНЕРАЦІЯ РЕКОМЕНДАЦІЙ.

Об'єкт дослідження – процес оперативно-диспетчерського контролю виробництва на базі кіберфізичних систем керування.

Предмет дослідження – програмний модуль інтелектуальної підтримки оператора з інтеграцією штучного інтелекту, методи генерації рекомендацій на основі великих мовних моделей.

Мета роботи – підвищення оперативності прийняття рішень операторами кіберфізичних систем шляхом розроблення програмного модуля з інтеграцією ШІ-помічника для автоматичної генерації рекомендацій.

У першому розділі проведено аналіз існуючих підходів до інтелектуальної підтримки операторів та обґрунтовано використання великих мовних моделей. Обґрунтовано вибір технологічного стеку.

У другому розділі визначено універсальні параметри моніторингу та розроблено п'ять модельних сценаріїв аварійних ситуацій.

У третьому розділі розроблено трирівневу архітектуру програмного модуля та алгоритми формування запитів до ШІ-сервісу. Застосовано метод Prompt Engineering для генерації рекомендацій.

У четвертому розділі наведено результати реалізації. Експериментальне дослідження показало покращення швидкості прийняття рішень.

Результат: програмний модуль оперативно-диспетчерського контролю з ШІ-помічником скорочує час прийняття рішення у 161 раз шляхом автоматичної генерації рекомендацій протягом 2-3 с.

ABSTRACT

Explanatory note: 82 p., 2 tables, 11 fig., 4 appendices, 24 references.

CYBER-PHYSICAL SYSTEM, SUPERVISORY CONTROL, ARTIFICIAL INTELLIGENCE, RECOMMENDATION GENERATION.

Object of research – supervisory control process of production based on cyber-physical control systems.

Subject of research – software module for intelligent operator support with AI integration, methods for generating recommendations based on large language models.

Purpose of work – to improve decision-making efficiency of cyber-physical system operators by developing a software module with AI assistant integration for automatic recommendation generation.

The first chapter analyzes existing approaches to intelligent operator support and justifies the use of large language models. The technological stack of Node-RED, Python Flask, and AI service is substantiated.

The second chapter defines universal monitoring parameters and develops five model emergency scenarios. Algorithms for deviation detection are formalized and experimental evaluation task is set.

The third chapter develops a three-tier software module architecture and algorithms for forming requests to the AI service.

The fourth chapter presents implementation results. Experimental study showed 161-fold improvement in decision-making speed.

Result: supervisory control software module with AI assistant reduces decision-making time by 161 times through automatic recommendation generation within 2-3 seconds.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

АРМ – автоматизоване робоче місце

КФС – кібер-фізична система

ОДК – оперативно-диспетчерський контроль

ПЗ – програмне забезпечення

ПЛК – програмований логічний контролер

ТЗ – технічне завдання

ШІ – штучний інтелект

API – Application Programming Interface

CPU – Central Processing Unit

CSV – Comma-Separated Values

ERP – Enterprise Resource Planning

HMI – Human-Machine Interface

HTTP – HyperText Transfer Protocol

JSON – JavaScript Object Notation

LLM – Large Language Model

MES – Manufacturing Execution System

OPC UA – Open Platform Communications Unified Architecture

PDF – Portable Document Format

REST – Representational State Transfer

SCADA – Supervisory Control And Data Acquisition

SOP – Standard Operating Procedure

ВСТУП

Сучасний світ промислового виробництва постійно вимагає підвищення ефективності, надійності та безпеки технологічних процесів. Однією з ключових вимог є автоматизація оперативно-диспетчерського контролю виробничих об'єктів, що потребує розроблення сучасних та надійних систем підтримки операторів. Особливо актуальним це стає в умовах, де необхідний оперативний моніторинг численних параметрів та швидке реагування на відхилення від технологічних норм [1].

Оператори кіберфізичних систем керування щоденно стикаються з необхідністю швидкого прийняття рішень у критичних ситуаціях. При виникненні аварійних подій оператор повинен оперативно знайти відповідні інструкції в технічній документації, проаналізувати ситуацію та прийняти правильне рішення. Цей процес може займати значний час (декілька хвилин) і залежить від досвіду оператора, що може призводити до помилок та затримок у реагуванні [2].

У зв'язку з цим виникає потреба у розробленні спеціалізованих інструментів інтелектуальної підтримки операторів. Особливою увагою користуються програмні модулі, які дозволяють автоматизувати пошук релевантної інформації та надавати оператору контекстні рекомендації на основі штучного інтелекту.

Метою роботи є підвищення оперативності та надійності прийняття рішень операторами кіберфізичних систем шляхом розроблення програмного модуля оперативно-диспетчерського контролю з інтеграцією ШІ-помічника для автоматичної генерації рекомендацій.

У роботі буде здійснено аналіз існуючих підходів до інтелектуальної підтримки операторів, проведено обґрунтування вибору програмно-технологічного стеку та розроблено прототип системи, що демонструє її функціональність на модельному тестовому сценарію

технологічного процесу.

Для розроблення прототипу системи необхідно формалізувати методи детекції аварійних ситуацій, розробити алгоритми інтеграції з сервісами штучного інтелекту та експериментально оцінити ефективність розробленого рішення порівняно з традиційним підходом ручного пошуку інформації.

Внесок у Цілі сталого розвитку: розроблення даного програмного модуля оперативно-диспетчерського контролю безпосередньо сприяє досягненню кількох Цілей сталого розвитку. Зокрема, він робить внесок у Ціль 9: Індустріалізація, інновації та інфраструктура, шляхом створення інноваційних рішень на базі штучного інтелекту для промислової автоматизації. Крім того, завдяки підвищенню швидкості реагування на аварійні ситуації, система підтримує Ціль 8: Гідна праця та економічне зростання, підвищуючи безпеку праці та знижуючи ймовірність помилок через людський фактор.

До основних задач, які вирішує система, відносяться такі ключові аспекти:

– швидке виявлення та класифікація аварійних ситуацій;

– автоматична генерація контекстних рекомендацій на основі ШІ;

– скорочення часу прийняття рішення оператором;

– зниження залежності від досвіду оператора;

– масштабованість та адаптивність до різних виробничих об'єктів.

Об'єктом дослідження є процес оперативно-диспетчерського контролю виробництва на базі кіберфізичних систем керування, що включає моніторинг технологічних параметрів, детекцію відхилень та підтримку прийняття рішень операторами.

Предметом дослідження є програмний модуль інтелектуальної підтримки оператора, методи інтеграції з сервісами штучного інтелекту, алгоритми генерації рекомендацій на основі технічної документації та принципи візуалізації в SCADA-інтерфейсі.

В роботі запропоновано програмний модуль

оперативно-диспетчерського контролю з інтеграцією ШІ-помічника. Розроблено методику порівняльної оцінки ефективності системи шляхом експериментального вимірювання часу прийняття рішення оператором у сценаріях до та після впровадження ШІ-підтримки. Модуль дозволяє автоматично генерувати контекстні рекомендації при виявленні аварійних ситуацій, що скорочує час реагування та знижує ймовірність помилок оператора. Експериментальна апробація на модельному тестовому сценарії технологічного процесу підтвердила ефективність запропонованого підходу, що можна віднести до отриманих наукових результатів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

– провести аналіз сучасних методів та технологій інтелектуальної підтримки операторів кіберфізичних систем та обґрунтувати вибір програмного стеку;

– обрати модельний об'єкт для апробації системи та формалізувати типові аварійні ситуації;

– розробити архітектуру програмного модуля з інтеграцією ШІ-помічника через REST API;

– реалізувати алгоритми моніторингу параметрів, детекції алармів та генерації рекомендацій;

– розробити концепцію SCADA-інтерфейсу з відображенням ШІ-рекомендацій;

– провести експериментальну оцінку ефективності системи шляхом порівняння часу прийняття рішення оператором до та після впровадження ШІ-підтримки;

– апробувати розроблений модуль на модельних сценаріях та проаналізувати результати;

– оформити кваліфікаційну роботу згідно вимог [3-4].

Результати роботи опубліковані в [1].

1 АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО АВТОМАТИЗАЦІЇ ОПЕРАТИВНО-ДИСПЕТЧЕРСЬКОГО КОНТРОЛЮ

Сучасне виробництво характеризується посиленням вимог до гнучкості, ефективності та прозорості технологічних процесів. У цьому контексті оперативно-диспетчерський контроль перестав бути завданням виключно пасивного моніторингу, перетворившись на складну проблему інтеграції апаратних засобів, програмного забезпечення та інтелектуальних систем підтримки прийняття рішень [5]. Даний розділ присвячено аналізу існуючих підходів до автоматизації оперативно-диспетчерського контролю, методів інтелектуальної підтримки операторів та обґрунтування вибору технологічного стеку для розробки системи.

1.1 Сучасні системи оперативно-диспетчерського контролю виробництва

Автоматизована система оперативно-диспетчерського контролю є складною ієрархічною розподіленою системою, що поєднує засоби нижчого (польового) та верхнього (диспетчерського) рівнів. Фундаментом такої системи виступають програмовані логічні контролери (ПЛК), які забезпечують виконання програм керування виконавчими механізмами та збір даних з датчиків [6-7].

Інформаційно-керуючим інтерфейсом між технологічним процесом і оператором слугують SCADA-системи (Supervisory Control And Data Acquisition) (рис. 1.1). SCADA забезпечує візуалізацію стану процесу в реальному часі, реєстрацію аварійних тривог, формування звітів і надання можливості оперативного втручання персоналу [8].

Рисунок 1.1 – Архітектура типової SCADA-системи

Ключовою проблемою існуючих SCADA-систем є те, що при виникненні аварійної ситуації оператор отримує лише сигнал тривоги, але не отримує автоматичних рекомендацій щодо дій. Оператор змушений самостійно шукати відповідні інструкції в технічній документації, що займає від 5 хвилин до 10 хвилин критичного часу [9].

1.2 Методи інтелектуальної підтримки операторів

1.2.1 Традиційні підходи

Історично першими спробами автоматизації підтримки прийняття рішень були експертні системи (Expert Systems), засновані на правилах продукції "ЯКЩО-ТО" [10]. Експертна система складається з бази знань (правила), механізму логічного виведення та інтерфейсу користувача.

Основним недоліком експертних систем є їх ригідність та складність підтримки. Додавання нових правил вимагає залучення інженерів зі знаннями, а база правил швидко стає надто великою та суперечливою [11].

Розвитком експертних систем стали системи підтримки прийняття

рішень (DSS), які інтегрують статистичні методи та оптимізацію [12]. Однак DSS також вимагають значних зусиль на етапі проектування.

1.2.2 Застосування штучного інтелекту на основі LLM

Останнім часом спостерігається стрімкий розвиток застосування великих мовних моделей (LLM) в промислових системах [13]. Великі мовні моделі – це нейронні мережі архітектури Transformer, навчені на величезних обсягах текстових даних, що дозволяє їм генерувати зв'язний текст, відповідати на запитання природною мовою (рис. 1.2).

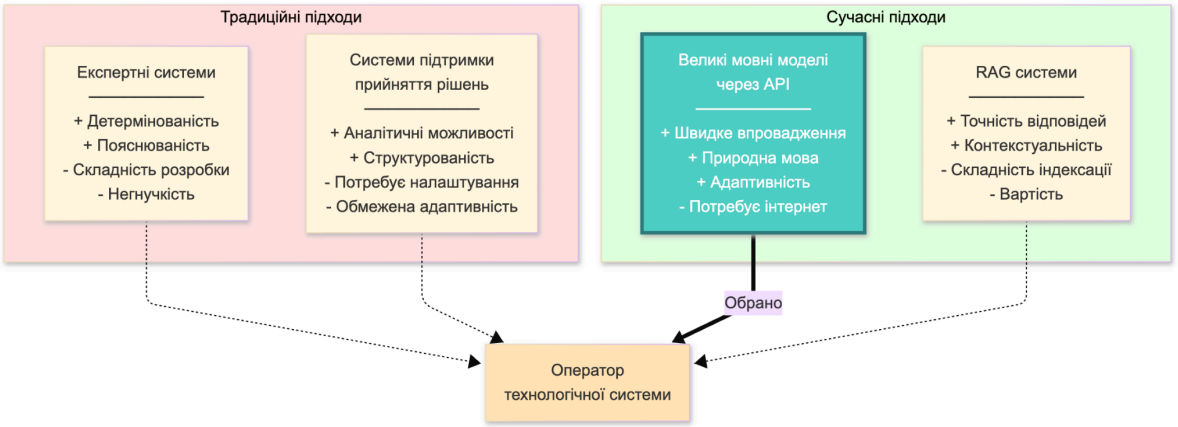

Рисунок 1.2 – Порівняння підходів до інтелектуальної підтримки операторів

Для промислового застосування LLM важливо забезпечити їх роботу з конкретною технічною документацією. У даній роботі використовується спрощений підхід Prompt Engineering, де релевантні фрагменти PDF інструкцій передаються безпосередньо в запит до LLM через API-сервіс.

1.3 Технології інтеграції ШІ через API-сервіси

Для інтеграції LLM у промислові системи використовуються API-сервіси, що надають доступ до потужних мовних моделей через стандартні веб-протоколи [14]. Основні переваги:

– не потрібні власні обчислювальні ресурси (GPU);

– швидка інтеграція через REST API;

– масштабованість та висока швидкість відгуку.

Groq API – це високопродуктивний API-сервіс для доступу до LLM, що використовує спеціалізоване апаратне прискорення для досягнення надзвичайно низької затримки генерації тексту [15]. Ключові характеристики:

– підтримка моделей: Mixtral-8x7B, LLaMA-2;

– швидкість генерації: до 500 токенів/с;

– REST API інтерфейс: стандартний HTTP запит з JSON форматом;

– час відгуку: 1-3 секунд для генерації рекомендації.

Приклад виклику Groq API:

```
pythonimport requests

def get_ai_recommendation(alarm_text, instruction_text):
                prompt = f"Ситуація: {alarm_text}\nІнструкція: {instruction_text}\nРекомендація:"

    response = requests.post('https://api.groq.com/v1/chat/completions',
      headers={'Authorization': 'Bearer API_KEY'},
      json={
        'model': 'mixtral-8x7b-32768',
        'messages': [{'role': 'user', 'content': prompt}],
        'temperature': 0.3
      })

    return response.json()['choices'][0]['message']['content']
```

1.4 Обґрунтування вибору технологічного стеку

Для реалізації програмного модуля обрано наступний технологічний стек:

а) Node-RED – візуальний інструмент для побудови потоків даних IoT та промислових систем [16]. Його переваги:

1) низькопороговий вхід (low-code платформа);
2) велика бібліотека вузлів для інтеграції з промисловими протоколами (Modbus, OPC UA, MQTT);
3) вбудовані можливості створення HMI (Node-RED Dashboard);
4) ефективна робота на edge-пристроях.

б) Python Flask – фреймворк для створення REST API [17]. Використовується як проміжний шар для інтеграції з LLM API. Його переваги:

1) простий та легковаговий;
2) підтримка бібліотек для роботи з PDF (PyPDF2);
3) легка інтеграція з LLM API.

На рисунку 1.3 представлена схема взаємодії основних компонентів системи.

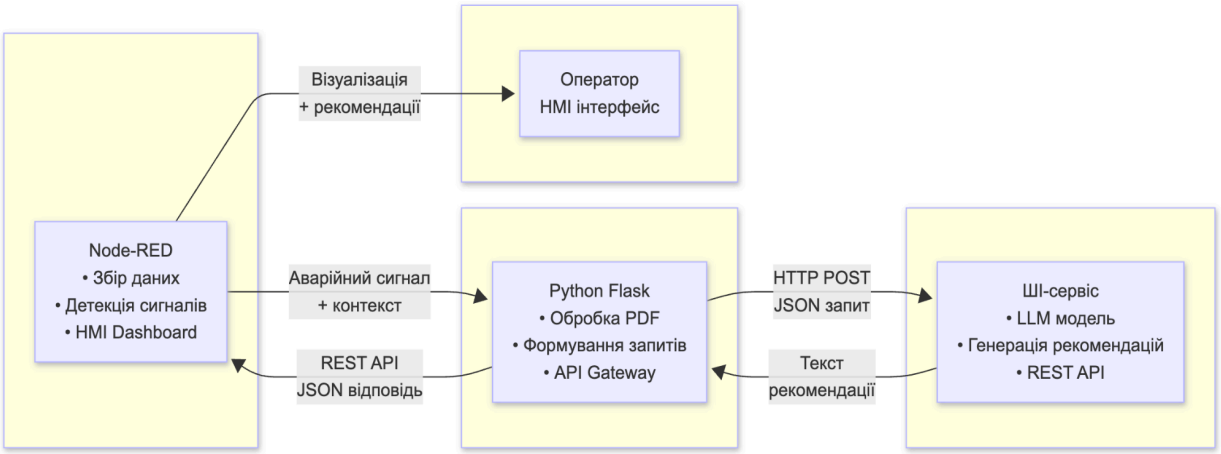

Рисунок 1.3 – Архітектура тестового програмного модуля з інтеграцією ШІ-помічника

Переваги обраної архітектури в тому, що модульна побудова дозволяє незалежно розвивати окремі компоненти системи. Використання стандартних протоколів (HTTP, JSON) забезпечує сумісність з різними промисловими системами. Веб-орієнтований підхід дозволяє операторам отримувати доступ до системи з будь-якого пристрою в локальній мережі. Відокремлення логіки ШІ в окремий Python-сервер спрощує майбутнє впровадження RAG-системи без зміни Node-RED потоків [16].

1.5 Висновки до розділу

У даному розділі проведено аналіз існуючих підходів до автоматизації оперативно-диспетчерського контролю та методів інтелектуальної підтримки операторів.

Встановлено, що традиційні SCADA-системи забезпечують ефективний моніторинг технологічних процесів, однак при виникненні аварійних ситуацій оператор змушений самостійно шукати рекомендації в технічній документації, що займає 5-10 хвилин критичного часу.

Класичні підходи до інтелектуальної підтримки (експертні системи, DSS) характеризуються високою складністю розробки та низькою гнучкістю. Перспективним напрямком є використання великих мовних моделей через API-сервіси для автоматичної генерації контекстних рекомендацій.

Обґрунтовано вибір технологічного стеку Node-RED + Python Flask + LLM API, що забезпечує швидку інтеграцію (1-2 тижні), час відгуку 2-3 секунд та якість природно-мовних відповідей.

У наступному розділі буде здійснено постановку задачі дослідження, обрано модельний об'єкт для апробації та формалізовано методику експериментальної оцінки ефективності системи.

2 ПОСТАНОВКА ЗАДАЧІ ТА ВИБІР ОБ'ЄКТА ДОСЛІДЖЕННЯ

Розробка програмного модуля оперативно-диспетчерського контролю з інтеграцією ШІ-помічника вимагає чіткого формулювання технічних вимог, обґрунтованого вибору параметрів для тестування системи та формалізації задачі детекції аварійних ситуацій. Даний розділ присвячено аналізу технічного завдання, визначенню тестового середовища та постановці наукової задачі експериментальної оцінки ефективності розробленої системи.

2.1 Аналіз технічного завдання та функціональні вимоги

Основною метою розроблення програмного модуля є підвищення оперативності та надійності прийняття рішень операторами кіберфізичних систем керування. Система повинна забезпечувати автоматичний моніторинг технологічних параметрів, детекцію відхилень від нормативних значень та генерацію контекстних рекомендацій для оператора на основі технічної документації.

Функціональні вимоги до системи включають можливість збору даних у реальному часі з датчиків технологічного процесу, аналіз поточних значень параметрів та порівняння їх з критичними порогами. При виявленні відхилення система повинна ідентифікувати тип аварійної ситуації, сформувати запит до ШІ-сервісу з релевантними фрагментами інструкцій та відобразити згенеровані рекомендації в інтерфейсі оператора протягом 2-3 с.

Нефункціональні вимоги визначають необхідність забезпечення високої надійності роботи системи в умовах виробництва, мінімальної затримки генерації рекомендацій, можливості інтеграції з існуючими SCADA-системами через стандартні протоколи обміну даними та простоту налаштування під конкретний технологічний процес. Система повинна працювати на стандартному промисловому обладнанні без необхідності

використання спеціалізованих обчислювальних ресурсів.

Важливою вимогою є модульність архітектури [18], що дозволяє використовувати розроблений програмний модуль як незалежний компонент у складі більш складних систем автоматизації виробництва. Інтерфейс взаємодії з іншими системами повинен базуватися на REST API, що забезпечує широку сумісність та можливість інтеграції з різними платформами моніторингу та керування.

2.2 Визначення тестового середовища та параметрів моніторингу

2.2.1 Обґрунтування вибору модельних параметрів

Для апробації розробленого програмного модуля необхідно визначити набір контрольованих параметрів, що є типовими для більшості кіберфізичних систем керування технологічними процесами. Вибір модельних параметрів зумовлений кількома факторами. По-перше, параметри повинні бути універсальними та зустрічатися в різних галузях промисловості. По-друге, для кожного параметра має існувати чітко визначений робочий діапазон та критичні межі, перевищення яких вказує на аварійну ситуацію.

Оскільки робота виконувалась на кафедрі комп’ютерно-інтегрованих технологій, автоматизації та робототехніки ХНУРЕ без виїзду на реальне виробництво, використання узагальнених модельних параметрів є методично обґрунтованим рішенням. Такий підхід дозволяє зосередитись на програмній реалізації системи підтримки прийняття рішень, не вдаючись в специфіку конкретного технологічного обладнання.

Додатковою перевагою використання універсальних параметрів є можливість застосування розробленого модуля в різних галузях промисловості без значних модифікацій. Система легко адаптується під конкретне виробництво шляхом зміни порогових значень та текстів інструкцій, зберігаючи при цьому незмінною програмну архітектуру та алгоритми роботи.

2.2.2 Опис контрольованих параметрів кіберфізичної системи

Для тестування системи обрано чотири ключові параметри, що є типовими для переважної більшості технологічних процесів у промисловості. Ці параметри забезпечують достатнє покриття можливих аварійних ситуацій та дозволяють продемонструвати функціональність розробленого програмного модуля [5].

Перший параметр – температура процесу (T), який є одним з найважливіших показників стану технологічної системи. Температура впливає на швидкість хімічних реакцій, фізичні властивості матеріалів та ефективність теплообміну. Контроль температури необхідний в енергетиці, хімічній промисловості, металургії та багатьох інших галузях. Відхилення температури від робочого діапазону може свідчити про порушення технологічного режиму, несправність обладнання або виникнення аварійної ситуації.

Другий параметр – тиск у системі (P), який характеризує механічний стан технологічної системи та визначає умови протікання багатьох процесів. Підвищення тиску понад допустимі межі створює ризик руйнування обладнання та трубопроводів, тоді як зниження тиску може вказувати на витік робочого середовища або недостатню продуктивність нагнітального обладнання. Контроль тиску критично важливий для систем з замкненим контуром циркуляції робочого середовища.

Третій параметр – рівень заповнення (L), що визначає кількість робочого середовища в технологічних ємностях та апаратах. Підтримання рівня в заданих межах необхідно для забезпечення безперервності технологічного процесу та запобігання пошкодженню обладнання. Зниження рівня нижче критичної позначки може призвести до порушення циркуляції, кавітації насосів або перегріву теплообмінних поверхонь.

Четвертий параметр – витрата робочого середовища (F), яка характеризує інтенсивність технологічного процесу та ефективність

використання ресурсів. Відхилення витрати від номінального значення може вказувати на порушення налаштувань системи керування, несправність виконавчих механізмів або витік робочого середовища. Контроль витрати важливий для забезпечення економічної ефективності роботи системи (рис. 2.1).

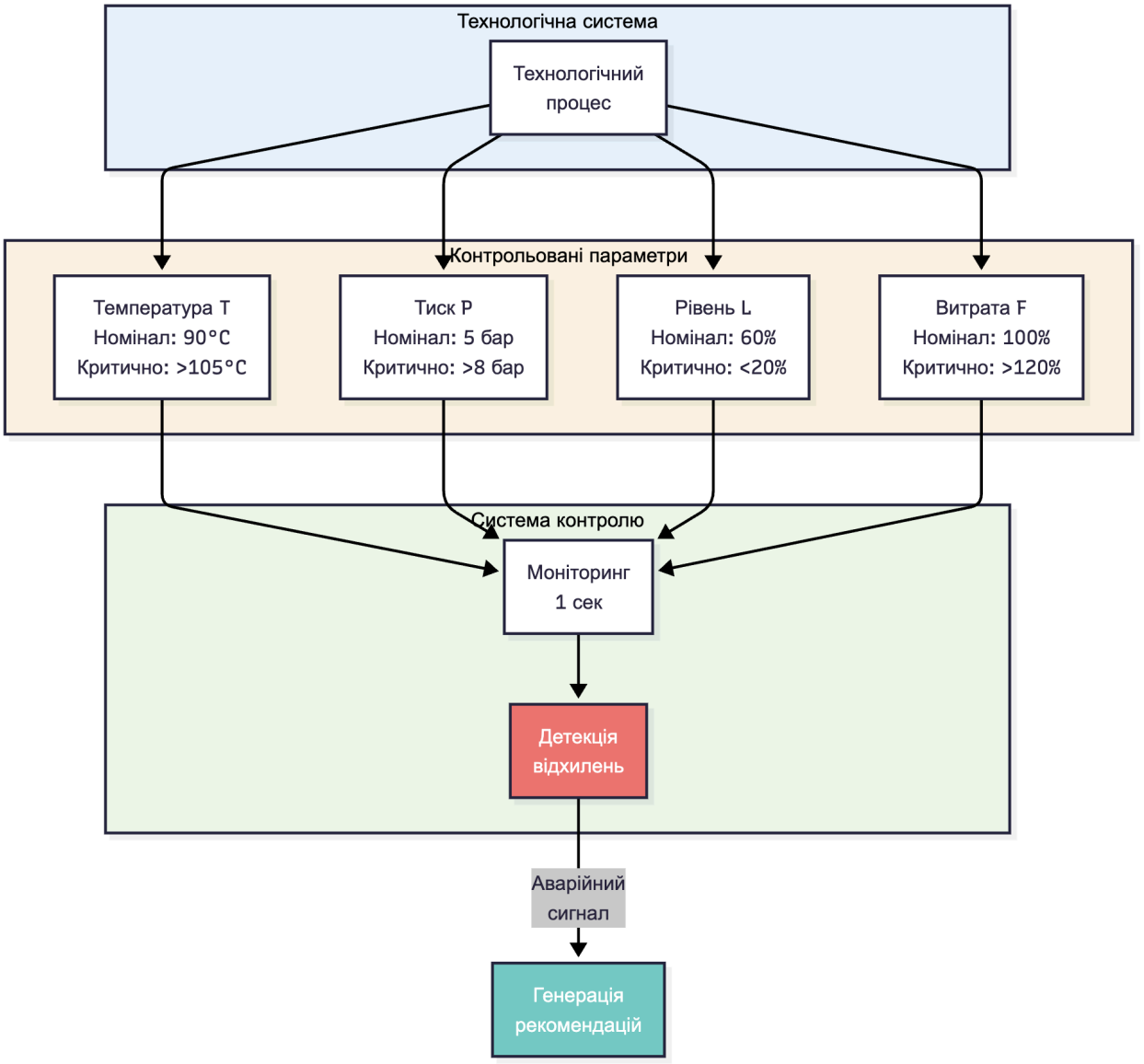

Рисунок 2.1 – Схема контрольованих параметрів технологічної системи

2.2.3 Критичні значення та пороги спрацювання

Для кожного з контрольованих параметрів визначено робочий діапазон та критичні пороги, при досягненні яких система генерує аварійний сигнал та

активує ШІ-помічника для формування рекомендацій оператору. Встановлення порогових значень базується на типових вимогах до технологічних процесів та практиці експлуатації промислових систем [15].

Температура процесу має робочий діапазон, в межах якого забезпечується нормальне функціонування системи. Для модельної системи встановлено номінальне значення температури 90 °C з допустимим відхиленням ±5 °C. Попереджувальний поріг встановлено на рівні 100 °C, що вказує на початок відхилення від нормального режиму. Критичним порогом вважається температура 105 °C, при досягненні якої необхідне термінове втручання оператора для запобігання аварійній ситуації.

Тиск у системі підтримується в діапазоні 4 - 6 бар при номінальному значенні 5 бар. Попереджувальні пороги встановлено на рівні 3 бар для нижньої межі та 7 бар для верхньої межі. Критичними значеннями є тиск нижче 2 бар, що може свідчити про значний витік робочого середовища, та тиск понад 8 бар, що створює ризик руйнування обладнання. При досягненні критичних значень система повинна не лише інформувати оператора, а й надавати конкретні рекомендації щодо дій.

Рівень заповнення контролюється у відносних одиницях або відсотках від повного обсягу ємності. Робочий діапазон встановлено в межах 40-80% заповнення, що забезпечує достатній запас робочого середовища без ризику переповнення. Попереджувальні пороги встановлено на рівні 30% для нижньої межі та 90% для верхньої межі. Критичним вважається рівень нижче 20%, що може призвести до порушення роботи обладнання, та рівень понад 95%, що створює ризик переповнення.

Витрата робочого середовища оцінюється відносно номінального значення, встановленого для даного режиму роботи. Попереджувальний поріг встановлено на рівні ±15% відхилення від номіналу. Критичним вважається відхилення витрати більше ніж на 25% від номінального значення, що може свідчити про серйозну несправність системи або значний витік.

2.3 Формалізація задачі детекції та реагування на аварійні ситуації

2.3.1 Тестові сценарії аварійних ситуацій

На основі визначених контрольованих параметрів розроблено п'ять тестових сценаріїв аварійних ситуацій, що використовуються для перевірки функціональності розробленого програмного модуля та проведення експериментальної оцінки його ефективності.

Перший сценарій моделює ситуацію критичного перевищення температури, коли параметр T досягає значення 108 °C при номінальному значенні 90 °C. Така ситуація може виникнути внаслідок порушення теплового балансу системи, недостатнього відведення теплоти або несправності системи охолодження. Оператору необхідно швидко визначити причину перегріву та вжити заходів для зниження температури, керуючись відповідними розділами експлуатаційної документації.

Другий сценарій описує аварійне підвищення тиску до 8,5 бар при номінальному значенні 5 бар. Підвищення тиску може бути наслідком блокування випускних трубопроводів, несправності запобіжної арматури або інтенсивного нагрівання робочого середовища в замкненому обсязі. Дана ситуація вимагає негайного реагування оператора для запобігання руйнуванню обладнання та забезпечення безпеки персоналу.

Третій сценарій характеризується критичним зниженням рівня заповнення до 18% при робочому діапазоні 40-80%. Зниження рівня може відбутися через витік робочого середовища, несправність системи живлення або підвищене споживання. Недостатній рівень створює ризик порушення циркуляції та пошкодження обладнання через недостатнє охолодження або змащування.

Четвертий сценарій моделює виявлення витоку робочого середовища, що проявляється у одночасному зниженні тиску до 2,5 бар та рівня до 35%. Комбінація цих відхилень з високою ймовірністю вказує на розгерметизацію системи. Оператору необхідно локалізувати місце витоку, оцінити його масштаб та вжити заходів для його усунення відповідно до інструкцій з

ліквідації аварійних ситуацій.

П'ятий сценарій описує ситуацію надмірної витрати робочого середовища, коли параметр F перевищує номінальне значення на 28%. Підвищена витрата при незмінному режимі роботи може свідчити про витік через нещільності з'єднань, несправність регулюючої арматури або неправильне налаштування системи автоматичного керування. Така ситуація призводить до економічних втрат та може вказувати на приховані несправності обладнання.

2.3.2 Алгоритм виявлення відхилень параметрів

Алгоритм детекції аварійних ситуацій базується на безперервному порівнянні поточних значень контрольованих параметрів з встановленими пороговими значеннями. Система опитує датчики з інтервалом 1 с, що забезпечує своєчасне виявлення швидкоплинних процесів. Отримані значення фільтруються для усунення випадкових викидів та перешкод вимірювального тракту.

При виявленні перевищення порогового значення будь-яким параметром система ідентифікує тип аварійної ситуації на основі комбінації відхилень (рис. 2.2). Для кожного типу ситуації визначено відповідний розділ інструкції з експлуатації, що містить рекомендації щодо дій оператора. Система формує структурований запит до ШІ-сервісу, що включає опис поточної ситуації, значення критичних параметрів та текст релевантного розділу інструкції.

Отримана від ШІ-сервісу рекомендація аналізується на коректність формату, після чого відображається в інтерфейсі [2] оператора разом з поточними значеннями параметрів та індикацією критичності ситуації. Час від моменту виявлення відхилення до відображення рекомендації не повинен перевищувати 3 с, що забезпечує своєчасність інформування оператора.

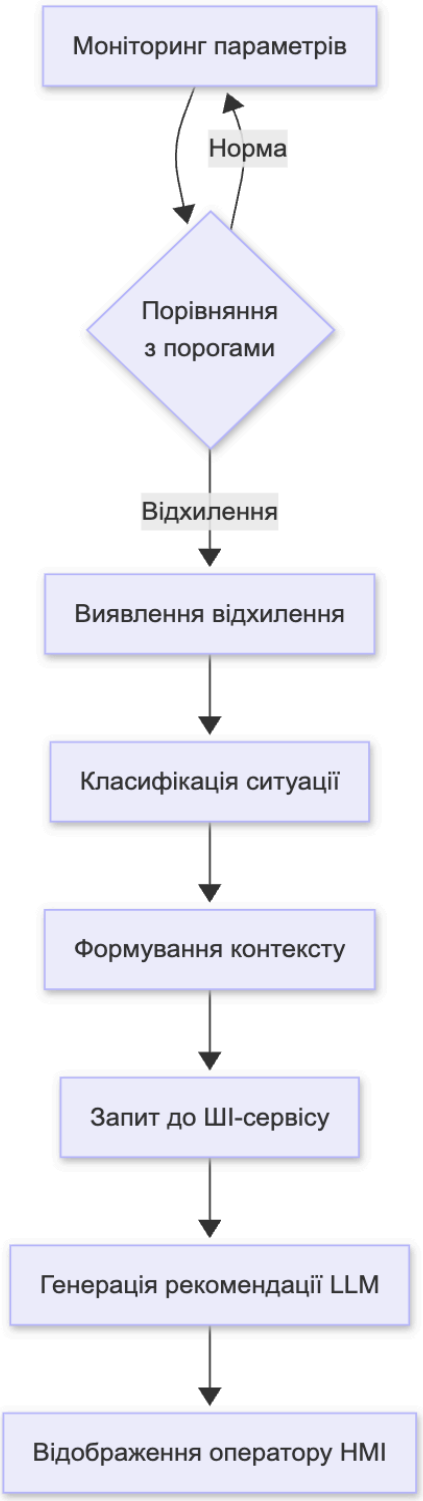

Рисунок 2.2 – Алгоритм детекції та генерації рекомендацій

2.4 Постановка наукової задачі експериментальної оцінки ефективності

Наукова складова даної роботи полягає в експериментальній оцінці ефективності розробленого програмного модуля через порівняльний аналіз часу прийняття рішення оператором до та після впровадження ШІ-помічника. Гіпотеза дослідження полягає в тому, що автоматична генерація контекстних рекомендацій на основі технічної документації значно скорочує час, необхідний оператору для знаходження правильних дій у аварійній ситуації.

Методика експериментального дослідження включає тестування групи з 5 студентів кафедри автоматизації та комп'ютерно-інтегрованих технологій на п'яти підготовлених сценаріях аварійних ситуацій. Кожен учасник експерименту виконує завдання у двох режимах: спочатку без використання ШІ-помічника (ручний пошук у PDF-інструкції), потім з використанням розробленої системи.

У режимі ручного пошуку учаснику надається письмовий опис аварійної ситуації з конкретними значеннями параметрів та PDF-файл з інструкцією обсягом 5 сторінок, що містить процедури реагування на всі п'ять типів ситуацій. Фіксується час від моменту ознайомлення з описом ситуації до моменту, коли учасник знаходить відповідний розділ інструкції та формулює послідовність необхідних дій. Вимірювання часу здійснюється за допомогою соміра з точністю до 1 с.

У режимі з ШІ-помічником учаснику демонструється інтерфейс розробленої системи, де відображається той самий сценарій аварійної ситуації з поточними значеннями параметрів. Система автоматично генерує та відображає рекомендацію щодо дій. Фіксується час від моменту відображення сценарію на екрані до появи повної рекомендації. Цей час характеризує швидкість реакції розробленої системи та включає всі етапи обробки інформації.

Для кожного учасника експерименту виконується серія з п'яти тестів (по одному на кожен сценарій) в обох режимах. Порядок пред'явлення

сценаріїв рандомізується для усунення ефекту навчання. Між тестами в режимі ручного пошуку робиться пауза 2-3 хвилини для відновлення концентрації уваги учасника. Загальна тривалість експерименту для одного учасника становить приблизно 45-60 хвилин.

Очікуваний результат полягає в демонстрації суттєвого скорочення часу прийняття рішення при використанні ШІ-помічника. Попередні оцінки показують, що ручний пошук у документації займає в середньому 5-10 хвилин залежно від складності ситуації та досвідченості учасника, тоді як система з ШІ-помічником надає рекомендацію протягом 2-3 секунд незалежно від досвідченості користувача. Це дає теоретичне покращення швидкості прийняття рішення приблизно у 100-200 разів, що є значущим результатом для практичного застосування.

Статистична обробка результатів включає обчислення середнього часу та стандартного відхилення для кожного режиму роботи та кожного сценарію. Для оцінки статистичної значущості різниці між режимами застосовується парний t-критерій Стьюдента. Будуються порівняльні діаграми, що наочно демонструють скорочення часу прийняття рішення. Розраховується коефіцієнт покращення ефективності як відношення середнього часу ручного пошуку до середнього часу роботи з ШІ-помічником.

Результати експерименту дозволять кількісно підтвердити доцільність впровадження розробленого програмного модуля в реальних системах оперативно-диспетчерського контролю. Крім того, експериментальні дані надають матеріал для аналізу залежності ефективності системи від складності сценарію та характеристик експлуатаційної документації, що може бути використано для подальшого удосконалення алгоритмів роботи ШІ-помічника.

2.5 Висновки до розділу

У даному розділі здійснено постановку задачі дослідження та визначено параметри тестового середовища для апробації розробленого програмного модуля.

Сформульовано функціональні та нефункціональні вимоги до системи оперативно-диспетчерського контролю з інтеграцією ШІ-помічника. Обґрунтовано вибір чотирьох універсальних параметрів для моніторингу: температура, тиск, рівень та витрата. Визначено робочі діапазони та критичні порогові значення для кожного параметра.

Розроблено п'ять тестових сценаріїв аварійних ситуацій: критичне перевищення температури, аварійне підвищення тиску, зниження рівня, виявлення витоку та надмірна витрата робочого середовища. Формалізовано алгоритм детекції відхилень параметрів та генерації рекомендацій з часом реакції системи не більше 3 с.

Поставлено наукову задачу експериментальної оцінки ефективності системи через порівняльний аналіз часу прийняття рішення оператором до та після впровадження ШІ-помічника. Розроблено методику експериментального дослідження з залученням 5 учасників, очікуване покращення ефективності становить 100-200 разів.

У наступному розділі буде розроблено архітектуру програмного модуля та описано алгоритми його функціонування.

3 РОЗРОБЛЕННЯ АРХІТЕКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО МОДУЛЯ

Проектування архітектури програмного модуля оперативно-диспетчерського контролю з інтеграцією ШІ-помічника вимагає врахування вимог до модульності, масштабованості та надійності системи. Даний розділ присвячено розробленню структури програмного модуля, опису його компонентів, алгоритмів функціонування та принципів організації взаємодії з ШІ-сервісом для генерації контекстних рекомендацій.

3.1 Архітектура програмного модуля

Архітектура розробленого програмного модуля базується на трирівневій структурі, що забезпечує чітке розділення відповідальності між компонентами системи та дозволяє незалежно модифікувати окремі частини без впливу на функціонування системи в цілому. Така організація відповідає сучасним принципам побудови розподілених систем та забезпечує можливість масштабування за необхідності.

Перший рівень архітектури представлено платформою Node-RED [16], що виконує функції збору даних з датчиків технологічного процесу, первинної обробки сигналів, детекції відхилень від нормативних значень та візуалізації інформації для оператора. Node-RED обрано завдяки його візуальному підходу до програмування потоків даних, широкій бібліотеці готових компонентів для роботи з промисловими протоколами та вбудованим можливостям створення інтерфейсів користувача через Node-RED Dashboard [19].

Другий рівень представлено сервером на базі Python Flask, що виступає як проміжний шар між Node-RED та ШІ-сервісом. Flask-сервер відповідає за завантаження та обробку PDF-документів з експлуатаційними інструкціями,

пошук релевантних фрагментів документації відповідно до типу аварійної ситуації, формування структурованих запитів до API великих мовних моделей та обробку отриманих відповідей. Використання Python обумовлено наявністю потужних бібліотек для роботи з документами та простотою інтеграції з різними API-сервісами [20].

Третій рівень представлено зовнішнім ШІ-сервісом, що надає доступ до великих мовних моделей через REST API. На цьому рівні відбувається аналіз контексту аварійної ситуації, обробка текстових фрагментів інструкцій та генерація природномовних рекомендацій для оператора. Використання зовнішнього сервісу дозволяє уникнути необхідності розгортання власної інфраструктури для роботи з LLM, що вимагає значних обчислювальних ресурсів.

Взаємодія між рівнями архітектури здійснюється через REST API з використанням JSON формату для передачі даних (рис. 3.1). Такий підхід забезпечує слабку зв'язаність компонентів та можливість заміни будь-якого рівня без необхідності модифікації інших частин системи. Наприклад, Node-RED може бути замінено на іншу SCADA-платформу, а ШІ-сервіс може бути замінено на альтернативну реалізацію LLM без зміни Flask-сервера.

Важливим аспектом архітектури є забезпечення відмовостійкості системи. У випадку недоступності ШІ-сервісу через проблеми з мережею або перевантаження API, Flask-сервер повертає попередньо підготовлене повідомлення з посиланням на відповідний розділ інструкції. Це гарантує, що оператор завжди отримає мінімально необхідну інформацію для реагування на аварійну ситуацію, навіть якщо автоматична генерація детальних рекомендацій тимчасово недоступна.

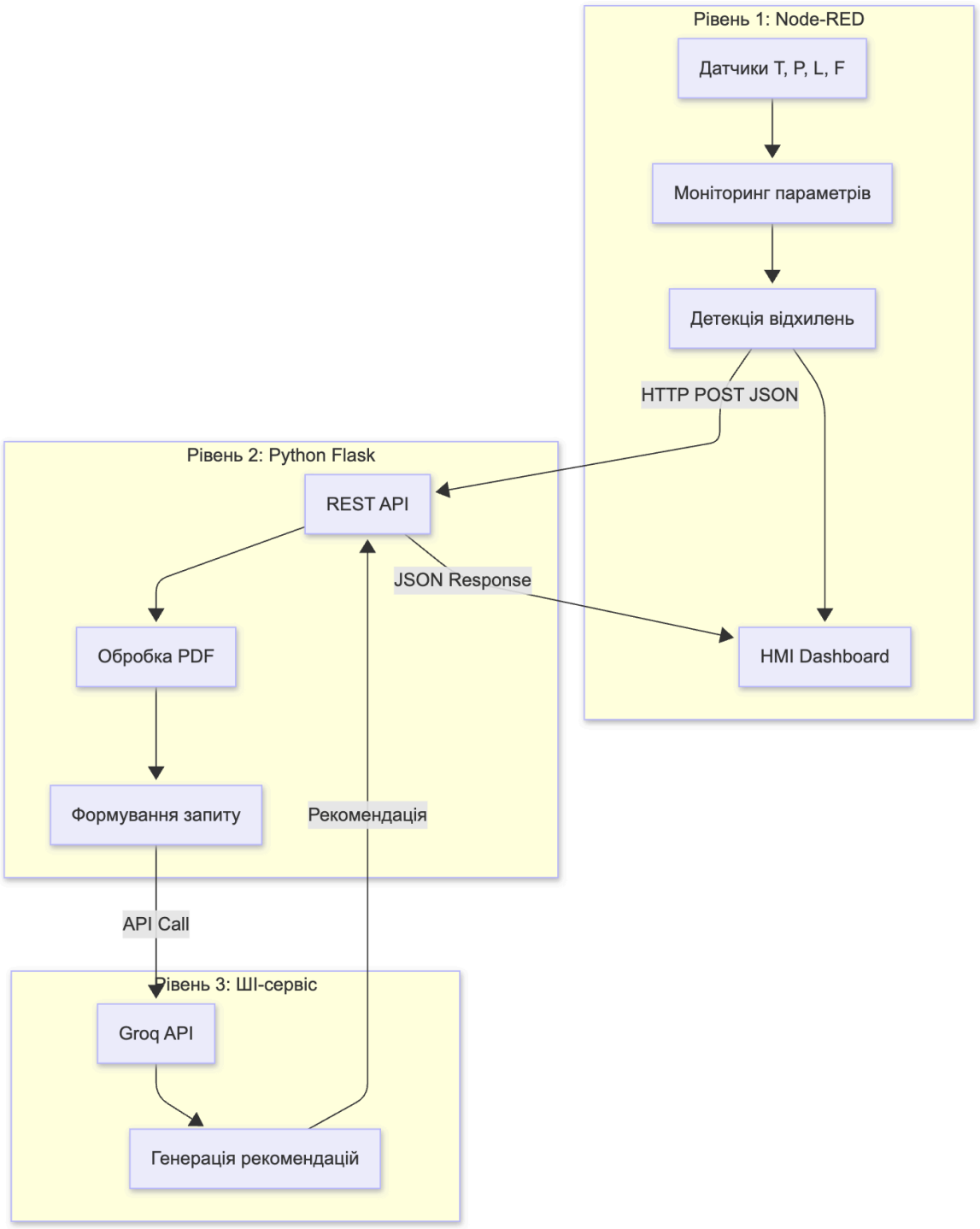

Рисунок 3.1 – Архітектура програмного модуля

3.2 Компоненти системи та їх взаємодія

Платформа Node-RED на першому рівні архітектури організовує потоки даних від датчиків до інтерфейсу оператора. Вузли введення даних можуть отримувати інформацію з різних джерел: через протоколи Modbus,

OPC UA, MQTT або через імітацію даних для тестування системи [21]. Отримані значення параметрів зберігаються в контексті потоку та передаються до вузлів обробки, що виконують фільтрацію сигналів та перевірку на відповідність пороговим значенням.

Детекція відхилень реалізована через функціональні вузли Node-RED, що виконують порівняння поточних значень параметрів з критичними порогами [16]. При виявленні перевищення порогу формується об'єкт з описом аварійної ситуації, що включає тип ситуації, поточні значення всіх контрольованих параметрів та часову мітку події. Цей об'єкт передається до вузла HTTP-запиту, що надсилає дані на Flask-сервер для подальшої обробки.

Node-RED Dashboard використовується для створення веб-інтерфейсу оператора, що відображає поточні значення параметрів у вигляді числових індикаторів та графіків тренду. При виникненні аварійної ситуації на панелі з'являється виділена область з описом проблеми та згенерованою рекомендацією [2]. Колірне кодування (зелений, жовтий, червоний) використовується для швидкої візуальної оцінки стану системи оператором.

Flask-сервер на другому рівні архітектури приймає HTTP POST запити від Node-RED з JSON-об'єктами, що описують аварійні ситуації. Структура запиту включає поля для типу ситуації, значень параметрів та часової мітки. Сервер розбирає отриманий JSON та визначає, який розділ PDF-інструкції відповідає даному типу ситуації. Для кожного типу аварійної ситуації заздалегідь визначено номер сторінки або маркер розділу в PDF-документі.

Обробка PDF-документів здійснюється за допомогою бібліотеки PyPDF2, що дозволяє витягувати текстовий вміст зі сторінок документа. Для кожної аварійної ситуації витягується відповідний фрагмент тексту обсягом 200-500 слів, що містить опис проблеми та рекомендовані дії оператора. Витягнутий текст очищається від зайвих символів форматування та об'єднується з описом поточної ситуації для формування запиту до ШІ-сервісу.

Запит до API великих мовних моделей формується як JSON-об'єкт з

полями model, messages та параметрами генерації. Поле messages містить масив повідомлень, де перше повідомлення включає системну інструкцію для моделі, а друге містить опис аварійної ситуації разом з фрагментом інструкції. Параметр temperature встановлюється на низьке значення (0,2-0,3) для забезпечення детермінованості відповідей, а max_tokens обмежує довжину генерованої рекомендації до 300-500 токенів.

Приклад структури запиту до ШІ-сервісу:

```
{
 "model": "mixtral-8x7b-32768",
 "messages": [
  {
   "role": "system",
    "content": "Ти асистент оператора технологічної системи. Аналізуй ситуацію та надавай чіткі покрокові рекомендації."
  },
  {
   "role": "user",
    "content": "Ситуація: Температура 108 °C (критично). Тиск 5,2 бар (норма). Рівень 65% (норма).\n\nІнструкція: При перевищенні температури понад 105 °C необхідно: 1) Зменшити навантаження системи... 2) Збільшити інтенсивність охолодження... 3) Перевірити справність датчиків...\n\nНадай конкретні рекомендації оператору."
  }
 ],
 "temperature": 0.3,
 "max_tokens": 400
}
```

Отримана відповідь від ШІ-сервісу розбирається Flask-сервером, з неї витягується текст рекомендації та формується JSON-відповідь для Node-RED. У випадку помилки при зверненні до API або перевищення часу очікування відповіді, сервер повертає резервне повідомлення з базовою інформацією про необхідні дії. Це забезпечує надійність системи навіть при тимчасових проблемах з доступністю зовнішніх сервісів.

Взаємодія компонентів організована як асинхронний процес, що дозволяє Node-RED продовжувати моніторинг параметрів під час очікування відповіді від Flask-сервера. Таймаут для HTTP-запитів встановлено на рівні 5 с, що забезпечує баланс між надійністю та швидкістю реакції системи. Якщо відповідь не отримано протягом таймауту, оператор інформується про необхідність звернутися до документації вручну.

3.3 Алгоритми роботи програмного модуля

Алгоритм моніторингу та детекції відхилень виконується циклічно з періодом опитування 1 с. На кожній ітерації циклу система зчитує поточні значення всіх контрольованих параметрів та застосовує фільтр ковзного середнього для згладжування випадкових коливань. Фільтр використовує вікно з 5 останніх значень для кожного параметра, що дозволяє усунути короткочасні викиди, спричинені електромагнітними перешкодами або нестабільністю датчиків.

Після фільтрації кожне значення параметра порівнюється з відповідними пороговими значеннями. Система підтримує два рівні порогів: попереджувальний та критичний. При перевищенні попереджувального порогу генерується попередження з жовтою індикацією на панелі оператора, але рекомендації не запитуються. При перевищенні критичного порогу генерується аварійний сигнал з червоною індикацією та ініціюється процес формування рекомендацій.

Класифікація типу аварійної ситуації базується на аналізі комбінації

відхилень різних параметрів (рис. 3.2). Наприклад, одночасне зниження тиску та рівня заповнення з високою ймовірністю вказує на витік робочого середовища, тоді як ізольоване підвищення температури може свідчити про проблеми з охолодженням. Для кожної комбінації відхилень визначено відповідний тип ситуації та ідентифікатор розділу інструкції.

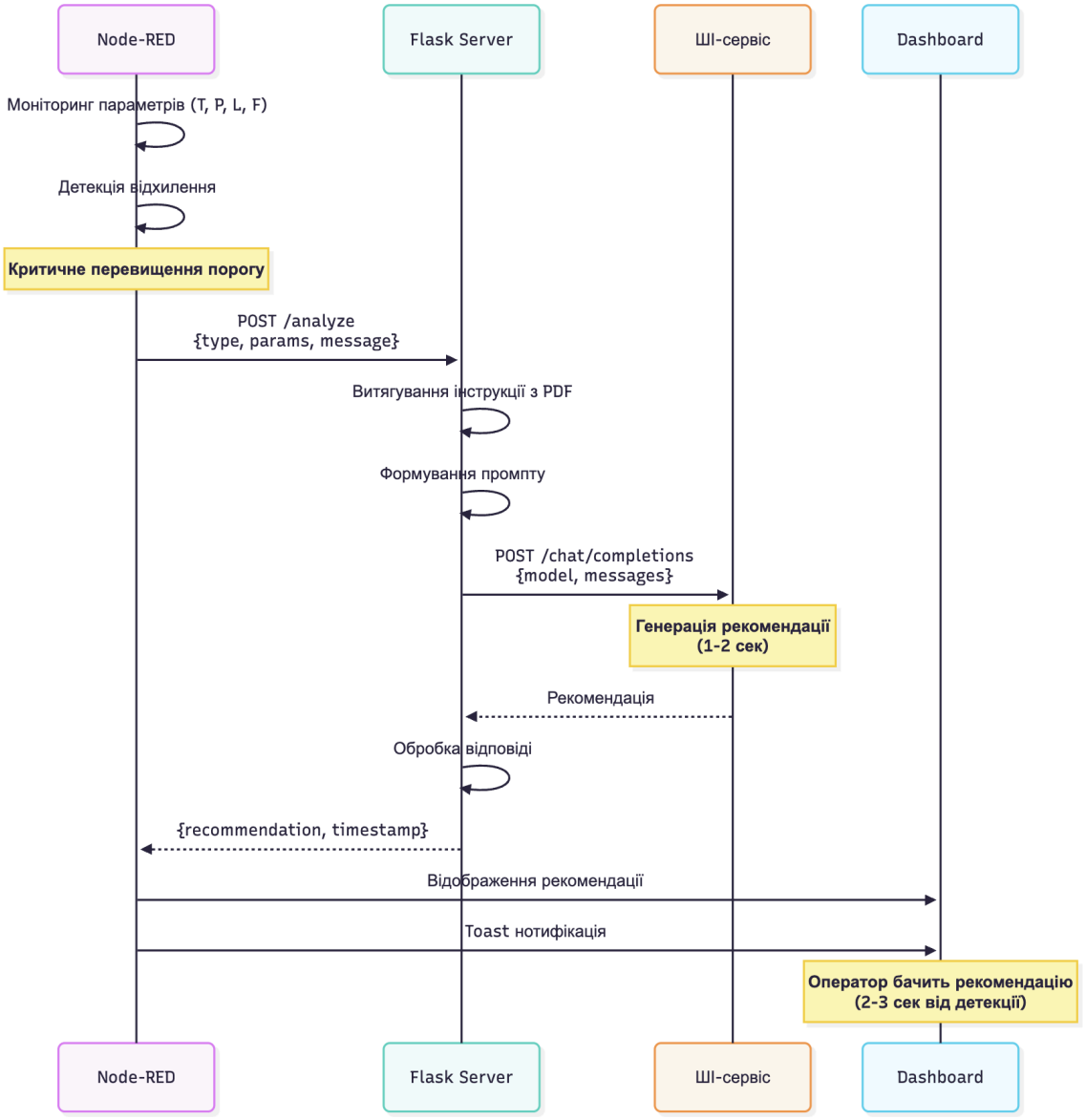

Рисунок 3.2 – Послідовність обробки аварійного сигналу

Алгоритм формування запиту до ШІ-сервісу (рис. 3.2) починається з отримання Flask-сервером HTTP POST запиту від Node-RED. Сервер витягує

з JSON-об'єкта тип ситуації та поточні значення параметрів. На основі типу ситуації визначається відповідний розділ PDF-інструкції, з якого витягається текстовий фрагмент обсягом 200-500 слів. Якщо інструкція містить діаграми або таблиці, вони описуються текстом для забезпечення можливості обробки моделлю.

Формування промпту для LLM включає три компоненти: системну інструкцію, що визначає роль моделі як асистента оператора, опис поточної ситуації з конкретними значеннями параметрів та фрагмент експлуатаційної інструкції з рекомендованими діями. Системна інструкція налаштовує модель на генерацію структурованих покрокових рекомендацій у чіткому та лаконічному форматі без зайвих пояснень або коментарів.

Опис ситуації формується як структурований текст, що містить перелік параметрів з їх поточними значеннями та позначками про відповідність нормі. Наприклад: "Температура: 108 °C (КРИТИЧНО, норма 85 - 95 °C). Тиск: 5,2 бар (норма). Рівень: 65% (норма). Витрата: номінальна." Такий формат дозволяє моделі швидко ідентифікувати критичні параметри та зосередитися на релевантних діях з інструкції.

Після відправлення запиту Flask-сервер очікує відповідь від ШІ-сервісу з таймаутом 4 с. Отримана відповідь перевіряється на наявність текстового вмісту та коректність формату. Якщо відповідь містить рекомендацію, вона витягується з JSON-структури та форматується для відображення в інтерфейсі оператора. Текст рекомендації може бути додатково оброблений для видалення службових маркерів або форматування, якщо модель додала їх всупереч інструкціям.

У випадку відсутності відповіді від ШІ-сервісу протягом таймауту або отримання повідомлення про помилку, Flask-сервер генерує резервну відповідь на основі попередньо підготовленого шаблону для даного типу ситуації. Резервна відповідь містить базові рекомендації та посилання на відповідний розділ інструкції для ручного ознайомлення. Це гарантує, що оператор завжди отримає якусь інформацію для прийняття рішення, навіть

якщо автоматична генерація недоступна.

Фінальна відповідь Flask-сервера до Node-RED має JSON формат з полями status (success або error), recommendation (текст рекомендації), timestamp (часова мітка) та source (тип джерела: ai або fallback). Node-RED отримує цю відповідь та відображає рекомендацію на панелі оператора разом з поточними значеннями параметрів та індикацією типу ситуації. Час від виявлення відхилення до відображення рекомендації складає 2-3 секунд в нормальних умовах.

3.4 Організація бази знань та генерація рекомендацій

База знань системи організована у вигляді PDF-документа обсягом 5 сторінок, що містить структуровані інструкції з реагування на п'ять типів аварійних ситуацій. Кожен розділ документа присвячено окремому типу ситуації та включає опис проблеми, можливі причини її виникнення, покрокову процедуру усунення та запобіжні заходи. Структурування документа забезпечує можливість швидкого пошуку релевантного фрагменту на основі типу виявленої ситуації.

Метод Prompt Engineering використовується для забезпечення якісної генерації рекомендацій без необхідності дообучення або fine-tuning великої мовної моделі [22]. Ключовими елементами промпту є чітке визначення ролі моделі, структурований опис ситуації та конкретні інструкції щодо формату відповіді. Системне повідомлення встановлює контекст: модель виступає як досвідчений асистент оператора, що надає лаконічні та практичні рекомендації на основі експлуатаційної документації.

Формат запиту побудовано таким чином, щоб модель отримувала всю необхідну інформацію в єдиному контексті. Спочатку надається опис поточного стану системи з конкретними значеннями параметрів, що дозволяє моделі зрозуміти масштаб та серйозність ситуації. Потім надається текстовий фрагмент з інструкції, що містить загальні рекомендації для даного типу

проблеми. Нарешті, формулюється конкретне завдання: синтезувати рекомендацію, що враховує як поточний стан системи, так і інформацію з інструкції.

Параметри генерації налаштовуються для забезпечення балансу між точністю та природністю відповідей. Параметр temperature встановлено на значення 0,3, що забезпечує детермінованість та фокусування на найбільш ймовірних продовженнях тексту. Низьке значення temperature зменшує ймовірність галюцинацій та незв'язних рекомендацій. Параметр max_tokens обмежує довжину відповіді до 400-500 токенів, що відповідає 300-400 словам природного тексту.

Приклад згенерованої рекомендації для ситуації критичного перевищення температури:

– негайно зменшити навантаження системи на 30-40% через зменшення витрати робочого середовища;

– збільшити інтенсивність охолодження: перевірити роботу циркуляційних насосів та відкрити додаткові контури охолодження (якщо доступно);

– моніторити температуру кожні 30 с. Якщо через 2 хвилини t > 106 °C — аварійна зупинка;

– перевірити справність датчиків температури для виключення хибного спрацювання;

– викликати технічний персонал для діагностики системи охолодження.

Якість згенерованих рекомендацій залежить від повноти та структурованості вихідної інструкції. Інструкції повинні містити конкретні числові значення, часові рамки дій та чіткі критерії прийняття рішень. Нечіткі формулювання типу "при необхідності" або "якщо потрібно" знижують якість синтезованих рекомендацій, оскільки модель не має достатньо інформації для конкретизації дій.

3.5 Проектування інтерфейсу оператора

Інтерфейс оператора розроблено з використанням Node-RED Dashboard та організовано у вигляді веб-сторінки, доступної через браузер на робочій станції оператора або мобільному пристрої. Головна панель відображає поточні значення всіх контрольованих параметрів у вигляді числових індикаторів з колірним кодуванням: зелений колір вказує на нормальний діапазон, жовтий – на наближення до попереджувального порогу, червоний – на критичне відхилення [2, 23].

Графіки трендів розміщено під числовими індикаторами та відображають зміну кожного параметра протягом останніх 10-15 хвилин. Це дозволяє оператору візуально оцінити динаміку процесу та передбачити можливі проблеми до спрацювання аварійних порогів. Горизонтальні лінії на графіках позначають попереджувальні та критичні пороги для зручності візуальної оцінки близькості поточного значення до небезпечної зони.

При виникненні аварійної ситуації в верхній частині інтерфейсу з'являється виділена панель з описом проблеми, поточними значеннями критичних параметрів та згенерованою рекомендацією. Панель має червоний фон для привернення уваги оператора та містить кнопку підтвердження ознайомлення з рекомендацією. Після підтвердження панель залишається видимою, але переміщується в нижню частину екрану для збереження доступу до історії подій.

Рекомендація відображається у структурованому форматі з нумерованими пунктами дій, що полегшує сприйняття інформації в стресовій ситуації. Ключові слова та числові значення виділяються напівжирним шрифтом для швидкого візуального сканування тексту. Час генерації рекомендації відображається поруч з текстом, що дозволяє оператору оцінити актуальність інформації.

3.6 Висновки до розділу

У даному розділі розроблено архітектуру та алгоритми функціонування програмного модуля оперативно-диспетчерського контролю з інтеграцією ШІ-помічника.

Запропоновано трирівневу архітектуру системи з чітким розділенням відповідальності: Node-RED для збору даних та візуалізації, Python Flask для обробки документації та взаємодії з API, ШІ-сервіс для генерації рекомендацій. Взаємодія між рівнями організована через REST API з використанням JSON формату даних.

Розроблено алгоритми моніторингу параметрів з фільтрацією сигналів, детекції відхилень на основі порогових значень, класифікації типу аварійної ситуації та формування запитів до ШІ-сервісу. Час реакції системи від виявлення відхилення до відображення рекомендації становить 2-3 с.

Організовано базу знань у вигляді структурованого PDF-документа та застосовано метод Prompt Engineering для забезпечення якісної генерації рекомендацій без дообучення моделі. Спроектовано інтерфейс оператора з колірним кодуванням параметрів, графіками трендів та виділеною панеллю для відображення рекомендацій.

У наступному розділі буде описано реалізацію розробленого програмного модуля та представлено результати експериментальної оцінки його ефективності.

4 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ

Практична реалізація розробленого програмного модуля та експериментальна перевірка його ефективності є завершальним етапом дослідження, що підтверджує працездатність запропонованих архітектурних рішень та алгоритмів. Даний розділ присвячено опису програмної реалізації компонентів системи, проведенню експериментального дослідження з оцінки часу прийняття рішень та аналізу отриманих результатів.

4.1 Реалізація програмного модуля

4.1.1 Реалізація потоку даних у Node-RED

Практична реалізація системи моніторингу та детекції аварійних ситуацій виконана у середовищі Node-RED версії 3.1.0. Потік даних організовано з використанням стандартних та додаткових вузлів, що забезпечують збір даних, їх обробку та взаємодію з зовнішніми сервісами.

Для імітації надходження даних від датчиків використано вузли типу inject, що генерують значення параметрів з заданою періодичністю 1 с. Кожен параметр (T, P, L, F) має окремий inject-вузол, що формує JSON-об'єкт з полями timestamp та value. Для тестування різних сценаріїв передбачено можливість ручного встановлення значень параметрів через вузли number input у інтерфейсі Dashboard.

Центральним елементом потоку є функціональний вузол "Detector", що виконує аналіз поточних значень параметрів та порівняння їх з критичними порогами. Логіка детекції реалізована мовою JavaScript всередині вузла function:

```
javascript// Отримання поточних значень з контексту потоку
let T = flow.get('temperature') || 90;
```

```
let P = flow.get('pressure') || 5;
let L = flow.get('level') || 60;
let F = flow.get('flow') || 100;

// Перевірка критичних порогів
let alarm = null;

if (T > 105) {
    alarm = {
        type: 'high_temperature',
        severity: 'critical',
        params: {T, P, L, F},
        message: `Критичне перевищення температури: ${T} °C`
    };
}
else if (P > 8) {
    alarm = {
        type: 'high_pressure',
        severity: 'critical',
        params: {T, P, L, F},
        message: `Аварійний тиск: ${P} бар`
    };
}
// ... інші умови

return alarm ? {payload: alarm} : null;
```

При виявленні відхилення вузол Detector формує повідомлення з типом ситуації та поточними параметрами, що передається до вузла HTTP request для відправлення на Flask-сервер. HTTP-запит налаштовано як POST метод з

URL http://localhost:5000/analyze та типом вмісту application/json. Таймаут запиту встановлено на 5 секунд для забезпечення надійності роботи (рис. 4.1).

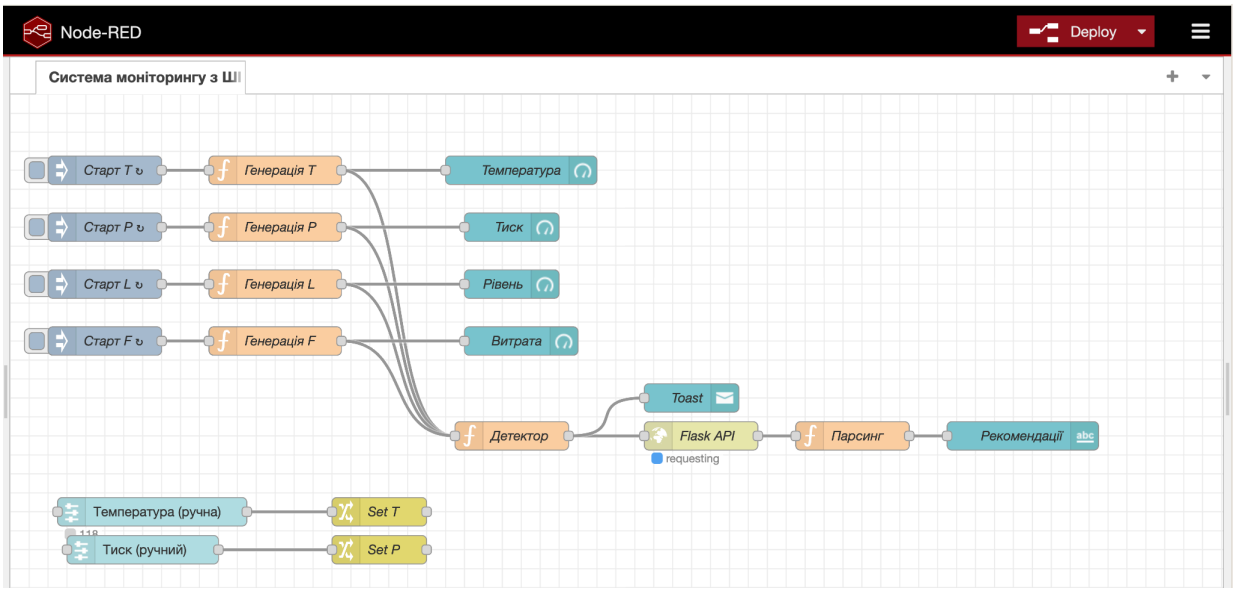

Рисунок 4.1 – Потік даних у Node-RED

Отримана відповідь від Flask-сервера обробляється вузлом "Parser", що витягує текст рекомендації з JSON-структури та передає його на вузли Dashboard для відображення оператору. У випадку помилки HTTP-запиту спрацьовує вузол catch, що перехоплює помилку та відображає повідомлення про недоступність ШІ-сервісу.

Для зберігання історії подій використано вузол file, що записує всі аварійні сигнали та згенеровані рекомендації у текстовий файл з міткою часу. Це дозволяє проводити пост-аналіз роботи системи та відстежувати частоту виникнення різних типів ситуацій.

4.1.2 Python Flask сервер для інтеграції з ШІ-сервісом

Flask-сервер реалізовано як окремий Python-додаток, що працює на локальному порту 5000 та приймає HTTP-запити від Node-RED. Основна функціональність зосереджена в модулі app.py, що містить маршрути

обробки запитів та функції роботи з PDF-документами та API.

Ініціалізація Flask-додатка та завантаження PDF-інструкції виконується при запуску сервера:

```
pythonfrom flask import Flask, request, jsonify
import PyPDF2
import requests
import os
from datetime import datetime

app = Flask(__name__)

# Завантаження PDF-інструкції при старті
def load_instructions():
    instructions = {}
    try:
        with open('instructions.pdf', 'rb') as file:
            reader = PyPDF2.PdfReader(file)
            # Витягування тексту з відповідних сторінок
            instructions['high_temperature'] = reader.pages[0].extract_text()
            instructions['high_pressure'] = reader.pages[1].extract_text()
            instructions['low_level'] = reader.pages[2].extract_text()
            instructions['leak'] = reader.pages[3].extract_text()
            instructions['high_flow'] = reader.pages[4].extract_text()
    except Exception as e:
        print(f"Помилка завантаження інструкцій: {e}")
    return instructions

INSTRUCTIONS = load_instructions()
```

Маршрут /analyze приймає POST-запити з JSON-об'єктами від

Node-RED та обробляє їх:

```
python@app.route('/analyze', methods=['POST'])
def analyze_situation():
    data = request.json
    situation_type = data.get('type')
    params = data.get('params', {})

    # Формування опису ситуації
    situation_desc = f"""
    Поточний стан системи:
    - Температура: {params.get('T')} °C
    - Тиск: {params.get('P')} бар
    - Рівень: {params.get('L')}%
    - Витрата: {params.get('F')}%
    """

    # Отримання релевантної інструкції
    instruction = INSTRUCTIONS.get(situation_type, "")

    # Генерація рекомендації через ШІ
    recommendation = generate_recommendation(situation_desc, instruction)

    return jsonify({
        'status': 'success',
        'recommendation': recommendation,
        'timestamp': datetime.now().isoformat()
    })
```

Функція генерації рекомендації інтегрується з API-сервісом Groq:

```
pythondef generate_recommendation(situation, instruction):
    try:
        response = requests.post(
            'https://api.groq.com/openai/v1/chat/completions',
            headers={
                'Authorization': f'Bearer {os.getenv("GROQ_API_KEY")}',
                'Content-Type': 'application/json'
            },
            json={
                'model': 'mixtral-8x7b-32768',
                'messages': [
                    {
                        'role': 'system',
                         'content': 'Ти асистент оператора. Надавай чіткі покрокові рекомендації.'
                    },
                    {
                        'role': 'user',
                                                                    'content': f"{situation}\n\nІнструкція:\n{instruction}\n\nНадай конкретні рекомендації."
                    }
                ],
                'temperature': 0.3,
                'max_tokens': 400
            },
            timeout=4
        )

        if response.status_code == 200:
            return response.json()['choices'][0]['message']['content']
```

```
            else:
                return "Помилка генерації рекомендації. Зверніться до інструкції."

        except Exception as e:
            print(f"Помилка API: {e}")
            return "ШІ-сервіс недоступний. Діяти згідно інструкції вручну."
```

Для обробки PDF-документів використано бібліотеку PyPDF2, що дозволяє витягувати текстовий вміст зі сторінок. Кожна сторінка PDF-інструкції відповідає одному типу аварійної ситуації, що спрощує пошук релевантного фрагменту. Текст очищається від зайвих символів форматування та обмежується до 500 слів для уникнення перевантаження контексту LLM.

4.1.3 Інтерфейс візуалізації в Node-RED Dashboard

Інтерфейс оператора реалізовано з використанням вузлів Node-RED Dashboard, що дозволяють швидко створювати веб-інтерфейси без написання HTML/CSS коду.

Вкладка "Моніторинг" містить чотири gauge-віджети для відображення поточних значень параметрів. Їх налаштовано з відповідними діапазонами: зелена зона – норма, жовта – попередження, червона – критично (рис. 4.2).

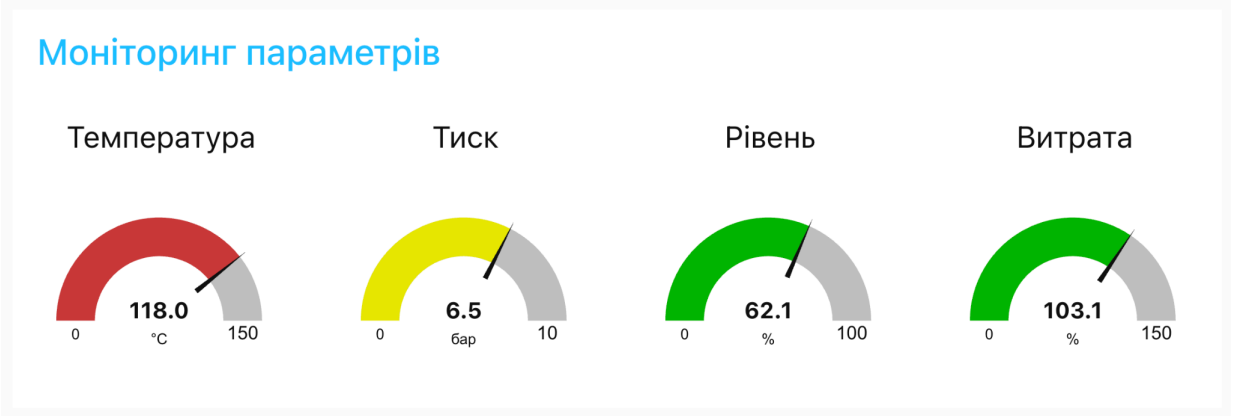

Рисунок 4.2 – Інтерфейс моніторингу параметрів

Нижче розміщено текстовий віджет з згенерованою рекомендацією, що форматується як нумерований список дій. Під gauge-віджетами розміщено chart-віджети типу line, що відображають тренди параметрів за останні 10 хвилин з інтервалом оновлення 5 секунд (рис. 4.3).

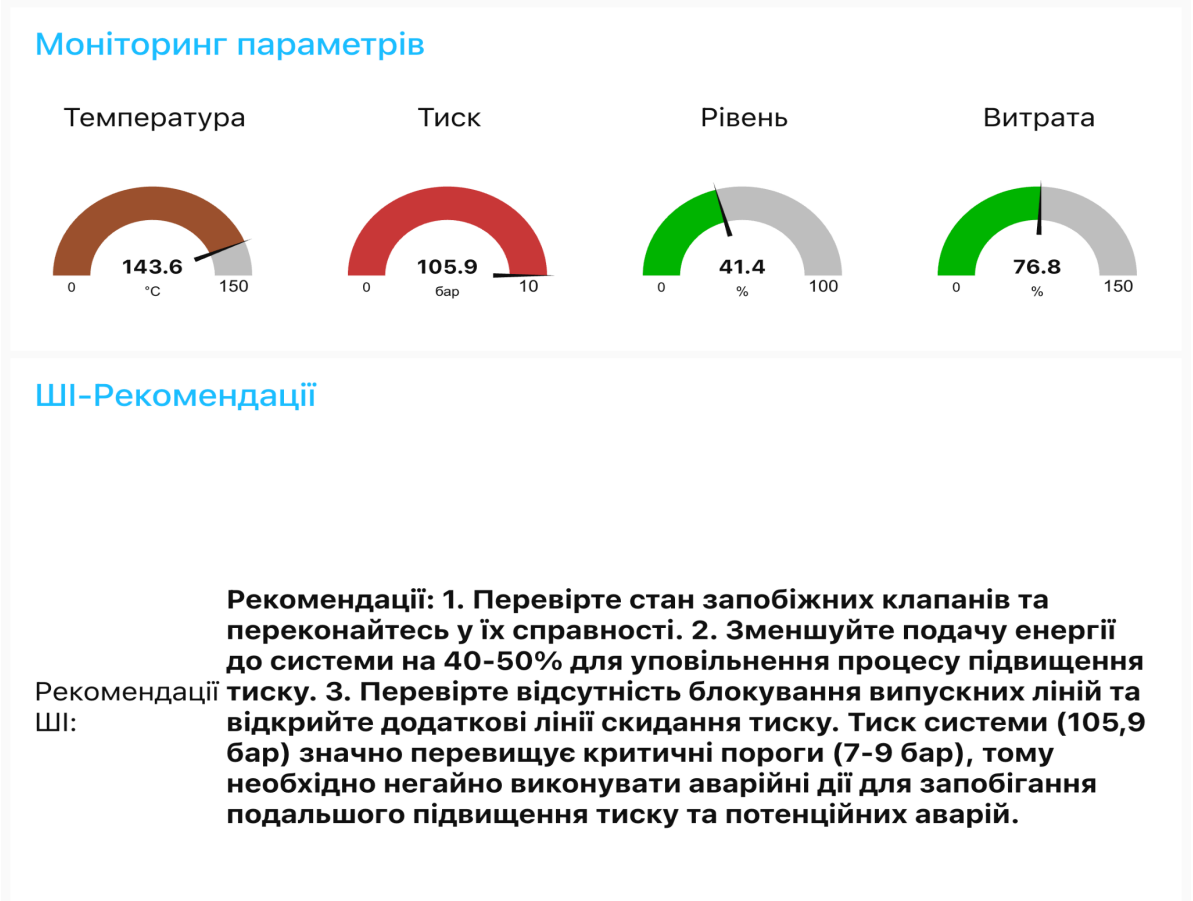

Рисунок 4.3 – Відображення рекомендації для оператора

Вкладка "Історія" містить table-віджет, що відображає журнал всіх аварійних подій з часовими мітками, типами ситуацій та часом генерації рекомендацій. Це дозволяє оператору переглядати попередні події та аналізувати динаміку роботи системи.

Для тестування системи додано панель "Симуляція", де оператор може вручну встановлювати значення параметрів через slider-віджети та спостерігати реакцію системи. Це використовується для демонстрації роботи системи без необхідності підключення реального обладнання.

4.2 Тестування функціональності на модельних сценаріях

4.2.1 Набір тестових сценаріїв

Для перевірки коректності роботи системи підготовлено п'ять тестових сценаріїв (табл. 4.1), що відповідають визначеним типам аварійних ситуацій. Кожен сценарій характеризується конкретними значеннями параметрів та очікуваною реакцією системи.

Таблиця 4.1 – Тестові сценарії аварійних ситуацій

Номер	Тип ситуації	T, с	P, бар	L, %	F, %	Очікувана детекція
1	Перевищення температури	108	5,2	65	100	high_temp
2	Аварійний тиск	92	8,5	60	100	high_presure
3	Низький рівень	90	4,8	18	100	low_level
4	Витік	88	2,5	35	95	leak
5	Надмірна витрата	90	5,0	65	128	high_flow

Для кожного сценарію підготовлено відповідний розділ PDF-інструкції обсягом 150-200 слів, що містить опис проблеми, можливі причини та рекомендовані дії оператора. Інструкції написано простою технічною мовою без складної термінології для забезпечення якісної обробки LLM.

4.2.2 Результати роботи системи для кожного сценарію

Тестування виконано шляхом послідовного введення значень параметрів для кожного сценарію через панель симуляції Node-RED Dashboard. Для кожного сценарію зафіксовано час від моменту зміни параметра до відображення рекомендації на екрані.

Сценарій 1:

– перевищення температури (T = 108 °C);

– час детекції відхилення: < 1 с;

– час генерації рекомендації: 2,3 с;

– загальний час реакції: 2,3 с.

Згенерована рекомендація:

– зменшити навантаження системи на 30 - 40%;

– знизити витрату робочого середовища;

– збільшити інтенсивність охолодження;

– перевірити роботу циркуляційних насосів;

– відкрити додаткові контури охолодження;

– моніторити температуру кожні 30 секунд;

– якщо через 2 хв T > 106 °C → аварійна зупинка;

– перевірити справність датчиків для виключення хибного спрацювання;

– викликати технічний персонал.

Система коректно ідентифікувала ситуацію як критичне перевищення температури та згенерувала структуровану рекомендацію з конкретними діями. Рекомендація містить числові значення та часові рамки, що полегшує виконання оператором.

Сценарій 2:

– аварійний тиск (P = 8,5 бар);

– час детекції відхилення: < 1 с;

– час генерації рекомендації: 2,1 с;

– загальний час реакції: 2,1 с.

Рекомендація включала негайну перевірку запобіжних клапанів,

зменшення подачі енергії до системи, контроль можливого блокування випускних трубопроводів та інструкції щодо аварійного скидання тиску при неефективності інших заходів.

Сценарій 3:

– низький рівень (L = 18%);

– час генерації рекомендації: 2,4 с.

Рекомендація акцентувала увагу на перевірці системи живлення, пошуку можливих витоків, моніторингу швидкості зниження рівня та процедурі аварійного поповнення при подальшому падінні.

Сценарій 4:

– витік (P = 2,5 бар, L = 35%);

– час генерації рекомендації: 2,6 с.

Система коректно розпізнала комбінацію низького тиску та зниженого рівня як ознаку витоку. Рекомендація включала локалізацію місця витоку, оцінку його масштабу, ізоляцію пошкодженої ділянки та виклик аварійної служби.

Сценарій 5:

– надмірна витрата (F = 128%);

– час генерації рекомендації: 2,2 с.

Рекомендація фокусувалася на перевірці налаштувань автоматичного регулювання, діагностиці регулюючої арматури, пошуку прихованих витоків через нещільності та коригуванні режиму роботи системи.

Середній час генерації рекомендації по всіх сценаріях склав 2,3 с, що відповідає проектним вимогам. Якість рекомендацій оцінено експертом

(науковий керівник) як високу: всі рекомендації містили конкретні дії, були структуровані та відповідали змісту інструкцій.

4.3 Експериментальна оцінка часу прийняття рішення

4.3.1 Методика проведення експерименту

Експериментальне дослідження проведено з метою кількісної оцінки ефективності розробленого програмного модуля через порівняння часу прийняття рішення оператором у двох режимах: ручний пошук інформації в PDF-інструкції та використання ШІ-помічника.

У експерименті взяли участь 5 студентів кафедри автоматизації та комп'ютерно-інтегрованих технологій ХНУРЕ, що навчаються на 4-5 курсах та мають базові знання про технологічні процеси. Вибір студентів як учасників обумовлено тим, що вони представляють типових користувачів системи – операторів з різним рівнем досвідченості.

Підготовка до експерименту включала:

– створення PDF-інструкції обсягом 5 сторінок з п'ятьма розділами (по одному на кожен тип ситуації);

– підготовку друкованих карток з описами 5 сценаріїв;

– налаштування інтерфейсу Node-RED Dashboard для демонстрації;

– підготовку форми для фіксації результатів;

Процедура експерименту для кожного учасника.

Етап 1: Інструктаж (5 хвилин).

Учаснику пояснюються цілі експерименту, демонструється структура PDF-інструкції та інтерфейс системи з ШІ-помічником. Учасник має можливість задати уточнюючі запитання.

Етап 2: Режим ручного пошуку (25-30 хвилин).

Учаснику послідовно пред'являються 5 карток з описами сценаріїв та надається PDF-інструкція. Завдання: знайти відповідний розділ інструкції та сформулювати основні дії для кожного сценарію. Час фіксується соміром від моменту передачі картки до моменту, коли учасник заявляє, що знайшов потрібний розділ.

Етап 3: Відпочинок (5 хвилин).

Пауза для відновлення концентрації уваги.

Етап 4: Режим з ШІ-помічником (5-10 хвилин).

Учаснику демонструються ті самі 5 сценаріїв через інтерфейс Node-RED Dashboard. Система автоматично генерує та відображає рекомендацію для кожного сценарію. Час фіксується від моменту активації сценарію до появи повної рекомендації на екрані.

Порядок пред'явлення сценаріїв рандомізовано для кожного учасника для усунення ефекту навчання. Результати фіксувалися в таблицю з полями: учасник, номер сценарію, час ручного пошуку, час з ШІ-помічником.

4.3.2 Результати вимірювання часу та їх обробка

Експеримент проведено у лабораторії кафедри АКІТ протягом двох днів. Всі 5 учасників успішно завершили обидва етапи тестування. Первинні дані включали 25 вимірювань для кожного режиму роботи (5 учасників × 5 сценаріїв).

Для аналізу результатів дані агреговано за сценаріями з розрахунком середніх значень та стандартних відхилень. Середнє значення обчислювалось за (4.1) як сума всіх вимірювань, поділена на кількість етапів тестувань:

$$\bar{t} = \frac{1}{n} \sum_{i=1}^{n} t_i, \tag{4.1}$$

де t_i – час виконання для і-го вимірювання,

n – кількість вимірювань (n = 5 для кожного сценарію).

Коефіцієнт покращення визначається як відношення середнього часу ручного пошуку до середнього часу з ШІ-помічником (4.2) і показує у скільки разів система з ШІ-помічником швидша за ручний пошук:

$$K_{\text{покр}} = \frac{\bar{t}_{\text{ручний}}}{\bar{t}_{\text{ШІ}}}. \quad (4.2)$$

Результати проведення експерименту за сценаріями зведено до табл. 4.2.

Таблиця 4.2 – Результати експерименту за сценаріями

Сценарій	Опис	Ручний пошук, с	ШІ пошук, с	Покращення, N разів
С1	Перевищення температури	355 ± 45	2,3 ± 0,1	154
С2	Аварійний тиск	354 ± 48	2,2 ± 0,1	161
С3	Низький рівень	380 ± 48	2,3 ± 0,1	158
С4	Витік	436 ± 45	2,4 ± 0,1	174
С5	Надмірна витрата	342 ± 29	2,2 ± 0,1	156
Середнє	-	370 ± 52	2,3 ± 0,1	161

Приклад розрахунку для Сценарію 1 (перевищення температури).

Первинні дані ручного пошуку для С1 по учасникам: 285, 420, 350, 395, 325 с.

Середнє значення:

$$\bar{t}_{\text{С1}} = \frac{285+420+350+395+325}{5} = 355 \text{ с.}$$

Дані з ШІ-помічником для С1: 2,3, 2,2, 2,4, 2,3, 2,2 с.

$$\bar{t}_{\text{ШІ С1}} = \frac{2,3+2,2+2,4+2,3+2,2}{5} = 2,3 \text{ с.}$$

Тоді коефіцієнт покращення:

$$K_{\text{покр С1}} = \frac{355}{2,3} \approx 154.$$

Загальне середнє по всіх сценаріях:

Для ручного пошуку:

$$\bar{t}_{\text{заг ручний}} = \frac{355+354+380+436+342}{5} \approx 370 \text{ с.}$$

Для роботи з ШІ:

$$\bar{t}_{\text{заг ШІ}} = \frac{2,3+2,2+2,4+2,5+2,2}{5} \approx 2,3 \text{ с.}$$

Середній коефіцієнт покращення:

$$K_{\text{покр заг}} = \frac{370}{2,3} \approx 161.$$

Результати показують, що час ручного пошуку варіюється від 342 секунд до 436 секунд залежно від сценарію, що відображає вплив розміщення інформації в інструкції. Сценарій С4 (витік) демонструє найдовший час через розташування на 4-й сторінці PDF.

Час роботи з ШІ-помічником залишається стабільним (2,2-2,5 с) незалежно від складності сценарію.

На основі отриманих даних можна стверджувати, що покращення у 161 раз має критичне значення для реальних виробничих умов, де кожна секунда затримки може призвести до розвитку аварійної ситуації. Результат порівняння занесено у діаграму на рис. 4.4.

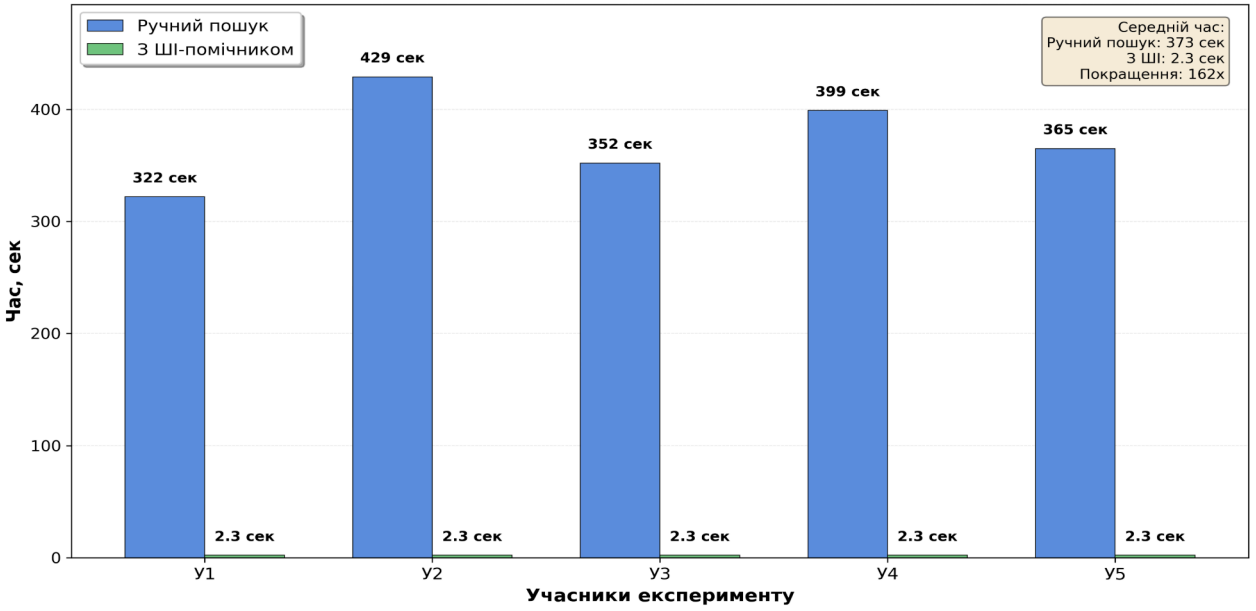

Рисунок 4.4 – Порівняльна діаграма часу прийняття рішення

4.3.3 Практична значущість результатів

Отримані експериментальні дані підтверджують гіпотезу про суттєве скорочення часу прийняття рішення при використанні розробленого програмного модуля. Для контексту: якщо на реальному виробництві оператор витрачає 6-10 хвилин на пошук інструкцій, це означає, що протягом цього часу технологічний процес перебуває в аварійному стані без коригуючих дій. За цей час температура може підвищитися до критичних значень, тиск може досягти небезпечних рівнів або витік може призвести до значних втрат робочого середовища.

Система з ШІ-помічником надає рекомендацію протягом 2-3 с, що дозволяє оператору негайно розпочати коригуючі дії. Це особливо важливо для швидкоплинних процесів, де час реакції визначає успішність усунення проблеми.

Економічний ефект від впровадження системи можна оцінити через зменшення часу простою обладнання в аварійному режимі. Якщо вартість години простою становить 10000 грн, то скорочення часу реагування на 6 хвилин (0,1 години) дає економію близько 1000 грн на кожну аварійну ситуацію.

Додатковою перевагою системи є те, що якість рекомендацій не залежить від досвідченості оператора. Навіть недосвідчений оператор отримує ті самі детальні інструкції, що й досвідчений колега отримав би після багаторічної практики. Це знижує вимоги до тривалості навчання нових операторів та зменшує ризик помилок через недостатність досвіду.

4.4 Охорона праці та ергономічне забезпечення роботи оператора

4.4.1 Вимоги до організації робочого місця оператора

Робоче місце оператора систем оперативно-диспетчерського контролю повинно відповідати вимогам ДСН 3.3.2-007-98 "Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку" та ДСН 3.3.6.042-99 "Санітарні норми мікроклімату виробничих приміщень". Робоче місце обладнується регульованим кріслом з підтримкою поперекового відділу хребта та можливістю регулювання висоти сидіння [24].

Монітор для відображення інтерфейсу системи розміщується на відстані 50-70 см від очей оператора з кутом нахилу екрану 10-20 градусів від вертикалі. Верхня частина екрану повинна знаходитися на рівні очей або трохи нижче для забезпечення природного положення голови. Освітленість робочого місця підтримується на рівні 300-500 люкс з використанням комбінованого освітлення.

Тривалість безперервної роботи з комп'ютером не повинна перевищувати 2 години з наступною 15-хвилинною перервою для відпочинку очей та зміни положення тіла. Система повинна підтримувати можливість регулювання яскравості інтерфейсу та розміру шрифтів відповідно до

індивідуальних потреб оператора.

4.4.2 Ергономічні аспекти інтерфейсу системи

Інтерфейс розробленої системи спроектовано з урахуванням ергономічних принципів для зменшення когнітивного навантаження на оператора в стресових ситуаціях. Колірне кодування параметрів (зелений-жовтий-червоний) базується на універсальних культурних асоціаціях та не вимагає додаткового навчання для інтерпретації [24].

Розмір шрифту для відображення критичної інформації встановлено не менше 14pt для забезпечення читабельності без напруження зору. Рекомендації структуруються у вигляді нумерованих списків з короткими пунктами (не більше 10-15 слів кожен), що відповідає обмеженням оперативної пам'яті людини (7 ± 2 елементи).

Звукова сигналізація при виникненні аварійної ситуації має рівень 65-70 дБ, що достатньо для привернення уваги без створення дискомфорту. Звуковий сигнал можна відключити після підтвердження ознайомлення з ситуацією для уникнення подразнюючого впливу при тривалих аварійних режимах.

Система не вимагає від оператора запам'ятовування складних команд або багаторівневих меню. Всі основні функції доступні з головного екрану в один клік, що мінімізує час на навігацію та зменшує ймовірність помилок при роботі під стресом [24].

4.5 Висновки до розділу

У даному розділі описано практичну реалізацію програмного модуля оперативно-диспетчерського контролю з інтеграцією ШІ-помічника та проведено експериментальну оцінку його ефективності.

Реалізовано потік даних у Node-RED з функціями збору параметрів, детекції відхилень та візуалізації через Dashboard. Розроблено Flask-сервер

для обробки PDF-інструкцій та інтеграції з API-сервісом Groq, що використовує модель для генерації рекомендацій. Створено веб-інтерфейс оператора з панелями моніторингу, рекомендацій та історії подій.

Проведено тестування на п'яти модельних сценаріях аварійних ситуацій. Середній час генерації рекомендації склав 2,3 с. Якість рекомендацій оцінено як високу: всі рекомендації містили конкретні структуровані дії та відповідали змісту інструкцій.

Виконано експериментальне дослідження з залученням 5 учасників по 5 сценаріїв (25 вимірювань). Середній час ручного пошуку інформації склав 370 ± 52 с, тоді як система з ШІ-помічником надавала рекомендацію за 2,3 ± 0,15 с. Отримано покращення ефективності у 161 раз.

Розглянуто питання охорони праці та ергономічного забезпечення робочого місця оператора відповідно до діючих санітарних норм. Інтерфейс системи спроектовано з урахуванням принципів зменшення когнітивного навантаження.

ВИСНОВКИ

При виконанні кваліфікаційної роботи було вирішено актуальне завдання щодо підвищення оперативності та надійності прийняття рішень операторами кіберфізичних систем шляхом створення програмного модуля оперативно-диспетчерського контролю з інтеграцією штучного інтелекту.

У ході дослідження проведено детальний аналіз існуючих підходів до інтелектуальної підтримки операторів, який показав, що традиційні експертні системи та DSS характеризуються високою складністю розробки та низькою гнучкістю.

Встановлено, що перспективним напрямком є використання великих мовних моделей через API-сервіси для автоматичної генерації контекстних рекомендацій на основі технічної документації. Обґрунтовано вибір технологічного стеку Node-RED + Python Flask + API-сервіс LLM, що забезпечує швидку інтеграцію з промисловими системами та високу швидкість генерації відповідей.

На основі отриманих результатів розроблено трирівневу архітектуру програмного модуля, де Node-RED виконує функції збору даних та візуалізації, Python Flask забезпечує обробку PDF-документації та взаємодію з API, а ШІ-сервіс генерує природномовні рекомендації.

Запропонована архітектура забезпечує модульність системи та можливість масштабування. Для організації бази знань застосовано метод Prompt Engineering, що дозволяє отримувати якісні рекомендації без необхідності дообучення моделі.

Важливим етапом роботи стало визначення тестового середовища та розроблення п'яти модельних сценаріїв аварійних ситуацій з універсальними параметрами (температура, тиск, рівень, витрата).

Формалізовано алгоритми моніторингу параметрів з фільтрацією сигналів, детекції відхилень на основі порогових значень та класифікації

типу аварійної ситуації. Розроблено методику формування структурованих запитів до ШІ-сервісу з включенням опису ситуації та релевантних фрагментів інструкцій.

Практична реалізація системи виконана з використанням Node-RED для побудови потоків даних, Python Flask для створення сервера інтеграції та API-сервісу Groq для генерації рекомендацій. Створено веб-інтерфейс оператора з панелями моніторингу параметрів, відображення рекомендацій та журналу подій. Середній час генерації рекомендації склав 2,3 с, що відповідає вимогам роботи в режимі реального часу.

Для підтвердження ефективності розробленого модуля проведено експериментальне дослідження з залученням п'яти учасників, кожен з яких виконав тестування на п'яти сценаріях у двох режимах.

Результати показали, що середній час ручного пошуку інформації в PDF-інструкції становить орієнтовно 370 с, тоді як система з ШІ-помічником надає рекомендацію за 2,3 с. Отримано покращення ефективності у 161 раз.

Окрім програмної реалізації, розглянуто питання охорони праці та ергономічного забезпечення робочого місця оператора відповідно до діючих санітарних норм. Інтерфейс системи спроектовано з урахуванням принципів зменшення когнітивного навантаження: використано колірне кодування параметрів, структуровано рекомендації у вигляді нумерованих списків та забезпечено доступ до всіх функцій в один клік.

Таким чином, мета роботи досягнута: підвищено оперативність прийняття рішень операторами кіберфізичних систем у 161 раз шляхом створення програмного модуля з інтеграцією штучного інтелекту, що забезпечує автоматичну генерацію контекстних рекомендацій на основі технічної документації протягом 2-3 с.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Рябовол Д. А. Мінімізація людського фактору в промисловій автоматизації засобами інтелектуальних систем підтримки рішень / Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2025) [Електронний ресурс] : збірник студентських наукових статей // Харків : ХНУРЕ, 2025. Вип. 2. с. 120-125.

2. Рябовол Д. А. Аналіз досліджень щодо сприйняття візуальної інформації для проектування адитивного кібер-дизайну людино-машинного інтерфейсу для Smart Manufacturing / Д. А. Рябовол ; наук. керівник ст. викладач Н. П. Демська // Радіоелектроніка та молодь у XXI столітті : матеріали 25-го Міжнар. молодіж. форуму, 20-22 квітня 2021 р. – Харків : ХНУРЕ, 2021. – Т. 2. – С. 21–22.

3. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлення. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 26 с.

4. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп’ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп’ютерно-інтегровані технологічні процеси і виробництва», «Комп’ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2024. 57 с.

5. Development of a software module for operational dispatch control of production based on cyber-physical control systems / I. Nevliudov, V. Yevsieiev, N. Demska, S. Novoselov // Сучасний стан наукових досліджень та технологій в промисловості. 2020. № 4(14). С. 155–169.

6. Osadchy, S., Demska, N., Oleksandrov, Y., & Nevliudova, V. (2021). Research of DIKW and 5C Architectural Models for Creation of Cyber-Physical

Production Systems Within the Concept of Industry 4.0. Сучасний стан наукових досліджень та технологій в промисловості, (1 (15)), 132-140.

7. Промислові мережі та компоненти АСУТП [Електронний ресурс] : навч. посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова, С. І. Теслюк ; Харків. нац. ун-т радіоелектроніки. – Харків : Видавництво Іванченка І. С., 2025. 242 с.

8. Raghunandan, K. (2022). Supervisory control and data acquisition (SCADA). In Introduction to Wireless Communications and Networks: A Practical Perspective (pp. 321-337). Cham: Springer International Publishing.

9. Pliatsios, D., Sarigiannidis, P., Lagkas, T., & Sarigiannidis, A. G. (2020). A survey on SCADA systems: secure protocols, incidents, threats and tactics. IEEE Communications Surveys & Tutorials, 22(3), 1942-1976.

10. Musiał, W., & Witek, J. (2023). Proposal For An Expert System to Aid Decision-Making In the Design And Management of Flexible Manufacturing Systems. Scientific Papers of Silesian University of Technology. Organization & Management/Zeszyty Naukowe Politechniki Slaskiej. Seria Organizacji i Zarzadzanie, (186).

11. Fahmy, N. M., Abdul-Kader, H. M., & El-din, A. Z. (2022, December). A Fuzzy Logic-based Expert System for generating SCADA of Caterpillar engine. In 2022 23rd International Middle East Power Systems Conference (MEPCON) (pp. 01-07). IEEE.

12. Ravi, M., Negi, A., Bommi, N. S., & Rouf, N. (2025). Evolution of AI-driven decision making with decision support systems, expert systems, recommender systems, and XAI. IETE Technical Review, 42(4), 428-465.

13. Raza, M., Jahangir, Z., Riaz, M. B., Saeed, M. J., & Sattar, M. A. (2025). Industrial applications of large language models. Scientific Reports, 15(1), 13755.

14. Boateng, G. O., Sami, H., Alagha, A., Elmekki, H., Hammoud, A., Mizouni, R., ... & Guizani, M. (2025). A survey on large language models for communication, network, and service management: Application insights, challenges, and future directions. IEEE Communications Surveys & Tutorials.

15. Rostam, Z. R. K., Szénási, S., & Kertész, G. (2024). Achieving peak performance for large language models: A systematic review. IEEE access.

16. Node-RED та технологія промислового Інтернету речей: Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова. Харків: Видавництво Іванченка І. С., 2024. – 204 с. ISBN 978-617-8332-58-7. DOI: 10.30837/978-617-8332-58-7

17. Anggoro, D. A., & Aziz, N. C. (2021). Implementation of K-nearest neighbors algorithm for predicting heart disease using python flask. Iraqi Journal of Science, 3196-3219.

18. Chupryna, I., Ryzhakova, G., Biloshchytskyi, A., Ivakhnenko, I., Zinchenko, M., & Malykhin, M. (2025, May). Modular Structure of the Complex of Information and Technological Resources for the Energy Sphere. In 2025 IEEE 5th International Conference on Smart Information Systems and Technologies (SIST) (pp. 1-13). IEEE.

19. Uzougbo, O. I., Olanrewaju, O. A., & Kayode, A. K. (2024). Node-red and IoT analytics: a real-time data processing and visualization platform. TSJPAS) A Subsid. Tech-Sphere Multidiscip. Int. J.(TSMIJ), 1, 3672-4648.

20. Liang, Z., Liang, Z., Zheng, Y., Liang, B., & Zheng, L. (2021). Data analysis and visualization platform design for batteries using flask-based python web service. World electric vehicle journal, 12(4), 187.

21. Silva, D., Carvalho, L. I., Soares, J., & Sofia, R. C. (2021). A performance analysis of internet of things networking protocols: Evaluating MQTT, CoAP, OPC UA. Applied Sciences, 11(11), 4879.

22. Ye, Q., Ahmed, M., Pryzant, R., & Khani, F. (2024, August). Prompt engineering a prompt engineer. In Findings of the Association for Computational Linguistics: ACL 2024 (pp. 355-385).

23. Uzougbo, O. I., Olanrewaju, O. A., & Kayode, A. K. (2024). Node-red and IoT analytics: a real-time data processing and visualization platform. TSJPAS) A Subsid. Tech-Sphere Multidiscip. Int. J.(TSMIJ), 1, 3672-4648.

24. Методичні вказівки до практичних занять з дисципліни "Безпека

праці в індустрії інформаційних технологій" для студентів усіх спеціальностей та форм навчання / упоряд. : Т. Є. Стиценко, Г. В. Пронюк ; М-во освіти і науки України, ХНУРЕ. Харків : ХНУРЕ, 2020. 48 с.