Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти і газу
Інститут інформаційних технологій
Кафедра інформаційно-телекомунікаційних технологій і систем

Бачевський Андрій

УДК 004.942:621.5.011:658.58

МАГІСТЕРСЬКА РОБОТА

«Розроблення цифрового двійника автоматизованої системи управління технологічним процесом»

Комп'ютеризовані системи управління та автоматики
(назва освітньої програми)

Спеціальність **151 Автоматизація та компютерно-інтегровані технології**
(код і назва спеціальності)

_____Андрій БАЧЕВСЬКИЙ
(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Заміховська Олена Леонідівна, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

Завідувач кафедри ІТТС

_____Леонід ЗАМІХОВСЬКИЙ
(підпис) (дата) (ініціали та прізвище)

Рецензент_____
(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2025 рік

АНОТАЦІЯ

Робота присвячена розробці цифрового двійника компресорної станції для підвищення стійкості критичної інфраструктури до раптових втрат електроживлення, спричинених атаками на енергосистему в умовах гібридної війни.

Запропоновано легковагову архітектуру на основі відкритого технологічного стеку (Python, Node-RED, MQTT), яка реалізує концепцію «цифрової стійкості». Система забезпечує автономний моніторинг ключових параметрів, детекцію аварій з затримкою менше 3 секунд, локальне зберігання даних та негайне сповіщення персоналу через Telegram-бот. Веб-дашборд з підтримкою PWA надає оперативний доступ до стану об'єкта.

Експериментальна перевірка прототипу підтвердила його ефективність: час доставки аварійного сповіщення становить до 1,1 секунди, а коефіцієнт готовності системи перевищує 99%. Розроблене рішення дозволяє запобігати каскадним технологічним аваріям та мінімізувати наслідки блекаутів для критичних об'єктів.

ABSTRACT

This work is dedicated to the development of a digital twin for a compressor station, aimed at enhancing the resilience of critical infrastructure against sudden power losses caused by attacks on the energy system in the context of hybrid warfare.

A lightweight architecture based on an open technology stack (Python, Node-RED, MQTT) is proposed, implementing the concept of "digital resilience". The system provides autonomous monitoring of key parameters, failure detection with a latency of under 3 seconds, local data storage, and immediate personnel notifications via a Telegram bot. A PWA-supported web dashboard offers real-time operational access to the object's status.

Experimental testing of the prototype confirmed its effectiveness: the emergency notification delivery time is up to 1.1 seconds, and the system availability rate exceeds 99%. The developed solution helps prevent cascading technological failures and minimizes the impact of blackouts on critical facilities.

РЕФЕРАТ

Текстова частина магістерської роботи 87 стор., 37 рис., 1 табл., 6 додатків, 25 літературних джерел.

Об'єкт дослідження: автоматизована система управління компресорною станцією КС-3 газотранспортної системи, що забезпечує транспортування природного газу.

Предмет дослідження: технології моделювання, збору даних, візуалізації та прогнозування параметрів цифрового двійника зазначеної компресорної станції, спрямовані на підвищення її ефективності та надійності.

Мета роботи − розробка цифрового двійника компресорної станції з інтеграцією в автоматизовану систему керування (АСК) для забезпечення моніторингу в реальному часі, виявлення аварій та прогнозування параметрів у умовах нестабільного електроживлення, зумовленого зовнішніми факторами.

У першому розділі МР було розглянуто теоретичні основи концепції цифрового двійника у контексті автоматизованих систем керування технологічними процесами. Описано багаторівневу архітектуру ЦД, що охоплює фізичний, керуючий та цифровий рівні, а також ключові компоненти його структури. Розглянуто методологічні аспекти розробки ЦД.

В другому розділі МР було розглянуто методологічні засади та інструментарій розробки ЦД АСК компресорної станції газотранспортної системи. Проведено аналіз проблемної області в контексті енергодефіциту та воєнного стану. На основі аналізу сформовано та формалізовано науково-технічну задачу дослідження, а також сформульовано комплекс функціональних та нефункціональних вимог до системи цифрового двійника.

В третьому розділі надано детальний опис інженерної реалізації та верифікації системи цифрового двійника АСК компресорної станції. Визначено науково-технічні результати. Проведено експериментальну верифікацію.
Обґрунтовано архітектуру цифрового двійника, засновану на концепції «цифрової стійкості» (Digital Resilience).

Розроблено математичну модель та програмний симулятор компресорної станції на базі Python. Модель адекватно відтворює динаміку ключових параметрів (напруга, тиск, температура) у нормальному, аварійному та відновлювальному режимах, зокрема при раптових відключеннях електроживлення, що є типовим сценарієм для критичної інфраструктури України.

Дослідження підтвердило, що основним дестабілізуючим фактором для функціонування компресорної станції є нестабільність зовнішнього електропостачання.

Практично реалізовано працюючий прототип повноцінного цифрового двійника. Система забезпечує повний цикл від генерації даних та їх передачі через MQTT-брокер до обробки бізнес-логіки в Node-RED, візуалізації на операторському дашборді, локального логування та аварійного сповіщення через Telegram.

Ключові слова: ц*ифровий двійник (ЦД), компресорна станція, автоматизована система керування (АСК), MQTT, Node-RED, енергостійкість, аварійне сповіщення, промисловий Інтернет речей (IIoT).*

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

АСК ТП Автоматизовані системи керування технологічними процесами

ЦД Цифрові двійники

КС Компресорна станція

MQTT **(**Message Queuing Telemetry Transport), *спрощений мережевий протокол, що працює на TCP/IP. В*ідкритий протокол обміну повідомленнями за принципом «видавець-підписник» (pub/sub), ідеальний для Інтернету речей (IoT) та систем з обмеженими ресурсами, оскільки він ефективно передає дані між пристроями за допомогою мінімального навантаження на мережу та підтримки ненадійних з'єднань

Node-RED Потужний інструмент візуального програмування з відкритим кодом, створений для поєднання апаратних пристроїв, API та онлайн-сервісів (IoT). Він базується на Node.js та дозволяє розробляти програми шляхом з'єднання вузлів (nodes) у потоки даних, що робить його ідеальним для автоматизації, Інтернету речей та обробки даних

IIoT Промисловий Інтернет речей

PWA (Progressive Web App), гібрид сайту та мобільної програми, що працює через браузер, але виглядає і функціонує як нативний додаток. PWA встановлюються на головний екран без магазинів (App Store/Google Play), працюють офлайн, підтримують push-сповіщення та швидко завантажуються.

Node-RED Dashboard Модуль (бібліотека вузлів) для середовища Node-RED, що дозволяє швидко створювати інтерактивні веб-інтерфейси користувача (приладові панелі) для IoT-проєктів. Він використовує набір віджетів (кнопки, графіки, перемикачі) для візуалізації даних та керування пристроями в режимі реального часу без написання HTML/CSS.

Python Високорівнева, інтерпретована мова програмування загального призначення

Генерація CSV *(англ. Comma Separated Values),* процес автоматичного створення текстових файлів, де табличні дані (числа та текст) розділені комами або крапками з комою, що дозволяє легко переносити інформацію між різними програмами

ЗМІСТ

ВСТУП

Сучасний етап індустріального розвитку ознаменований радикальною модернізацією класичних виробничих структур, спричиненою реалізацією принципів Індустрії 4.0 та цілісним переходом до цифрових моделей господарювання. Ключовими драйверами цієї трансформації виступають інтеграція кіберфізичних систем, розгортання мереж промислового Інтернету речей (IIoT) та активне використання технології цифрових двійників. Згідно з даними аналізу глобального ринку, капіталізація напряму цифрових двійників демонструє експоненційне зростання: від скромних кількох мільярдів доларів США на початку 2020-х років до прогнозованих десятків мільярдів до середини десятиліття. Така динаміка підтверджує високий технологічний рівень та значний економічний вплив даної технологічної парадигми. [1, 2].

У контексті української промисловості, особливо після повномасштабної агресії РФ, постала гостра потреба у модернізації виробничих потужностей, підвищенні їх енергоефективності та адаптації до сучасних вимог ринку.

Автоматизовані системи керування технологічними процесами (АСК ТП) − критично важливий компонент сучасного виробництва, однак традиційні підходи до їх проектування та експлуатації мають суттєві обмеження, зокрема відсутність можливості прогнозування стану системи, обмежені можливості оптимізації в реальному часі та недостатній рівень інтеграції з сучасними аналітичними інструментами. Цифрові двійники (ЦД) виникають як революційна технологія, що дозволяє подолати ці обмеження шляхом створення віртуальних копій фізичних систем з двостороннім обміном даними протягом усього життєвого циклу. Особливу актуальність набуває розробка ЦД саме для українських промислових підприємств, які мають специфічні умови експлуатації, обмежені ресурси для модернізації та потребують адаптованих рішень. Як зазначають вітчизняні дослідники, рівень впровадження ЦД в Україні не перевищує 18%, що значно нижче за світові показники [3, 4].

Актуальність роботи зумовлена потребою у підвищенні стійкості критичної інфраструктури до зовнішніх загроз, зокрема до нестабільного електроживлення внаслідок пошкодження енергетичної інфраструктури.

Тема випускної кваліфікаційної магістерської роботи «Розроблення цифрового двійника автоматизованої системи управління технологічним процесом».

Наукова проблема: У сучасних умовах гібридної війни проти України, що характеризується систематичними атаками на енергетичну інфраструктуру, критичною є вразливість традиційних АСК промислових об'єктів до раптових та повних втрат електроживлення. Компресорні станції, як технологічні серцевини виробництва (наприклад, кисневого заводу), газотранспортних систем та теплоенергетики, після миттєвої зупинки через блэкаут залишаються в неконтрольованому аварійному стані. Відсутність інтелектуальних систем, здатних не лише фіксувати факт аварії, але й автономно виконувати запобіжні алгоритми та ініціювати відновлювальні процеси в умовах обмеженої зв'язності та присутності персоналу, призводить до каскадних технологічних аварій, значних матеріальних збитків та загроз техногенної безпеки.

Сучасний аналітичний огляд наукової літератури фіксує інтенсифікацію досліджень концепції цифрових двійників, що розвивається на перетині кіберфізичних систем, комп'ютерного моделювання та наук про дані. Фундаментальні праці Майкла Грівза заклали теоретичні основи цієї концепції, визначивши її як формальне цифрове представлення фізичного об'єкта або системи [5]. Подальші дослідження Tao F. та Zhang M. розширили це поняття, запропонувавши концепцію "Digital Twin Shop-Floor" для цілісного моделювання виробничих систем [6].

Класифікаційні підходи, запропоновані Kritzinger W. та співавт. [7], розрізняють цифрові моделі, тіні та повноцінні двійники, а дослідження щодо їх інтеграції в кібер-фізичні системи виявили ключові архітектурні принципи [8].

Вітчизняні науковці також внесли значний внесок, досліджуючи специфіку моделювання для української промисловості [9] та методи інтеграції

в існуючі системи керування [10]. Окрему увагу приділено застосуванню методів штучного інтелекту для прогнозування стану обладнання [11, 12].

Незважаючи на інтенсивний розвиток теорії, залишається недостатньо розробленою галузь практичної реалізації архітектур цифрових двійників для автоматизованих систем управління технологічними процесами (АСУ ТП), зокрема адаптованих до специфічних умов українських промислових підприємств, що характеризуються обмеженою ресурсною базою та нестабільністю енергопостачання.Особливої уваги потребують методи побудови гібридних моделей критичної інфраструктури та інтеграції легковагових засобів моніторингу в реальному часі. Саме на вирішення цих практичних завдань спрямоване дане дослідження.

Об'єктом дослідження: автоматизована система управління компресорною станцією КС-3 газотранспортної системи, що забезпечує транспортування природного газу.

Предметом дослідження: технології моделювання, збору даних, візуалізації та прогнозування параметрів цифрового двійника зазначеної компресорної станції, спрямовані на підвищення її ефективності та надійності.

Мета роботи: розробка цифрового двійника компресорної станції з інтеграцією в автоматизовану систему керування (АСК) для забезпечення моніторингу в реальному часі, виявлення аварій та прогнозування параметрів у умовах нестабільного електроживлення, зумовленого зовнішніми факторами.

Гіпотеза дослідження полягає в тому, що реалізація ЦД АСК на основі відкритих технологій (MQTT, Node-RED, InfluxDB, Grafana) дозволить досягти суттєвого покращення ключових показників ефективності порівняно з традиційними системами моніторингу. Очікується значне підвищення оперативності контролю, скорочення часу реагування на критичні події та зменшення експлуатаційних витрат за рахунок автоматизації рутинних операцій та превентивного аналізу даних.

Для досягнення поставленої мети в рамках розробки цифрового двійника АСК компресорної станції необхідно виконати наступні завдання:

1. Дослідити теоретичні основи та сучасні технологічні рішення для побудови цифрових двійників промислових об’єктів та систем моніторингу в реальному часі.

2. Обґрунтувати вибір архітектури та технологічного стеку (MQTT, Node-RED, InfluxDB, Grafana) для реалізації цифрового двійника компресорної станції.

3. Розробити математичну модель та програмний симулятор (на базі Python) роботи компресорної станції, здатний генерувати реалістичні дані про ключові параметри (напруга, тиск, температура) в нормальному, аварійному та відновлювальному режимах.

4. Реалізувати центральну систему збору, обробки та маршрутизації даних на базі платформи Node-RED, включаючи логіку детекції аварійних ситуацій за пороговими значеннями.

5. Створити інтерактивний операторський веб-дашборд (Node-RED Dashboard) та мобільний інтерфейс (PWA) для візуалізації стану системи в реальному часі, відображення графіків та керування.

6. Налаштувати систему сповіщень через Telegram-бота для миттєвого оповіщення персоналу про критичні стани та аварійні події на компресорній станції.

7. Провести комплексне тестування розробленого прототипу, оцінити його функціональність, час реакції на аварії та ефективність роботи в умовах імітації нестабільного електроживлення.

Методи дослідження: теоретичний аналіз, системне проектування, об'єктно-орієнтоване програмування (Python), візуальна розробка засобами low-code (Node-RED), моделювання, тестування.

Наукова новизна: уперше запропоновано цілісну легковагову архітектуру ЦД для АСК, адаптовану до умов енергодефіциту, що інтегрує симулятор, low-code логіку, PWI та Telegram-сповіщення в єдиний контур реального часу.

Практична значущість: створено працюючий прототип, готовий до адаптації на реальних об’єктах; методика може бути тиражована для інших типів

критичної інфраструктури. Результати роботи можуть бути використані промисловими підприємствами − газотранспортними компаніями (компресорні станції), енергетичними об'єктами (котельні, ТЕЦ), водоканалами (насосні станції). Також навчальними закладами для навчання студентів на лабораторних стендах, симуляція промислових процесів без реального обладнання, системними інтеграторами: як референсна архітектура для проектів IoT/Industry 4.0, для швидкого прототипування рішень.

У першому розділі МР було розглянуто теоретичні основи концепції цифрового двійника у контексті автоматизованих систем керування технологічними процесами. Описано багаторівневу архітектуру ЦД, що охоплює фізичний, керуючий та цифровий рівні, а також ключові компоненти його структури. Розглянуто методологічні аспекти розробки ЦД.

В другому розділі МР було розглянуто методологічні засади та інструментарій розробки ЦД АСК компресорної станції газотранспортної системи. Проведено аналіз проблемної області в контексті енергодефіциту та воєнного стану. На основі аналізу сформовано та формалізовано науково-технічну задачу дослідження, а також сформульовано комплекс функціональних та нефункціональних вимог до системи цифрового двійника.

В третьому розділі надано детальний опис інженерної реалізації та верифікації системи цифрового двійника АСК компресорної станції. Визначено науково-технічні результати. Проведено експериментальну верифікацію.

1 ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ СУЧАСНИХ РІШЕНЬ ДЛЯ СТВОРЕННЯ ЦИФРОВИХ ДВІЙНИКІВ ДЛЯ АВТОМАТИЗОВАНИХ СИСТЕМ КЕРУВАННЯ

1.1 Концепція цифрового двійника: еволюція та роль в індустрії 4.0

Концепція цифрового двійника, що динамічно відображає стан і поведінку фізичного об'єкта у віртуальному просторі, стала одним із ключових драйверів четвертої промислової революції. Хоч сам термін набув широкої популярності у 2010-х роках, його теоретичні основи сягають раніших принципів комп'ютерного моделювання та симуляції.

Цифровий двійник (ЦД) − це віртуальна модель фізичного об’єкта або системи, яка представляє його реальний аналог у цифровому світі. Інженери та фахівці в галузі інформаційних технологій можуть використовувати їх для кращого розуміння, тестування та оптимізації процесів. Рішення ухвалюються на основі аналізу даних, що призводить до підвищення ефективності та якості продукції [1].

Сучасне розуміння цифрового двійника виходить за рамки простої статичної моделі. Це динамічна цифрова репрезентація фізичного об'єкта, процесу або системи, яка підтримує свою актуальність завдяки постійному двонаправленому обміну даними з реальним світом у режимі, близькому до реального часу. Головна відмінність ЦД від класичного симулятора полягає у його глибокій інтеграції з оперативними даними: він отримує дані з реальних датчиків, систем керування та іншої інфраструктури підприємства, трансформуючись з інструменту аналізу в ядро для підтримки прийняття рішень та автоматизованого керування.

В контексті Індустрії 4.0 та промислового Інтернету речей ЦД виступає центральною ланкою кіберфізичних систем. Він забезпечує зв'язок між фізичним рівнем, представленим обладнанням, виконавчими механізмами та сенсорами, і кібернетичним рівнем, на якому відбувається аналітика, прогнозування та

оптимізація за допомогою передових алгоритмів. Таке поєднання дозволяє реалізувати ключові принципи сучасного виробництва: прозорість процесів, предиктивне обслуговування обладнання, гнучку адаптацію до змін попиту та значну економію ресурсів.

У виробничій галузі ЦД використовуються для моніторингу та оптимізації виробничих процесів. Вони допомагають уникнути часу простою, інтегруючи дані в режимі реального часу, можуть розпізнавати аномалії на ранній стадії і пропонувати варіанти проактивного обслуговування. Моделювання складних сценаріїв особливо важливе в галузях, де безпека і точність мають вирішальне значення. В енергетичній галузі цифрові двійники підтримують моніторинг мереж і систем, що також допомагає скоротити викиди. Вони допомагають у логістиці, роблячи ланцюги постачання більш ефективними [1].

1.2 Класифікація та архітектурні підходи до цифрових двійників в АСК ТП

У автоматизованих системах керування технологічними процесами (АСК ТП) спостерігається різноманіття підходів до класифікації цифрових двійників, зумовлене складністю об’єктів керування. У науково-технічній літературі зафіксовано виокремлення декількох ієрархічних рівнів деталізації. Найнижчим є компонентний рівень, на якому моделюються окремі складові елементи обладнання. Наступний — агрегатний рівень — передбачає опис цілісних технологічних установок або машин. Системний цифровий двійник охоплює комплексні об’єкти, такі як виробнича лінія або цех, тоді як процесний рівень відтворює технологічний процес в цілому, що є особливо важливим для безперервних типів виробництв. Окремо класифікуються цифрові двійники типу «прототип», що застосовуються на стадії проєктування, та типу «екземпляр», що відповідає конкретному фізичному об’єкту в режимі експлуатації. Для практичних завдань АСК ТП найбільш поширеним є гібридний підхід, за якого

модель використовується одночасно як інструмент налагодження та супроводу експлуатації системи.

Рівень абстракції цифрового двійника залежить від умов використання, для яких він розробляється. *Окремий цифровий двійник* – це єдине ціле, що створює цінність без потреби подальшого розбиття (рис. 1.1) [13].

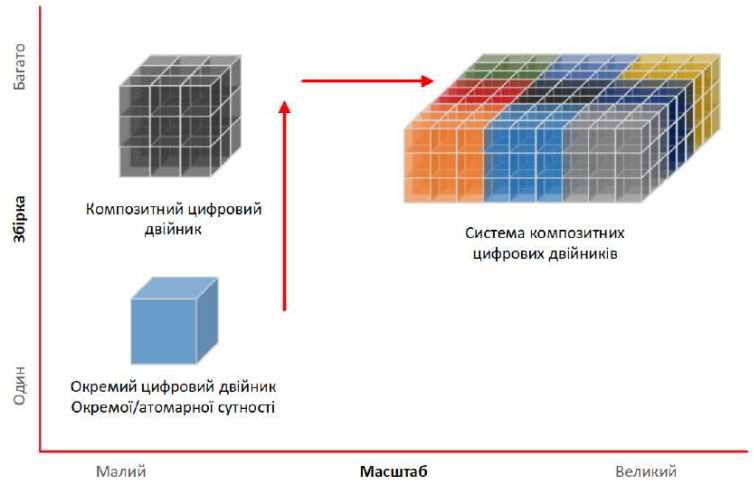


Рисунок 1.1 − Утворення композитного цифрового двійника [13]

В композитних цифрових двійниках взаємозв'язки між компонентами (рис. 1.2) класифікуються на три основні типи.

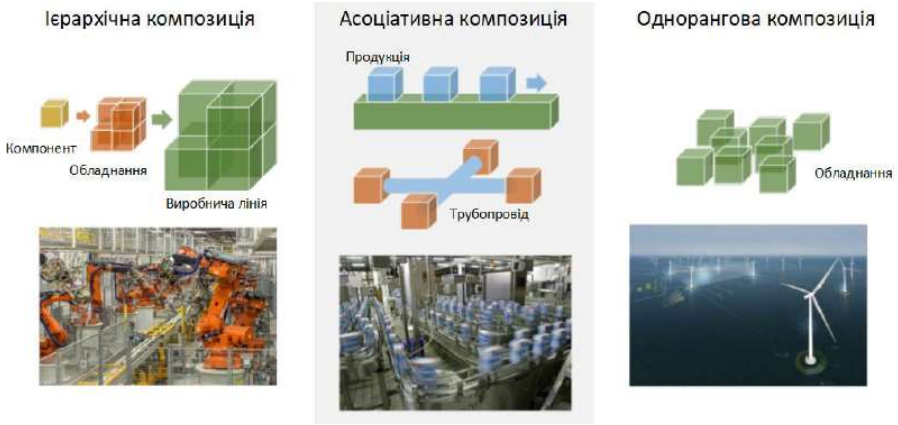


Рисунок 1.2 − Типи зв'язків між цифровими двійниками в композитній структурі [13]

1. Ієрархічні: ці зв'язки відтворюють структуру реальних фізичних об'єктів, де цифрові двійники нижчого рівня агрегуються в об'єкти вищого рівня. Зокрема, набір цифрових двійників-компонентів утворює ЦД одиниці устаткування, кілька таких двійників устаткування інтегруються в цифрового двійника виробничої лінії, а сукупність двійників ліній формує ЦД підприємства в цілому;

2. Асоціативні: даний тип відображає функціональні або процесні залежності між об'єктами, аналогічні до зв'язків між їхніми фізичними прототипами. Наприклад, ЦД газопроводу може бути пов'язаний асоціативними зв'язками із ЦД процесів видобутку та споживання газу;

3. Однорангові (peer-to-peer): такі зв'язки виникають між рівноправними вузлами, наприклад, у групі однотипного або схожого обладнання, що виконує ідентичні чи близькі функції. Загальний результат роботи такої груви є адитивною сумою ефектів, що генеруються кожним окремим елементом [13].

Цифрові двійники різних об'єктів, аналогічно реальним системам, пов'язані семантичними зв'язками. Без них двійник стає лише ізольованим сховищем даних. Автоматизована побудова таких зв'язків − ключова проблема через гетерогенність джерел, форматів і часу надходження інформації.

Створення єдиної точки інтеграції даних життєвого циклу та зв'язків між двійниками забезпечує ряд переваг:

− платформа для аналітики та ШІ, що слугує основою для розширеної аналітики та штучного інтелекту, які можуть бути як зовнішніми застосуваннями, так і вбудованими модулями, перетворюючи двійник на автономний інтелектуальний об'єкт;

− віртуальні сенсори дозволяють створювати високоточні програмні датчики для параметрів, які неможливо або важко виміряти фізично;

− прогнозування та симуляція дають можливість моделювати та прогнозувати поведінку технологічного процесу;

− валідація даних, вбудовані фізичні моделі допомагають узгоджувати показання датчиків, відрізняючи відмови обладнання від похибок вимірювань, підвищуючи достовірність інформації;

− колаборація: спрощують спільну роботу на всіх етапах життєвого циклу, зменшуючи витрати часу на пошук і обмін даними;

− експлуатація та обслуговування: допомагають запобігати дорогим простоям шляхом швидкого вирішення проблем;

− підвищення якості: усунення помилок, викликаних недостатньою або несвоєчасною інформацією;

− глобальна доступність: забезпечують цілодобовий доступ до експертизи та даних, дозволяючи віддалену підтримку та швидке реагування [13].

Таким чином, цифрові двійники є системною методологією та інструментарієм для представлення складних систем, що забезпечує ефективний моніторинг, діагностику, прогнозування та оптимізацію.

1.2.1 Архітектура цифрового двійника для АСК ТП

Архітектура ЦД для АСК ТП як правило, має багаторівневу структуру (рис. 1.3) [6].

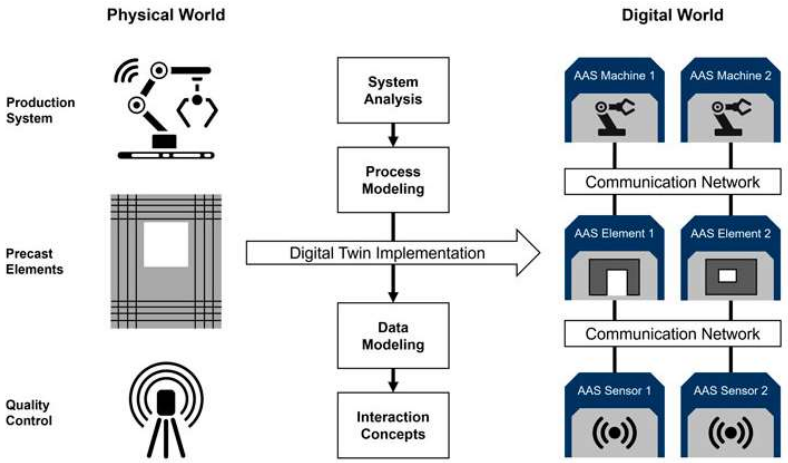


Рисунок 1.1 − Механічні системи цифрових двійників Індустрії 4.0 [6]

Нижній, фізичний рівень, включає безпосередньо об'єкт керування, датчики та виконавчі механізми. Рівень керування представлений програмованими логічними контролерами, дисперсними системами керування або SCADA-системами. Найвищий, цифровий рівень, об'єднує математичну модель процесу, сервіси збору та обробки даних, аналітичні модулі, а також інтерфейси для взаємодії з персоналом. Критично важливим є шар інтеграції, який забезпечує зв'язок між рівнями. Для цього широко застосовуються промислові протоколи зв'язку, такі як OPC UA, Modbus TCP, та інтерфейси виробників обладнання, що дозволяє створювати як системи одностороннього моніторингу, так і повноцінні контури зворотного зв'язку для керування.

1.3 Функціональні можливості та сценарії застосування в АСК ТП

Функціонал ЦД в контурі АСК можна умовно поділити на три основні групи застосувань. Перша група – моніторинг та віртуальне відображення. Цифровий двійник стає прозорим віртуальним інтерфейсом, через який можна спостерігати за станом процесу в режимі реального часу, у тому числі за параметрами, недоступними для прямого вимірювання. Це підвищує ситуаційну обізнаність операторів.

Друга група – діагностика та предиктивне обслуговування. Аналізуючи розбіжності між поведінкою реального об'єкта та його ідеальної цифрової копії, система може виявляти аномалії на ранніх стадіях, діагностувати поступове погіршення характеристик обладнання та прогнозувати ймовірність відмови. Це дозволяє перейти від планово-попереджувального до обслуговування за фактичним станом.

Третя, найбільш складна група – оптимізація та керування. Цифровий двійник виступає як полігон для тестування нових законів регулювання, режимів роботи або алгоритмів адаптивного керування без ризику для реального обладнання. Він дозволяє виконувати «what-if» аналіз, оптимізувати витрату

енергії та сировини, а також служити тренажером для операторів для відпрацювання дій в аварійних ситуаціях.

Особливої уваги заслуговують сценарії віртуального введення в експлуатацію. Інтегруючи цифровий двійник з реальним програмованим логічним контролером, інженери можуть провести повноцінне фабричне або монтажне приймальне випробування програмного забезпечення АСК, значно знижуючи ризики та терміни при запуску нових об'єктів.

1.4 Методологія розробки цифрового двійника

Структурно ЦД (рис. 1.4) складається з даних, розрахункових моделей та сервісних інтерфейсів, аналогічно до об'єктно-орієнтованої парадигми програмування, що включає дані, методи та інтерфейси [13].

Рисунок 1.4 − Концептуальна модель структури цифрового двійника [13]

Для динамічного представлення реальних об'єктів екземпляри ЦД повинні бути інтегровані з їхніми фізичними прототипами, часто в режимі реального часу, з метою збору та структурування даних відповідних реальних об'єктів. Для аналізу цих даних цифровий двійник має включати розрахункові та аналітичні

моделі, що дозволяють описувати, діагностувати, прогнозувати та імітувати стани та поведінку реальних об'єктів і систем. Результати такого аналізу можуть бути інтегровані з бізнес-логікою та цілями для оптимізації виробничих процесів. Для реалізації цього процесу розробка ЦД повинна передбачати сервісні інтерфейси для інтелектуальних промислових застосувань, що забезпечують доступ до даних та аналітичних результатів.

Дані: ЦД має містити дані про свій фізичний відповідник, необхідні моделям для відображення та оцінки його станів і поведінки. Часто ці дані охоплюють повний життєвий цикл реального об'єкта. Для випадку промислового устаткування це можуть бути:

− дані етапу проектування: специфікації, конструкторські моделі, виробничі процеси, технічні характеристики;

− дані етапу виробництва: інформація про персонал, обладнання, матеріали, технології виробництва та контроль якості;

− дані етапу експлуатації: параметри встановлення та конфігурації, поточні та історичні операційні дані, записи про технічне обслуговування;

− дані етапу утилізації [13].

Також у складі даних можуть присутні комерційні відомості, такі як записи транзакцій.

Моделі. Для коректного відображення, аналізу та прогнозування станів і поведінки фізичного об'єкта цифровий двійник обов'язково включає розрахункові та аналітичні моделі, включаючи моделі для формування керуючих впливів на основі бізнес-логіки та цільових показників. До таких моделей належать фізико-хімічні, інженерні, імітаційні, а також статистичні моделі та моделі, що базуються на методах машинного навчання і штучного інтелекту. Для поліпшення інтерпретації людиною поточних станів також можуть застосовуватися просторові моделі та моделі з елементами доповненої реальності.

Сервіси (інтерфейси). Для забезпечення доступу до власних даних і функціональності цифровий двійник надає набір сервісних інтерфейсів,

призначених для взаємодії з промисловими застосунками або іншими цифровими двійниками.

Незважаючи на велику різноманітність реальних об'єктів, необхідно визначати високорівневі інваріантні структури з уніфікованим набором атрибутів даних і моделей, що забезпечує стандартизований механізм їх виклику та взаємодії.

Цифрові двійники можуть створюватися відповідно до типів їх фізичних аналогів (рис. 1.5). Конкретні екземпляри генеруються на основі типових шаблонів з урахуванням конфігурації середовища. Аналогічно можуть бути встановлені типізовані логічні зв'язки між окремими екземплярами.

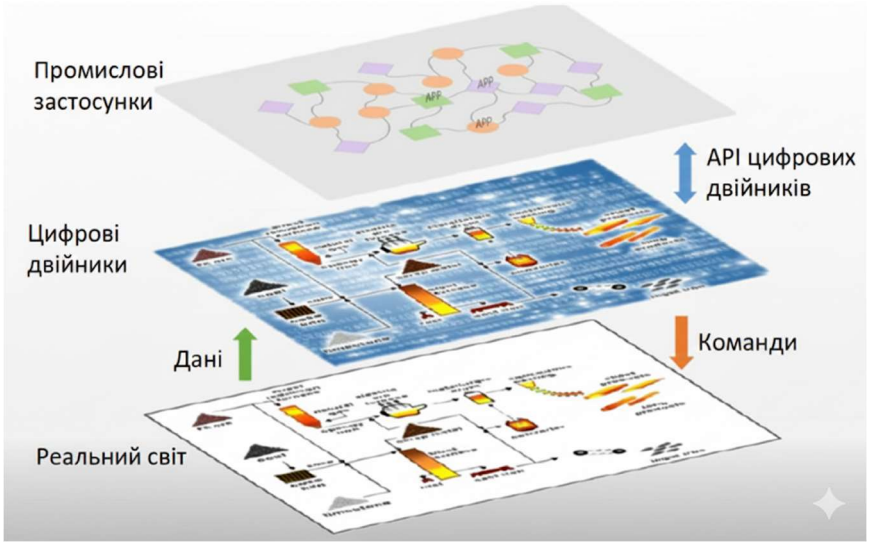


Рисунок 1.5 − Цифрові двійники як поєднання процесів розробки та використання [13]

Розробка ЦД передбачає створення віртуальної моделі фізичної системи, здатної відтворювати її поведінку в реальному часі на основі даних від сенсорів та керувальних сигналів. Ключовим технічним аспектом є забезпечення синхронізації між фізичним і цифровим об’єктами, що досягається шляхом використання математичних моделей, методів ідентифікації систем, спостерігачів стану або спеціалізованих алгоритмів керування. Архітектура ЦД

повинна забезпечувати безперервний обмін даними, коректну обробку затримок і завад у мережі, а також мінімізацію похибки між реальними та змодельованими параметрами, що є критично важливим для його практичного використання в промислових системах.

Деякі технічні аспекти цифрового двійника, що розглядаються далі, схематично представлені на рис. 1.6 та охоплюють такі ключові компоненти:

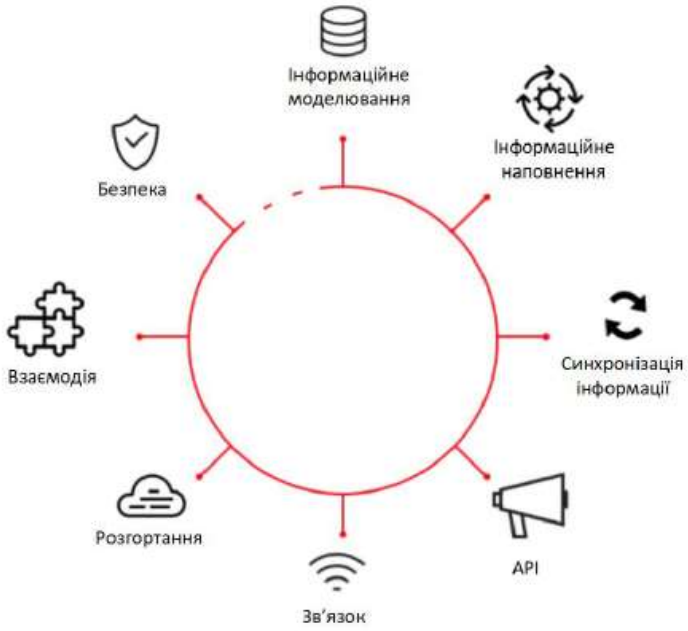


Рисунок 1.6 − Основні технічні компоненти цифрового двійника [13]

– інформаційне моделювання полягає у формуванні цифрової моделі об'єкта на основі математичних, логічних або імітаційних представлень, що відображають його структуру, параметри та поведінку;

– інформаційне наповнення забезпечує актуалізацію ЦД шляхом надходження даних від сенсорів, систем керування та зовнішніх інформаційних джерел;

– синхронізація інформації відповідає за узгодження станів фізичного об'єкта та його ЦД в реальному або квазіреальному часі з урахуванням затримок і похибок передавання даних;

– API (інтерфейси прикладного програмування) забезпечують стандартизований доступ до функцій ЦД та інтеграцію з іншими програмними системами, сервісами й платформами;

– зв’язок є ключовим фактором взаємодії між ЦД та фізичними об’єктами, оскільки визначає надійність, швидкодію та цілісність обміну даними;

– розгортання ЦД передбачає вибір середовища виконання (хмарне, локальне або гібридне), а також налаштування обчислювальних і мережевих ресурсів. Один екземпляр цифрового двійника доцільно визначати як головний (master-екземпляр) (рис. 1.7), у якому зосереджені базові моделі, ключові дані та відповідні формалізовані визначення, що зберігаються в централізованому репозиторії. Інші екземпляри цифрового двійника можуть створюватися на його основі та адаптуватися відповідно до конкретних сценаріїв використання;

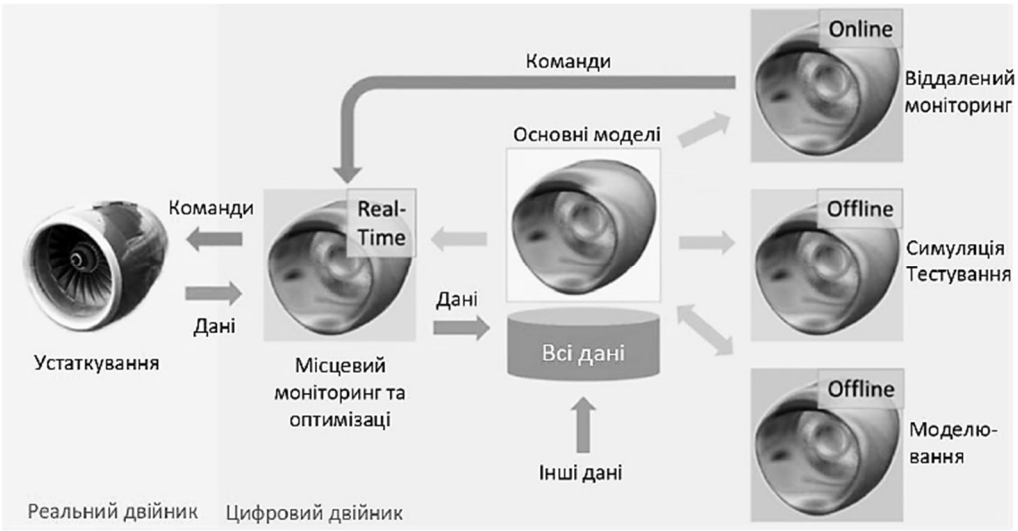


Рисунок 1.7 − Структура розгортання цифрових двійників в системі [13]

– безпека взаємодії охоплює захист даних, автентифікацію доступу та запобігання несанкціонованому втручанню в роботу цифрового двійника;

– взаємодія забезпечує обмін інформацією між цифровим двійником, користувачами та іншими цифровими системами з метою моніторингу, аналізу та прийняття рішень.

1.5 Стандарти та моделі цифрових двійників

Стандартизація цифрових двійників активно розвивається, хоча термін не завжди використовується. МЕК 62832 описує цифрові активи в структурі заводу, не називаючи їх цифровими двійниками. ISO/IEC JTC1 виділяє їх як пріоритетну сферу досліджень із можливістю співпраці з open source спільнотою. У 2019 році ISO/TC 184 зафіксувала відсутність стандартизованої архітектури даних для цифрових двійників, що спричинило створення робочої групи, а IEEE запустила проект P2806 для системної архітектури цифрових представлень фізичних об’єктів у виробництві. Подібний підхід реалізовано в ISO/AWI 23247 для підключення елементів двійників через інтерфейси та функції.

Платформа Industrie 4.0 впровадила Asset Administration Shell для цифрових двійників у розумному виробництві, підтриманому Францією, Італією та Німеччиною. ISO TS 18101-1 визначає цифровий двійник як цифровий актив, на якому можна надавати послуги, що приносять організаційне значення, незалежно від фізичної природи. Open source проекти Eclipse BaSyx, Ditto та Vorto забезпечують SDK і фреймворки для цифрових двійників відповідно до Asset Administration Shell. Крім ISO та IEC, над стандартами цифрових представлень у контексті IoT працюють і консорціуми, як-от W3C Web of Things [13].

У даному розділі МР було розглянуто теоретичні основи концепції ЦД у контексті АСК ТП, проаналізовано еволюцію ЦД від статичних моделей до динамічних, інтегрованих у реальний час кіберфізичних систем, що є ключовим елементом Індустрії 4.0. Надано класифікацію ЦД за рівнями деталізації та типами зв’язків у композитних структурах.

Описано багаторівневу архітектуру ЦД, що охоплює фізичний, керуючий та цифровий рівні, а також ключові компоненти його структури: дані, моделі та сервісні інтерфейси. Визначено основні функціональні можливості цифрових двійників у АСК ТП − від моніторингу в реальному часі до предиктивного обслуговування та оптимізації керування.

Окремо розглянуто методологічні аспекти розробки ЦД: інформаційне моделювання, синхронізацію, інтеграцію через API, а також питання безпеки та розгортання. Зазначено, що, незважаючи на активний розвиток стандартів (ISO/IEC, IEC 62832, Asset Administration Shell, IEEE P2806), ще не сформовано уніфікованої архітектури даних, що відкриває простір для інноваційних рішень, особливо в умовах обмежених ресурсів.

Таким чином, виконаний аналіз підтверджує наукову та практичну актуальність розробки адаптованого, легковагового цифрового двійника для АСК критичних об’єктів, зокрема компресорної станції, з урахуванням специфіки української промисловості та сучасних викликів, пов’язаних із енергонестабільністю та гібридною війною. Отримані теоретичні положення становлять міцну основу для подальшої проектної реалізації системи.

2 МЕТОДОЛОГІЯ ТА ІНСТРУМЕНТАРІЙ РОЗРОБКИ ЦИФРОВОГО ДВІЙНИКА ДЛЯ АСК КОМПРЕСОРНОЇ СТАНЦІЇ

2.1 Аналіз проблемної області та умов експлуатації в контексті енергодефіциту

В сучасних умовах Україна переживає суттєву трансформацію вимог до експлуатації ключової промислової інфраструктури, зумовлену повномасштабними воєнними діями. Починаючи з 2022 року, внаслідок бойових дій об'єкти критичної енергетичної інфраструктури України постраждали від низки ударів, що призвело до системних перебоїв у електропостачанні та гострого дефіциту генеруючих потужностей. Пошкодження генерації та мереж призвели до погодинних відключень, підкреслюючи потребу в ефективному моніторингу та управлінні енергоспоживанням для підтримки автономності критичних об’єктів [14].

Компресорні станції газотранспортної системи, особливо вразливі під час відключень електроенергії, зазнають зростання аварійності та втрат через пошкодження обладнання та недопостачання газу. Це підкреслює необхідність підвищення енергетичної автономності та стабільності їх роботи [15].

Унаслідок систематичних обстрілів енергетичних об’єктів у 2025 році було зафіксовано понад 1 200 збоїв у роботі електромереж та близько 70 інцидентів на об’єктах паливно-енергетичного комплексу, що спричиняє часті аварійні відключення в Україні та дефіцит доступних потужностей. Планові обмеження електропостачання у 2024 році сумарно тривали до 1 951 години по всій країні, що значно впливає на стабільність роботи критичних об’єктів, включно з об’єктами газотранспортної інфраструктури. Окремі випадки підтверджують прямі атаки на компресорні станції газотранспортної системи, що посилює потребу в автономних системах живлення та управління технологічними процесами в умовах енергодефіциту [16].

Систематичні атаки на енергетичну систему призвели до формування нового операційного середовища, для якого характерні:

− високий ризик раптових та тривалих відключень електроживлення;

− обмежений фізичний доступ персоналу через загрози безпеки та комендантську годину;

− деградуюча якість каналів через перевантаження мережі та фізичні пошкодження інфраструктури.

Традиційні автоматизовані системи керування технологічними процесами демонструють критичну вразливість у таких умовах через централізовану архітектуру, повну залежність від зовнішнього живлення та відсутність механізмів автономного функціонування. Компресорні станції газотранспортної системи, як об'єкти безперервного циклу роботи, потребують особливого підходу до забезпечення стійкості, оскільки їх раптова зупинка загрожує каскадними технологічними аваріями, пошкодженням обладнання та перервами у постачанні газу.

2.2 Постановка задачі з урахуванням умов енергодефіциту та автономності

Розглянемо типовий критичний сценарій для компресорної станції промислового підприємства. При раптовому відключенні зовнішнього електроживлення спостерігається наступна послідовність подій:

− t = 0 с: електропривід компресорів миттєво зупиняється внаслідок втрати живлення мережі 220В;

− t = 0-5 с: тиск у технологічному контурі починає різко падати через відсутність підтримки компресією (швидкість падіння ~ 0,3 – 0,5 бар/с);

− t = 5-10 с: традиційна АСК ТП, що живиться від тієї ж мережі, втрачає працездатність, роблячи об'єкт «сліпим» − оператори не мають інформації про стан параметрів;

− t = 10-300 с: без керованого втручання виникає загроза розгерметизації системи, гідравлічних ударів, пошкодження компресорного обладнання та запірної арматури.

Критичність ситуації посилюється тим, що в умовах повітряної тривоги або комендантської години оперативний персонал може бути фізично відсутнім на об'єкті або не мати можливості швидко дістатися до нього. За даними операційних служб, середній час прибуття технічного персоналу до об'єкта в умовах тривоги становить 45 − 90 хвилин, що є критично недостатнім для запобігання розвитку аварійної ситуації.

Аналіз показує, що традиційні АСК ТП та SCADA-системи володіють рядом обмежень у контексті забезпечення стійкості до енергетичних збоїв:

- відсутність автономності: повна залежність від зовнішнього електроживлення без вбудованих механізмів продовження критичних функцій. Центральні сервери та HMI-станції SCADA виходять з ладу без резервного живлення, порушуючи зв'язок з польовими пристроями та призводячи до «сліпих зон» у мережі [17];
- централізована архітектура: втрата зв'язку з центральним сервером призводить до повної втрати керованості об'єктом. Обчислення зосереджені на хмарі/сервері, тому при збоях даних не обробляються локально, що критично для АСК ТП у реальному часі (затримки 100-200 мс) [18];
- обмежені канали комунікації: залежність від провідних мереж, вразливих до фізичних пошкоджень;
- висока вартість резервування: дублювання повної SCADA-інфраструктури економічно недоцільне для більшості об'єктів.

В умовах частих енергокриз та атак legacy SCADA на базі застарілих протоколів (IEC 60870-5-104) не витримують, спричиняючи простої та збитки. Низька топологічна надмірність та неоновлені ПЗ посилюють ризики, особливо в електроенергетиці з обмеженими ресурсами.

2.3 Вибір методології розробки

Враховуючи дослідницький характер роботи, потребу у швидкому отриманні працюючого прототипу та ітеративному вдосконаленні функціоналу, для розробки цифрового двійника було обрано ітеративно-інкрементальну методологію**.** Цей підхід передбачає послідовну реалізацію та інтеграцію незалежних модулів системи з подальшим циклом тестування та покращення.

Етапи розробки включають наступне:

− прототипування ядра системи: створення математичної моделі, симулятора та організація базового каналу зв'язку (MQTT);

− розробка логіки керування: реалізація в середовищі Node-RED потоків обробки даних, детекції аномалій та автоматичного реагування;

− створення інтерфейсів користувача: розробка операторського дашборду, мобільного PWA та каналів сповіщення;

− інтеграція та тестування: комплексна перевірка взаємодії всіх компонентів, вимірювання продуктивності та відпрацювання аварійних сценаріїв.

Така методологія дозволяє мінімізувати ризики розробки, забезпечити гнучкість у прийнятті рішень та демонструвати проміжні результати на кожному етапі.

2.4 Формалізація науково-технічної задачі

На основі аналізу проблемної області сформульовано головну задачу дослідження: розробити архітектуру та працюючий прототип цифрового двійника АСК компресорної станції, здатного забезпечити її функціональну стійкість в умовах енергодефіциту та обмеженого доступу.

2.4.1 Вхідні дані та обмеження

Об'єкт моделювання:

Компресорна станція КС-3 з контрольованими параметрами:

- напруга живлення $U \in [0, 250]$ В (номінал 220 В ± 10%);
- тиск газу в системі $P \in [0, 10]$ бар (робочий діапазон 5,5 – 7,0 бар);
- температура газу $T \in [0, 100]$ °C (номінал 40 − 50 °C);
- стан клапанів $V \in \{OPEN, CLOSED\}$;
- стан компресора $C \in \{RUNNING, STOPPED\}$.

Аварійні (критичні) події:

- E_1: втрата електроживлення (U < 100 В);
- E_2: критичне падіння тиску (P < 4 Бар);
- E_3: перегрів (перевищення температури) (T > 60 °C).

Експлуатаційні обмеження: висока ймовірність відключень електромережі, можлива відсутність персоналу на об'єкті, необхідність мобільного доступу до інформації.

Потрібно розробити систему − цифровий двійник автоматизованої системи управління компресорною станцією, яка забезпечує наступне:

− безперервний моніторинг та візуалізацію:відображення параметрів у реальному часі (оновлення ≤ 1 с) через веб-дашборд та мобільний PWA-інтерфейс;

− автоматичне аварійне реагування: виявлення критичних подій за час t_detection ≤ 3 с та виконання захисних дій (імітація закриття клапанів) за t_response ≤ 1 с.

− багатоканальне сповіщення: гарантована доставка алертів через Telegram-бота та браузерні сповіщення з часом t_notification ≤ 5 с;

− дистанційний моніторинг та управління: доступ до Dashboard через веб-інтерфейс та мобільні пристрої (PWA);

− забезпечення візуалізації параметрів у реальному часі з частотою оновлення ≤ 1 секунда;

− можливість віддаленого керування з будь-якої точки при наявності мережевого з'єднання;

− автономність та локальне логування: збереження функцій моніторингу та запису даних у CSV-файли при відсутності зовнішнього живлення або мережі;

− аналітику в реальному часі: розрахунок та відображення сесійної статистики (тривалість, середні значення, кількість аварій) з можливістю експорту всіх даних.

2.5 Вимоги до системи цифрового двійника

2.5.1 Функціональні вимоги

1 Моніторинг: збір та відображення параметрів U(напруга), P (тиск), T (температура), статусів клапанів та компресора з частотою 1 Гц;

2 Детекція аномалій: кКласифікація стану на рівні: INFO (норма), WARNING (U=180 − 200 В, P=4 − 5 бар), CRITICAL (U < 180 В, P < 4 бар, T > 60 °C);

3 Автоматичне реагування: при виявленні події E_1 система має автоматично ініціювати послідовність: фіксація часу − логічне закриття клапанів − генерація сповіщення;

4 Візуалізація: надання інтерфейсу з аналоговими індикаторами (gauge), графіками реального часу, таблицею подій та панеллю статистики;

5 Логування та експорт: автоматичний запис кожного виміру в CSV-файл. Наявність кнопки для експорту всіх даних сесії.

2.5.2 Нефункціональні вимоги

1 Продуктивність: затримка в контурі «симулятор – Dashboard» < 1 с. Час оновлення UI < 500 мс;

2 Надійність: коефіцієнт готовності системи моніторингу ≥ 0,99. Можливість роботи основних компонентів від резервного живлення ≥ 2 години;

3 Масштабованість: архітектура дозволяє додавання нових датчиків або станцій без зміни ядра системи;

4 Безпека: аутентифікація доступу до Dashboard;

5 Шифрування комунікації по каналах передачі даних (Telegram API);

6 Економічна ефективність: використання виключно відкритого ПЗ (Open Source). Можливість розгортання на обладнанні низької та середньої потужності (одноплатні комп'ютери типу Raspberry Pi);

7 Вартість апаратної частини для одного об'єкта ≤ 20 000 грн. (на 1 об'єкт).

2.5.3 Апаратні та програмні обмеження

Апаратна платформа:

Обчислювальна платформа: ПК або одноплатний комп'ютер ОЗП ≥ 2 ГБ з наявністю резервного живлення (UPS або АКБ ємністю, достатньою для 2+ годин автономної роботи).

Мережеве підключення: Ethernet або WiFi, з можливістю fallback на мобільний інтернет.

Програмне середовище:

- операційна система: Windows 10/11 або Linux-based (Ubuntu, Debian);
- мова програмування симулятора: Python 3.12+;
- платформа обробки даних: Node-RED 3.x;
- брокер повідомлень: Eclipse Mosquitto 2.x;
- канали сповіщення: Telegram Bot API;
- сховище даних**:** через пріоритет завдань швидкої розробки та автономності обрано спрощену модель – локальне зберігання в CSV-файлах та в пам'яті Node-RED, що достатньо для демонстрації повного циклу роботи системи.

2.6 Концепція цифрової стійкості як методологічна основа

Концепція «цифрової стійкості» (Digital Resilience) визначає здатність промислових систем, бізнес-процесів та інфраструктури адаптуватися до криз, збоїв чи змін за допомогою цифрових технологій, забезпечуючи безперервність операцій та швидке відновлення. Вона поєднує стійкість до кібератак,

енергетичних відключень та економічних потрясінь з елементами IoT, AI та edge-обчислень для локальної обробки даних [19].

Ця концепція реалізується через чотири рівні автономності, що забезпечують поступовий перехід від централізованого керування до саморегулюючих систем у промисловості. Ці рівні базуються на принципах Industry 4.0 та CPS, дозволяючи системам працювати автономно під час збоїв, таких як енергетичні відключення [19]:

1 Рівень повної функціональності: за наявності мережі та живлення – повний цикл: симуляція − обробка − візуалізація − сповіщення – логування;

2 Рівень обмеженої функціональності: при втраті зовнішнього живлення – система працює від резервного джерела, зберігаючи функції моніторингу, локального логування та аварійного сповіщення;

3 Рівень мінімальної функціональності: при втраті мережі – працюють локальна обробка даних, детекція аварій та запис у файл. Дані синхронізуються після відновлення зв'язку;

4 Рівень аварійного зв'язку: при повній відмові локальної інфраструктури – спрацьовує незалежний канал сповіщення через Telegram Bot API, що працює через мобільний інтернет.

Такий підхід усуває єдину точку відмови та забезпечує життєздатність системи навіть при часткових руйнуваннях інфраструктури.

Ефективність розробленого рішення оцінюється за такими критеріями:

- Технічні: час повного циклу реагування на аварію (від її імітації до сповіщення) ≤ 5 с;
- Функціональні: коректна класифікація 100% імітованих аварійних сценаріїв; безперервний запис усіх даних у лог-файл;
- Експлуатаційні: інтуїтивний інтерфейс, доступний з ПК та мобільного пристрою; час розгортання системи на чистому апаратному забезпеченні ≤ 4 години;
- Економічні: вартість програмної частини = 0 (відкрите ПЗ); орієнтовна окупність апаратних витрат за рахунок запобігання однієї аварії.

2.7 Обмеження дослідження

У рамках даного дослідження слід зазначити ряд обмежень, що визначають його межі та напрями подальшого розвитку.

По-перше, об'єктом моделювання виступає програмний симулятор фізичного процесу, а не реальне промислове обладнання. Такий підхід, хоч і дозволяє безпечно та контрольовано відтворювати задані аварійні та штатні сценарії, не може врахувати повний спектр нюансів, властивих роботі конкретного фізичного обладнання в реальних умовах.

По-друге, функціональні рамки розробленого прототипу обмежені задачами оперативного моніторингу, порогової детекції аномалій та базового автоматичного реагування. Розробка складних прогностичних моделей на основі методів штучного інтелекту для предиктивного аналізу стану виходить за межі поточної роботи і становить окремий науковий інтерес.

По-третє, архітектура системи передбачає використання спрощеної файлової моделі (CSV) для збереження даних. Це рішення є цілком достатнім для демонстрації життєздатності концепції (Proof of Concept) та відповідає вимогам до простоти й автономності, проте накладає обмеження на глибину та швидкість історичного аналізу порівняно зі спеціалізованими промисловими базами даних часових рядів.

Нарешті, експериментальна перевірка системи проводилася в умовах стабільної лабораторної мережі. Валідація працездатності критично важливих компонентів, таких як резервні канали зв'язку через стільникові мережі в умовах імітованого чи реального відключення електроживлення (блэкауту), є предметом наступного етапу – польових випробувань на реальній або максимально наближеній до реальної інфраструктурі.

2.8 Аналіз та обґрунтування вибору технологічного стеку

Технологічний стек

- симулятор: Python 3.x з бібліотеками paho-mqtt, json;

- платформа обробки даних: Node-RED;
- мережевий протокол: MQTT (брокер Mosquitto);
- візуалізація: Node-RED Dashboard;
- зберігання даних: CSV файли, SQLite;
- мобільний доступ: PWA (Progressive Web App);
- сповіщення: Telegram Bot API, Browser Notifications.

2.8.1 Вихідні критерії вибору технологій

Вибір технологічних інструментів для втілення концепції ЦД детермінований комплексом раніше визначених системних вимог. Першочерговими критеріями є відкритість ПЗ та відсутність ліцензійних витрат, що має вирішальне значення для впровадження рішення на українських промислових підприємствах в умовах обмежених фінансових ресурсів. Не менш важливою вимогою є простота інтеграції окремих компонентів та низький поріг освоєння технологій, що забезпечує оперативну розробку функціонального прототипу та його подальше супроводження інженерними кадрами без необхідності глибокої програмної підготовки.

Архітектура системи обов'язково повинна підтримувати роботу в режимі реального часу та асинхронну обробку потоків даних, оскільки це становить фундаментальну основу для ефективного моніторингу технологічних параметрів і здійснення оперативних керуючих впливів. Враховуючи сучасні реалії, стійкість до нестабільних або тимчасово втрачених мережевих з'єднань виступає ключовою вимогою для забезпечення безперебійної роботи в умовах погіршення якості комунікаційних каналів. Обрана технологічна платформа також повинна володіти якостями модульності та масштабованості, що відкриває можливості для поступового розширення функціональності системи, підключення нових типів сенсорів та адаптації до змінних виробничих сценаріїв. Нарешті, наявність активної спільноти розробників та якісної документації розглядається як гарантія доступності необхідних ресурсів для технічного налагодження,

вирішення експлуатаційних проблем та довгострокової підтримки всього програмного комплексу.

2.8.2 Порівняльний аналіз альтернатив та обґрунтування вибору

На основі детального аналізу поставлених вимог було обґрунтовано вибір технологічного стеку для кожного рівня архітектури ЦД.

На рівні комунікації та передачі даних основною альтернативою розглядалися такі протоколи, як HTTP REST API, WebSocket, OPC UA та Modbus TCP. Остаточний вибір припав на протокол MQTT (Message Queuing Telemetry Transport) з брокером Eclipse Mosquitto. Цей легковаговий протокол архітектури «видавець-підписник» спеціально розроблений для IoT-систем та умов нестабільних мереж. Його ключові переваги включають мінімальні накладні витрати (заголовок від 2 байт), вбудовані механізми надійності через три рівні якості обслуговування (QoS), стійкість до розривів зв'язку завдяки persistent session та можливість ефективної маршрутизації даних через ієрархічну структуру топіків. Eclipse Mosquitto обраний як найпоширеніший високопродуктивний брокер з відкритим кодом [20].

На рівні бізнес-логіки та інтеграції серед альтернатив прямого програмування (Python/Node.js) та інших low-code платформ (n8n, Huginn) пріоритет було надано середовищу Node-RED. Це low-code рішення для потокового програмування ідеально відповідає задачі швидкої розробки.

Node-RED є інструментом візуального програмування, призначеним для інтеграції апаратних пристроїв, API та онлайн-сервісів [21]. Його ключовою особливістю є браузерний редактор, який дозволяє створювати логічні потоки шляхом компонування готових вузлів з палітри та розгортати їх одним кліком. Цей підхід суттєво підвищує швидкість розробки та спрощує подальший супровід системи.

Значущість Node-RED як промислового рішення підтверджується його інтеграцією в офіційні платформи автоматизації, такі як SIMATIC IOT2000 та Harmony iPC, що робить його доступним інструментом для вирішення завдань

збору та обробки даних у виробничих середовищах [21]. Критично важливим для промислового застосування є наявність спеціалізованої екосистеми готових вузлів, що забезпечують підтримку ключових промислових протоколів зв'язку (наприклад, MQTT, OPC UA, Modbus) та інтеграцію з зовнішніми сервісами (HTTP, Telegram, бази даних). Високі показники завантаження цих модулів свідчать про активне використання платформи в реальних проектах [21].

Таким чином, Node-RED поєднує переваги low-code підходу, що забезпечує низький поріг входження та швидку ітерацію, з потужністю та стабільністю, необхідними для промислової експлуатації. Його можливості, включаючи вбудований засіб створення операторських інтерфейсів (Dashboard), дозволяють реалізувати повний цикл — від збору даних з обладнання до візуалізації та управління, ставлячи його в один ряд із професійними комерційними SCADA-рішеннями.

На рівні моделювання та генерації даних, на відміну від спеціалізованих середовищ (MATLAB/Simulink, LabVIEW), було обрано мову програмування загального призначення Python. Цей вибір обумовлений її простотою, читабельністю, що дозволяє швидко ітераційно розробляти та модифікувати математичну модель, а також найбільшою екосистемою бібліотек для наукових обчислень та прототипування (paho-mqtt, json тощо). Python також забезпечує гнучкість для майбутнього розширення моделей за рахунок інтеграції бібліотек машинного навчання.

На рівні збереження даних відбулася корекція початкового задуму. Незважаючи на те, що комбінація часової бази даних InfluxDB та платформи візуалізації Grafana є індустріальним стандартом для потужної аналітики, для етапу доказу концепції було прийнято рішення використовувати спрощене локальне файлове сховище (CSV/SQLite).

У межах архітектури ЦД для забезпечення його функціональної стійкості в умовах нестабільності зовнішніх комунікацій та енергопостачання реалізовано механізм локального кешування даних. Цей механізм базується на використанні легковагових файлових сховищ у форматі CSV та вбудованої бази даних SQLite,

інтегрованих у потоки обробки середовища Node-RED. Даний підхід дозволяє гарантовано зберігати потоки телеметричних даних (показники сенсорів, повідомлення MQTT) безпосередньо на edge-пристрої, усуваючи критичну залежність від зовнішніх сервісів [22].

CSV-формат застосовується для надійного потокового логування даних у реальному часі. Його переваги − мінімальне навантаження на систему, можливість безперебійного додавання записів та універсальність для подальшого імпорту в інструменти аналітики. SQLite, як повноцінна вбудована SQL-база, використовується для більш структурованого зберігання, коли виникає потреба у складних вибірках або агрегації історичних даних [22].

Практична реалізація в Node-RED будується за логікою потоку MQTT in − Обробка − CSV/SQLite out − Dashboard. Спеціальні вузли забезпечують безперервний запис даних, який продовжується навіть при тимчасовій недоступності мережі або інтерфейсу оператора. Ця архітектура ефективно усуває ризик втрати даних, властивий централізованим SCADA-системам під час каскадних збоїв, що набуває особливої актуальності в умовах сучасних викликів для критичної інфраструктури [22].

Це обґрунтовано потребою пріоритезувати максимальну автономність та простоту розгортання в умовах енергодефіциту. Такий підхід спрощує архітектуру, підвищує надійність, усуваючи зайві точки відмови, забезпечує абсолютну автономність запису даних навіть при повній відсутності мережі та повністю відповідає функціональним вимогам щодо логування та експорту. Крім того, це значно скоротило час розробки та зменшило вимоги до апаратних ресурсів.

На рівні інтерфейсу користувача та сповіщень було обрано комбінацію трьох технологій: Node-RED Dashboard дозволяє швидко створювати базовий операторський інтерфейс без написання коду, технологія PWA (Progressive Web App) перетворює цей веб-інтерфейс на додаток для мобільного пристрою, забезпечуючи офлайн-доступ та push-сповіщення. Для гарантованого аварійного сповіщення обрано Telegram Bot API як найдоступніший, безкоштовний та

надійний канал, що працює через стільникові мережі незалежно від стану локальної інфраструктури.

2.8.3 Синтез архітектури на основі обраного стеку

Обраний технологічний стек забезпечує реалізацію принципів цифрової стійкості через чітко сегментовану, але взаємопов'язану архітектуру. Її основу складає модель взаємодії «видавець-підписник» (Pub/Sub), де Python-симулятор, відповідальний за генерацію даних фізичної моделі, виступає видавцем, а платформа Node-RED, що реалізує бізнес-логіку, − підписником. Така архітектура забезпечує асинхронний обмін даними та слабку зв'язність компонентів, усуваючи пряму залежність логіки обробки від стану генератора даних.

Ключовою якістю системи є чіткий поділ функціональних рівнів. Python-компонент інкапсулює математичну модель об'єкта, Node-RED − алгоритми моніторингу, аналізу та автоматичного реагування, а рівень представлення (Dashboard, Telegram Bot) − інтерфейси для взаємодії з людиною. Це дозволяє незалежно розвивати та підтримувати кожен модуль.

Архітектура реалізує принцип поступового зниження функціональності (graceful degradation). Система розроблена так, що відмова нефункціональних компонентів (наприклад, графічного інтерфейсу Dashboard) не призводить до повної втрати працездатності. Критичні функції, такі як обробка даних, локальне логування та аварійне сповіщення через альтернативні канали (Telegram), продовжують виконуватися, забезпечуючи базову життєздатність цифрового двійника в умовах часткових збоїв інфраструктури.

В даному розділі магістерської роботи було розглянуто методологічні засади та інструментарій розробки цифрового двійника автоматизованої системи керування (АСК) компресорної станції газотранспортної системи. Проведено аналіз проблемної області в контексті енергодефіциту та воєнного стану, що виявив критичну вразливість традиційних SCADA-систем і автоматизованих систем керування технологічними процесами (АСК ТП) до раптових та тривалих

відключень електроживлення, обмеженого фізичного доступу персоналу та деградації каналів зв'язку.

На основі аналізу сформовано та формалізовано науково-технічну задачу дослідження, а саме − розробити архітектуру та працюючий прототип цифрового двійника АСК компресорної станції, здатного забезпечити її функціональну стійкість в умовах енергодефіциту та обмеженого доступу. Визначено контрольовані параметри об'єкта, аварійні події, встановлено вхідні дані та обмеження.

Також сформульовано комплекс функціональних та нефункціональних вимог до системи цифрового двійника, що включають безперервний моніторинг, автоматичне аварійне реагування, багатоканальне сповіщення, локальне логування та економічну ефективність. Також визначено апаратні та програмні обмеження для реалізації. Обрано ітеративно-інкрементальну методологію розробки, яка передбачає циклічне прототипування, тестування та вдосконалення модулів системи, що дозволяє мінімізувати ризики та забезпечити гнучкість процесу.

Запропоновано концепцію «цифрової стійкості» (Digital Resilience) як методологічну основу, яка визначає здатність системи до адаптації та автономного функціонування в умовах кризових явищ. Концепція реалізується через чотири рівні поступового зниження функціональності (graceful degradation), що гарантують життєздатність системи навіть при часткових руйнуваннях інфраструктури.

Було проведено аналіз та обґрунтовано вибір технологічного стеку розробки на кожному рівні архітектури:

− для комунікації обрано протокол MQTT з брокером Eclipse Mosquitto;

− для реалізації бізнес-логіки – low-code середовище Node-RED;

− для моделювання – мову програмування Python;

− для зберігання даних – спрощену локальну файлову модель (CSV/SQLite);

− для інтерфейсу – комбінацію Node-RED Dashboard, PWA та Telegram Bot API для сповіщень.

Окреслено обмеження дослідження, серед яких: використання програмного симулятора замість реального обладнання, обмеженість функціоналу прототипу базовими задачами моніторингу та реагування, використання спрощеної моделі зберігання даних та проведення валідації в умовах лабораторної мережі.

Отже, сформовано повноцінну методологічну, технічну та архітектурну основу для подальшого етапу проектування та практичної реалізації цифрового двійника АСК компресорної станції, адаптованого до екстремальних умов експлуатації в Україні.

3 ІНЖЕНЕРНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ЦИФРОВОГО ДВІЙНИКА АСК КОМПРЕСОРНОЇ СТАНЦІЇ

3.1 Розробка симулятора компресорної станції на основі Python

Розробка цифрового двійника АСК компресорної станції ґрунтується на архітектурних принципах, викладених у розділі 1, та методології цифрової стійкості з розділу 2. Система реалізує гібридну архітектуру, що відповідає принципам розподілених систем та промислового Інтернету речей (IIoT) та поєднує:

1. Рівень фізичної абстракції (Python-симулятор) − відповідає за емуляцію поведінки реального об'єкта;
2. Рівень обробки даних (Node-RED) − реалізує бізнес-логіку та аналітику;
3. Рівень інтерфейсів (Dashboard, Telegram) − забезпечує взаємодію з оператором.

Архітектура реалізує принцип слабкої зв'язності компонентів через MQTT-брокер, що дозволяє незалежно модифікувати та масштабувати окремі модулі. Такий підхід відповідає концепції композитних цифрових двійників [13], розглянутий у підрозділі 1.2.

Детальний опис архітекури системи цифрового двійника АСК компресорної станції представлено в Додатку А.

Відповідно до концепції цифрової стійкості (розділ 2.6), система реалізує чотири рівні функціональності:

1. Повна функціональність − усі компоненти активні, мережа стабільна;
2. Обмежена функціональність − втрата зовнішнього живлення, робота від UPS;
3. Мінімальна функціональність − втрата мережі, локальне логування та обробка;

4. Аварійний зв'язок – лише Telegram-сповіщення через мобільний інтернет.

Загальна схема архітектури наведена на рис. 3.1.

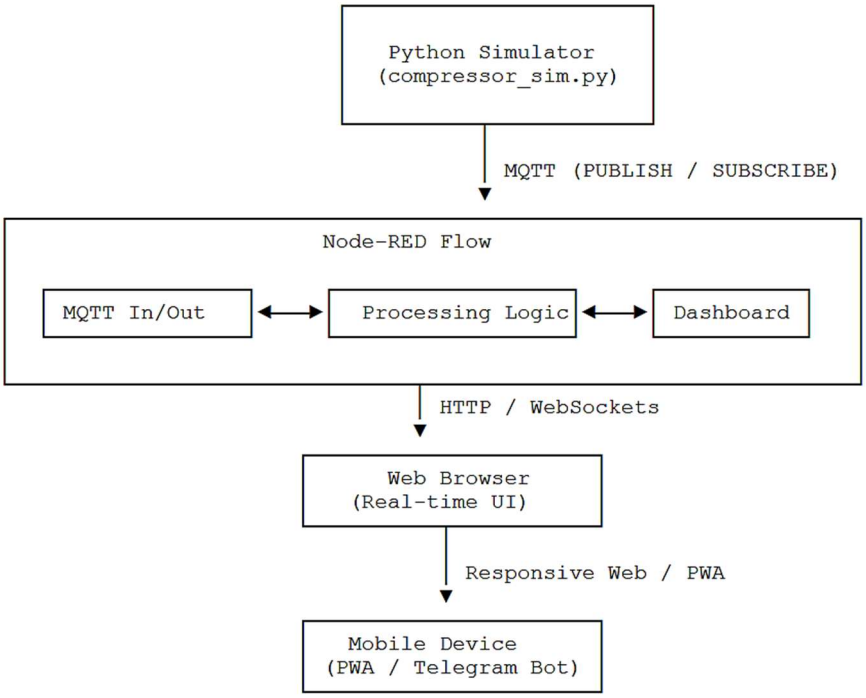


Рисунок 3.1 – Загальна архітектура системи цифрового двійника

Критичним технічним рішенням став відхід від централізованої InfluxDB+Grafana на користь локального CSV+SQLite сховища. Це рішення обґрунтоване потребою максимізації автономності в умовах енергодефіциту, що є ключовою проблемою, визначеною в підрозділі 2.1.

3.1.1 Архітектура та принципи роботи симулятора

Першим етапом практичної реалізації системи стала розробка програмного симулятора, що імітує функціонування реальної компресорної станції КС-3. Симулятор розроблено на мові програмування Python 3.14 з використанням об'єктно-орієнтованої парадигми, що забезпечує високу модульність коду та простоту його подальшої модифікації. Вибір Python обґрунтований його

екосистемою для наукових обчислень та швидкого прототипування, що відповідає вимогам ітеративної розробки (підрозділ 2.3).

Основу архітектури симулятора становить клас CompressorStation, який інкапсулює всі характеристики фізичного об'єкта та методи їх зміни. Концептуально модель базується на принципі скінченного автомату з трьома дискретними станами, що відображає реальну поведінку промислового об'єкта:

− NORMAL − нормальний режим роботи з параметрами в межах номінальних значень;

− CRITICAL − аварійний стан, спричинений відключенням електроживлення;

− RECOVERING − режим відновлення після усунення причини аварії.

На рис. 3.2 представлено діаграму станів симулятора компресорної станції (state diagram).

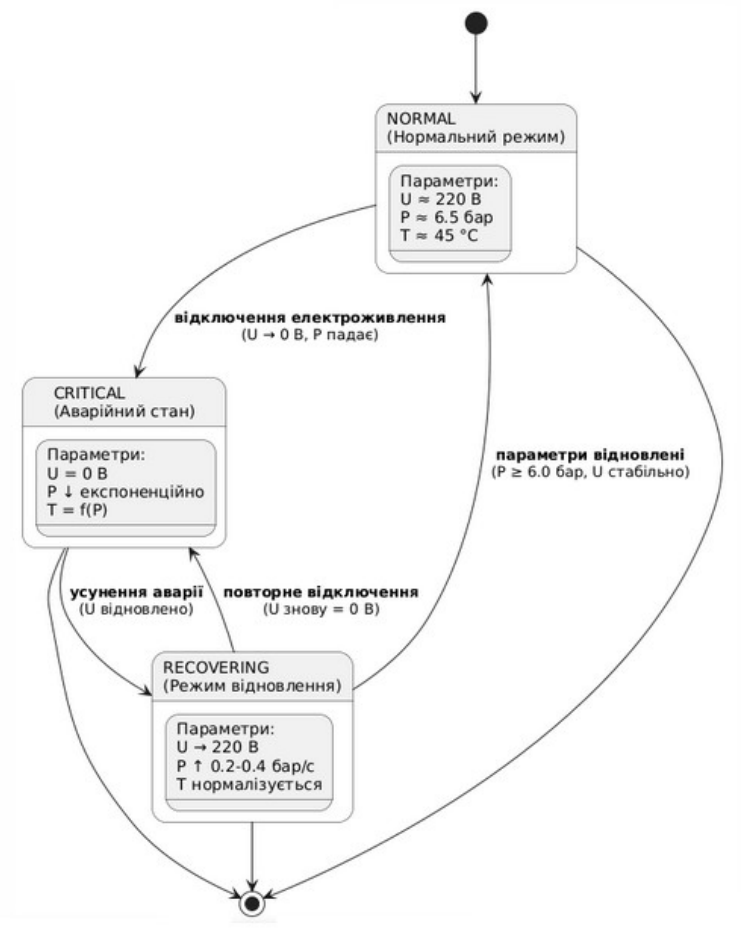


Рисунок 3.2 − Діаграма станів симулятора компресорної станції

Перехід між станами визначається часовим циклом роботи програми. Кожен цикл симулятора відповідає 1 секунді реального часу, що забезпечує частоту оновлення даних 1 Гц − достатню для коректного моніторингу динамічних процесів згідно з теоремою Котельникова-Найквіста (теореми відліків), яка є фундаментальною в теорії цифрової обробки сигналів та системах реального часу [24].

Практичне застосування теореми у системі:

1. Симулятор (1 Гц) забезпечує мінімально достатню частоту для моніторингу основних технологічних параметрів, хоча формально 1 Гц < 20 Гц, а для практичних цілей промислового моніторингу це є прийнятним, оскільки:

- температурні та тискові процеси мають високу інерційність;
- миттєве відключення живлення все одно фіксується (стан змінюється в межах одного циклу);

2. Система моніторингу реального часу працює з подвійною частотою (2 Гц) у внутрішніх буферах Node-RED для забезпечення коректної інтерполяції на графіках;

3. Переваги частоти 1 Гц:

- мінімальне навантаження на мережу, кожне MQTT-повідомлення має розмір ~500 байт, що дає трафік ~4 кбіт/с;
- сумісність з обмеженими каналами, працює навіть при низькій швидкості Інтернет (GPRS/EDGE з пропускною здатністю 50-200 кбіт/с), що особливо актуально в умовах пошкодженої інфраструктури або віддалених промислових об'єктів;
- висока енергоефективність;
- оптимальна для сприйняття оператором.

3.1.2 Математична модель технологічного процесу

Для коректної імітації фізичних процесів розроблено спрощену математичну модель, що описує зміну ключових параметрів компресорної станції.

Модель напруги живлення:

У нормальному режимі напруга моделюється як випадкова величина з нормальним розподілом навколо номінального значення:

$$U(t) = U\_nom + \xi(t)),$$

де U_nom = 220 В − номінальна напруга мережі,

$\xi(t) \sim N(0, \sigma^2)$ − гаусівський шум з σ = 2,5 В, що відповідає допустимим коливанням ±5 В (±2.3%).

При імітації аварії відбувається миттєве обнулення напруги:

$$U(t\_аварія) = 0 \text{ В}$$

Модель тиску газу:

Тиск у системі залежить від стану компресора. У робочому режимі підтримується близько номінального значення з невеликими флуктуаціями:

$$P(t) = P\_nom + \delta P(t),$$

де P_nom = 6.5 бар, $\delta P(t) \sim U(-0{,}2, 0{,}2)$ − рівномірний шум.

При зупинці компресора тиск падає з експоненційною швидкістю:

$$P(t) = P(t-\Delta t) - \alpha \cdot \Delta t,$$

де $\alpha \in [0.3, 0.5]$ бар/с − швидкість падіння, обрана на основі типових характеристик компресорних установок потужністю 5 − 10 кВт.

Модель температури:

Температура газу змінюється як функція режиму роботи:

$T(t) = T\_nom + \beta \cdot P(t) + \varepsilon(t)$

де T_nom = 45 °C, β = 0,5 °C/бар − коефіцієнт залежності температури від тиску (адіабатичне стиснення), $\varepsilon(t) \sim U(-2, 3)$ − температурні флуктуації.

3.1.3 Структура класу та реалізація логіки

Для реалізації моделі скінченного автомата з трьома станами розроблено клас CompressorStation на мові Python. Об'єктно-орієнтований підхід забезпечив

інкапсуляцію стану об'єкта та чітке розмежування відповідальності між методами класу. Повний програмний код симулятора наведено в додатку Б.

Структура класу CompressorStation (фрагмент ініціалізації):

```
class CompressorStation:
    def __init__(self, station_id="КС-3"):
        self.station_id = station_id
```

Нормальні параметри:

```
        self.voltage = 220.0
        self.pressure = 6.5
        self.temperature = 45.0
        self.status = "NORMAL"
        self.valve_status = "OPEN"
```

Параметри роботи:

```
        self.compressor_running = True
        self.power_failure = False
        self.emergency_mode = False
```

Історія подій:

```
        self.events = []
```

Клас містить атрибути для зберігання поточних значень фізичних параметрів, булевих прапорців стану та списку подій для логування історії змін.

Логіка зміни станів реалізована через три основні методи:

1. `simulate_normal_operation()` – генерує параметри нормального режиму з додаванням стохастичного шуму:

Метод simulate_normal_operation:

```
def simulate_normal_operation(self):
```

Нормальний режим роботи:

```
    self.voltage = 220 + random.uniform(-5, 5)
    self.pressure = 6.5 + random.uniform(-0.2, 0.2)
    self.temperature = 45 + random.uniform(-2, 3)
    self.status = "NORMAL"
```

2. `simulate_power_failure()` – моделює аварію з відключення електроживлення та активує автоматичне реагування.

Метод simulate_power_failure (фрагмент):

```
def simulate_power_failure(self):
```

Аварія – відключення електрики:

```
    self.voltage = 0
    self.power_failure = True
    self.compressor_running = False
    self.status = "CRITICAL"
```

Тиск падає:

```
  if self.pressure > 2.0:
        self.pressure -= random.uniform(0.3, 0.5)

    if not self.emergency_mode:
        self.trigger_emergency()
```

3. `recover_power()` – імітує процес відновлення після усунення аварії:

```
def recover_power(self):
```

Відновлення електрики:

```
  self.voltage = 220 + random.uniform(-5, 5)
    self.power_failure = False
    self.status = "RECOVERING"

    if self.pressure < 6.5:
        self.pressure += random.uniform(0.2, 0.4)
```

3.1.4 Інтеграція з MQTT протоколом

Для забезпечення взаємодії з іншими компонентами системи симулятор інтегровано з MQTT-брокером за допомогою бібліотеки paho-mqtt версії 2.1.0. Архітектура взаємодії реалізована за моделлю «видавець» (Publisher), де

симулятор публікує дані у топік compressor/station3 з рівнем якості QoS 2 (гарантована доставка).

MQTT інтеграція в головному циклі:

MQTT Configuration:

```
MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC = "compressor/station3"
def main():
    station = CompressorStation("KS-3")
    client = mqtt.Client()

    try:
        client.connect(MQTT_BROKER, MQTT_PORT, 60)
        client.loop_start()
    except Exception:
        client = None

    cycle = 0

    while True:
        cycle += 1
```

Логіка зміни станів за часовим графіком:

```
        if cycle < 10:
            station.simulate_normal_operation()
        elif cycle < 25:
            station.simulate_power_failure()
        else:
            station.recover_power()
            if cycle > 40:
                cycle = 0

        data = station.get_data()
```

Публікація в MQTT:

```
        if client:
            payload = json.dumps(data)
            client.publish(MQTT_TOPIC, payload)


        time.sleep(1)
```

Структура повідомлення MQTT формується методом get_data(), який серіалізує стан станції у JSON-формат згідно з наступною схемою:

Структура JSON-повідомлення з прикладом:

```
{
  "station_id": "КС-3",
  "timestamp": "2024-12-09T14:30:05.123456",
  "voltage": 220.5,
  "pressure": 6.5,
  "temperature": 45.2,
  "status": "NORMAL",
  "valve_status": "OPEN",
  "compressor_running": true,
  "emergency_mode": false,
  "events": ["14:30:02 - Подія 1", ...]
}
```

3.1.5 Валідація коректності симуляції

Для підтвердження адекватності розробленої моделі проведено серію експериментів з порівнянням поведінки симулятора з теоретичними очікуваннями та доступними даними з технічної документації реальних компресорних установок.

Сценарій моделювання:

1. Нормальна робота (0 − 10 циклів):
- напруга: 220 ± 5V;
- тиск: 6,5 ± 0,2 Bar;
- температура: 45 ± 3°C.
2. Аварійний режим (10 − 25 циклів):
- повне відключення живлення (0V);
- падіння тиску до 2.0 Bar;
- зростання температури;
- автоматичне закриття клапанів.
3. Відновлення (після 25 циклів):
- поступове відновлення параметрів;

- перезапуск компресора;
- перехід до нормального режиму.

Скріншоти конфігураційних файлів та етапи процесу запуску системи ЦД компресорної станції представлено на рис. 3.3 – 3.5.

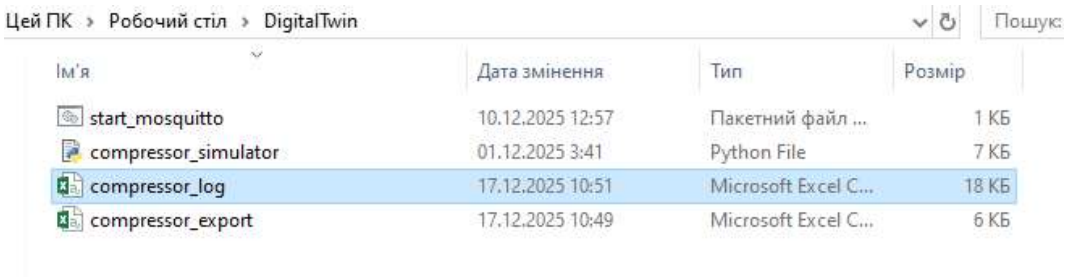


Рисунок 3.3 − Файлова структура ЦД компресорної станції

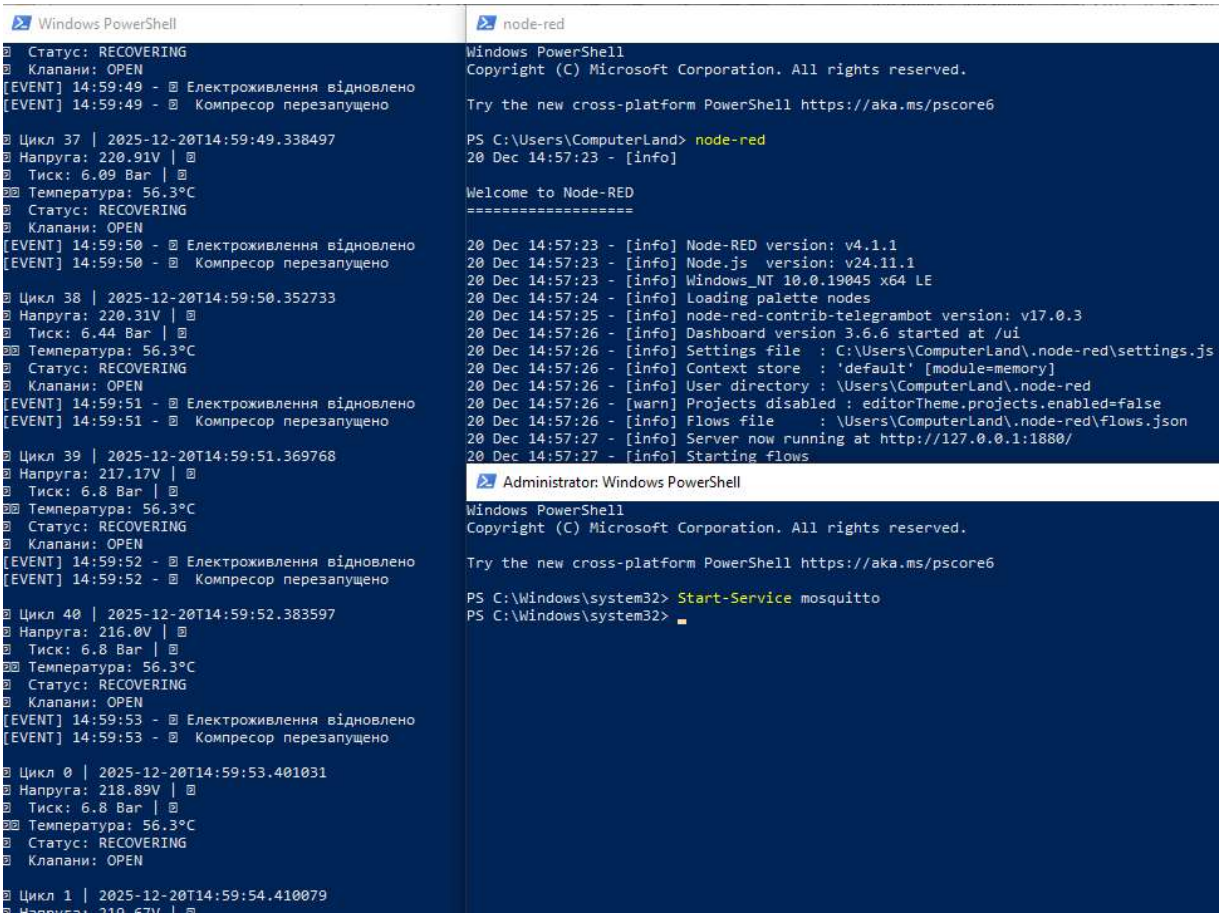


Рисунок 3.4 – Скріншоти запуску симулятора, брокера mosquitto та Node-Red

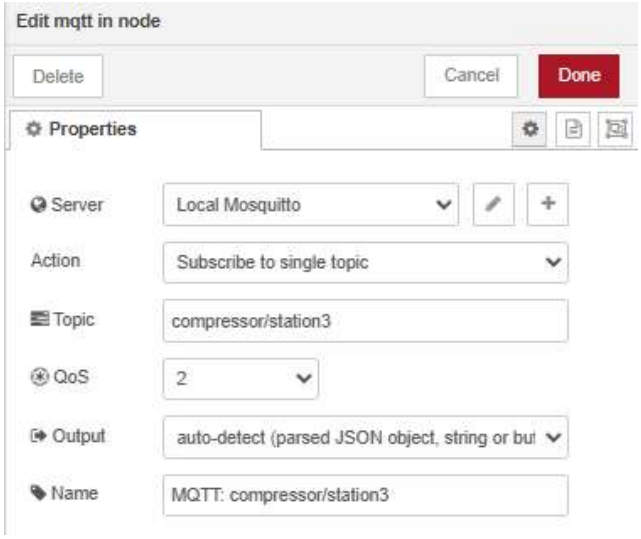


Рисунок 3.5 – Налаштування ноди MQTT

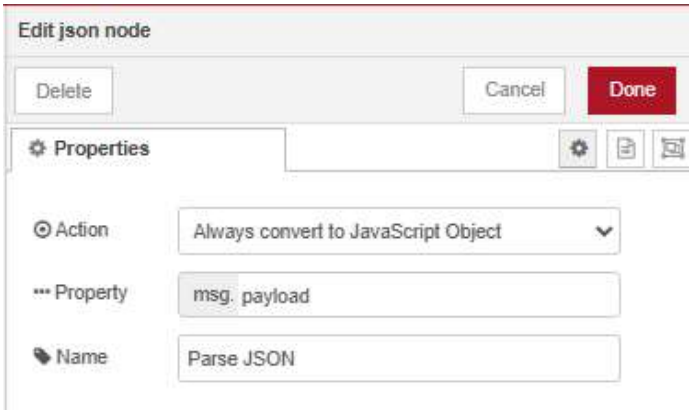


Рисунок 3.6 – Налаштування ноди JSON

Тест 1: Стабільність параметрів у нормальному режимі

Проведено 300-секундний тест роботи у нормальному режимі. Статистичний аналіз згенерованих даних показав:

- середнє значення напруги: μ_U = 219,8 В (відхилення від номіналу 0,1%);
- стандартне відхилення: σ_U = 2,47 В (в межах заданого σ = 2,5 В);
- аналогічна точність для тиску та температури.

Параметри системи в нормальному режимі представлено на рис. 3.7.

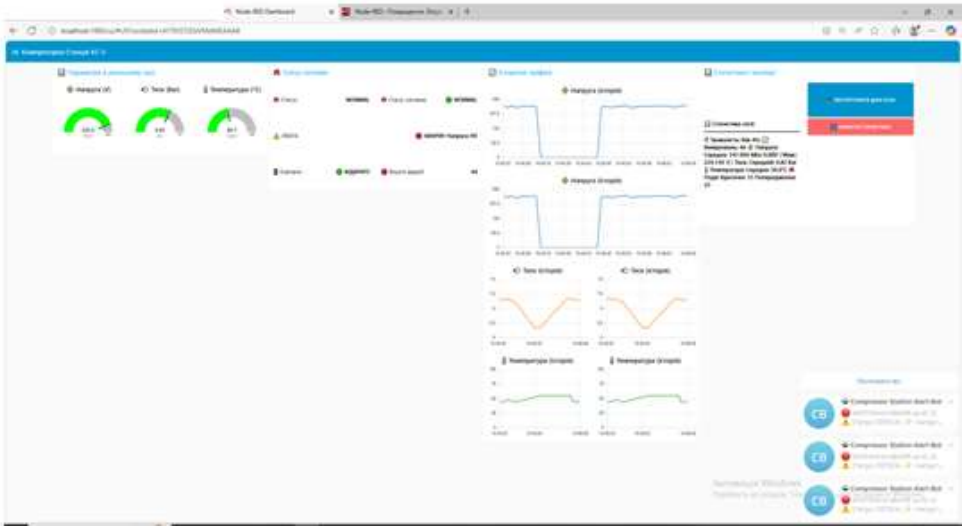


Рисунок 3.7 − Графіки параметрів у нормальному режимі (300 с)

Тест 2: Динаміка аварії

Проведено імітацію аварії з фіксацією часових характеристик (рис. 3.8):

– t = 0 с: U = 220 В − U = 0 В (миттєве падіння);

– t = 1 − 15 с: Тиск (P) падає з 6,5 до 2,1 бар (середня швидкість 0,37 бар/с);

– час спрацювання аварійної логіки: < 0,1 с після детекції U = 0.

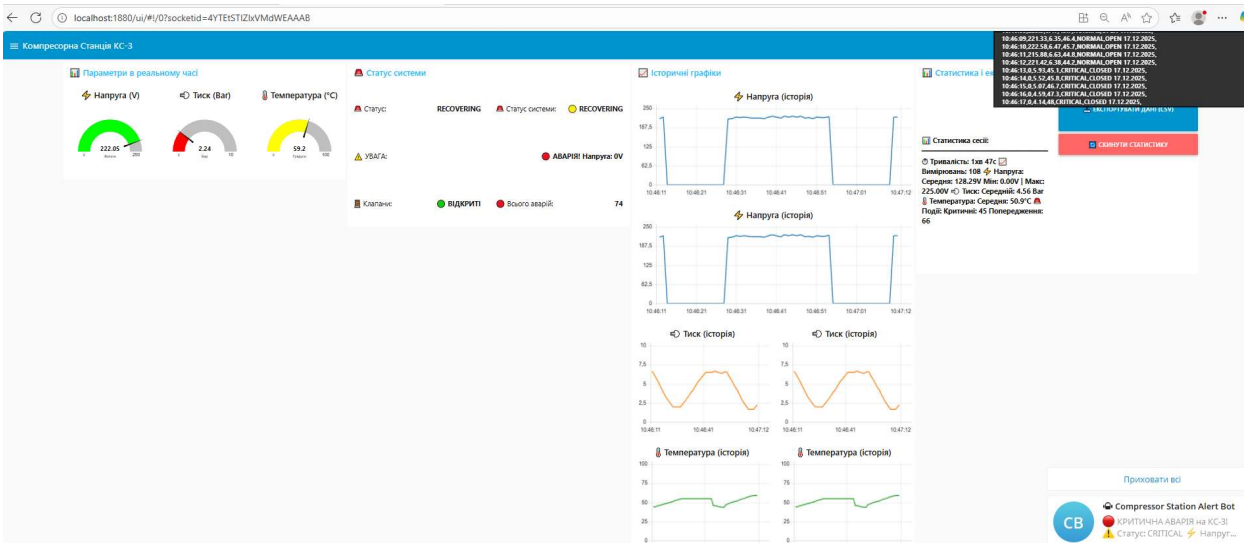


Рисунок 3.8 − Графік перехідних процесів під час аварії

Результати валідації підтверджують, що розроблений симулятор коректно відтворює задані сценарії та може бути використаний як надійне джерело тестових даних для відлагодження інших компонентів системи.

3.2 Реалізація центральної логіки обробки даних у Node-RED

3.2.1 Архітектура потоків обробки даних

Node-RED як low-code платформа для flow-based programming забезпечує візуальне проектування логіки обробки даних через граф взаємопов'язаних вузлів. Розроблена архітектура складається з чотирьох функціональних шарів (рис. 3.9):

1. Шар прийому даних − MQTT Input вузол для підписки на топік compressor/station3;
2. Шар десеріалізації − JSON Parser для перетворення текстового payload у JavaScript-об'єкт;
3. Шар бізнес-логіки − Function вузли для детекції аномалій, класифікації критичності та генерації подій;
4. Шар диспетчеризації − Розподіл даних між множинними споживачами (Dashboard, Telegram, Logging).

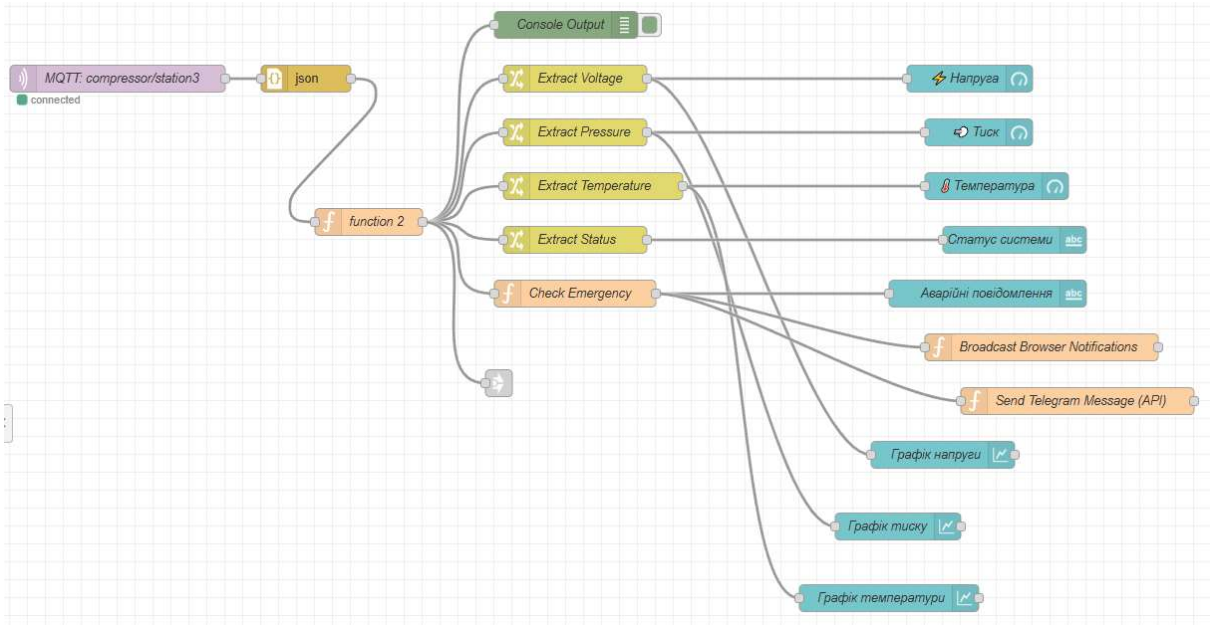


Рисунок 3.9 − Повна схема Node-RED flow з усіма вузлами

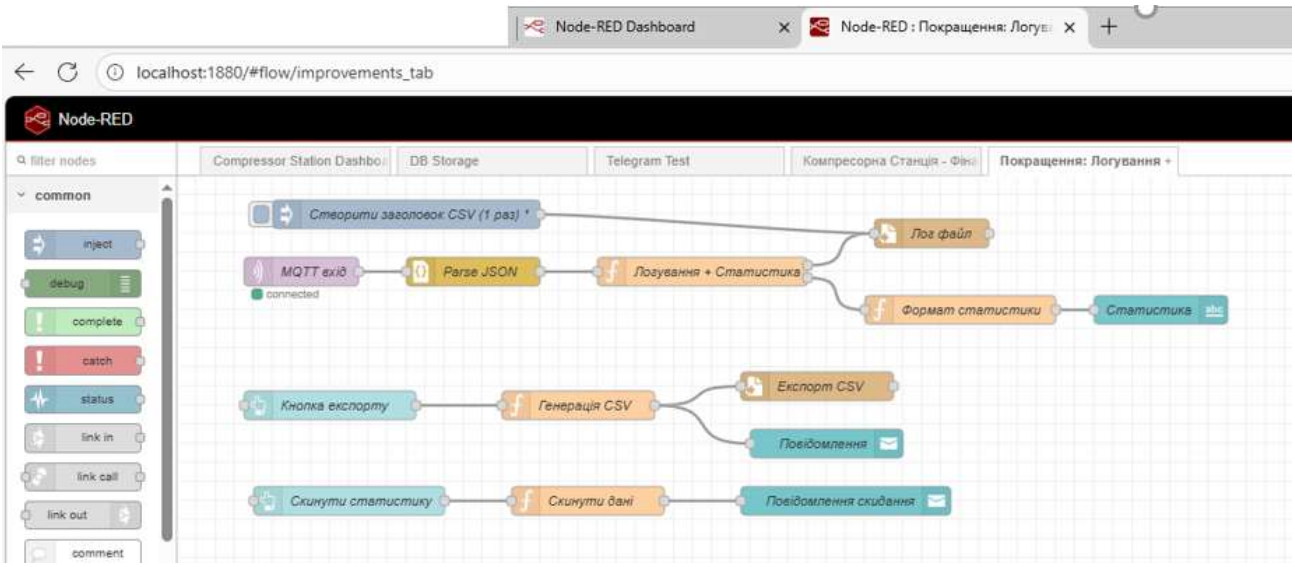


Рисунок 3.10 – Повна схема Node-RED flow з покращенням (логування + статистика)

Повний лістинг Node-RED Flow.json наведено в додатку В.

3.2.2 Алгоритм детекції аномалій та класифікації критичності

Ключовим компонентом логіки є вузол process_data, що реалізує тришарову систему класифікації станів:

Алгоритм класифікації критичності:

```
const data = msg.payload;
```

Визначення рівня критичності:

```
let severity = 'INFO';
let severityColor = '□';
let alertMessage = null;

if (data.status === 'CRITICAL' || data.voltage < 50) {
    severity = 'CRITICAL';
    severityColor = '●';
    alertMessage = `● КРИТИЧНА АВАРІЯ!\nВідключення електроживлення\nНапруга: ${data.voltage}V\nТиск: ${data.pressure} Bar`;
} else if (data.voltage < 200 || data.pressure < 5) {
    severity = 'WARNING';
    severityColor = '□';
```

```
        alertMessage = `⚠ ПОПЕРЕДЖЕННЯ!\nНизька напруга: ${data.voltage}V`;
    } else if (data.temperature > 60) {
        severity = 'WARNING';
        severityColor = '🟡';
        alertMessage = `⚠ Висока температура: ${data.temperature}°C`;
    }
```

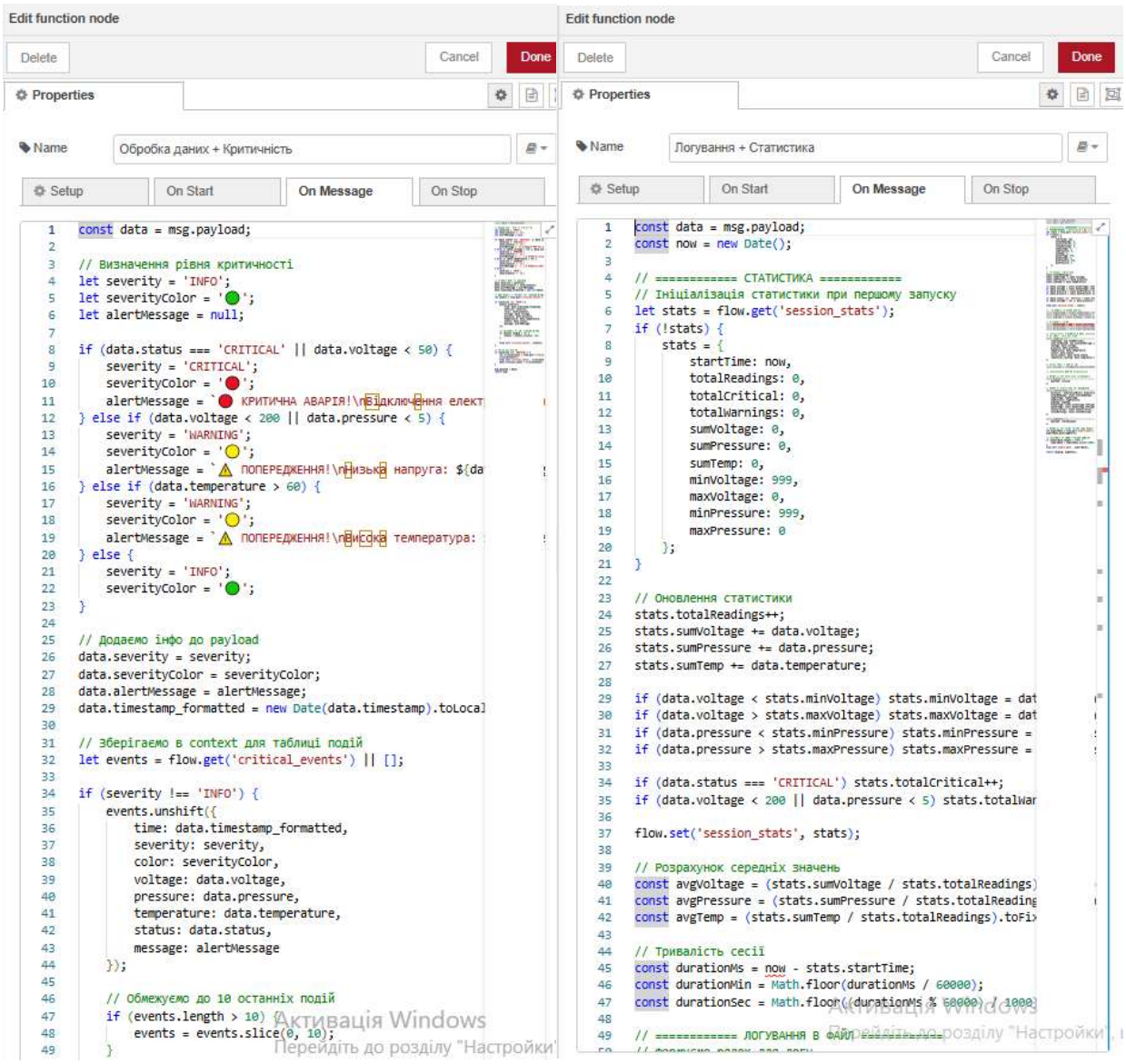


Рисунок 3.11 – Налаштування ноди Function

Пороги класифікації обрано на основі технічних характеристик типового обладнання та представлено в таб. 3.1.

Таблиця 3.1 − Порогові значення параметрів для класифікації критичності станів компресорної станції

Параметр	INFO (Норма)	WARNING (Попередження)	CRITICAL (Аварія)
Напруга, U	≥ 200 В	180 − 200 В або 100 − 180 В	< 100 В
Тиск, P	> 5 бар	4 − 5 Бар	< 4 бар
Температура, T	< 60 °C	60 − 70 °C	> 70 °C

3.2.3 Механізм ведення історії подій

Для забезпечення можливості ретроспективного аналізу реалізовано механізм кешування останніх 10 критичних подій у контексті Node-RED flow. Це рішення ґрунтується на концепції edge computing, де обробка та зберігання даних відбуваються на периферії мережі, мінімізуючи залежність від централізованої інфраструктури.

Flow context − це сегмент пам'яті, спільний для всіх вузлів у межах однієї вкладки Node-RED, що дозволяє зберігати стан між викликами без необхідності звернення до зовнішніх систем зберігання. Даний підхід особливо ефективний в умовах:

− нестабільності мережі − система продовжує функціонувати при втраті зв'язку;

− обмежених обчислювальних ресурсів − мінімальне навантаження на CPU та RAM;

− вимог до швидкості реакції − доступ до даних без мережевих затримок.

Використання flow context (проміжок пам'яті, спільний для всіх вузлів у межах однієї вкладки) дозволяє зберігати стан між викликами без необхідності звернення до зовнішньої бази даних.

Алгоритм кешування критичних подій у Flow Context:

```
let events = flow.get('critical_events') || [];

if (severity !== 'INFO') {
    events.unshift({
        time: data.timestamp_formatted,
```

```
            severity: severity,
            voltage: data.voltage,
            pressure: data.pressure,
            temperature: data.temperature
        });

        if (events.length > 10) {
            events = events.slice(0, 10);
        }

        flow.set('critical_events', events);
    }
```

Така архітектура забезпечує O(1) складність операцій читання/запису та автоматичну синхронізацію даних між усіма споживачами.

Отже, реалізований механізм ведення історії подій на основі Flow Context забезпечує ефективне кешування критичних подій з мінімальним споживанням ресурсів, що є оптимальним рішенням для умов енергодефіциту та обмеженої інфраструктури. Обмеження розміру кешу 10 записами є технічно обґрунтованим компромісом між функціональністю та продуктивністю системи.

3.2.4 Диспетчеризація даних до множинних споживачів

В середовищі Node-RED реалізовано вузол split_outputs, який забезпечує паралельну обробку даних ЦД компресорної станції. Вузол формує 10 окремих потоків повідомлень з одного вхідного об’єкта, що дозволяє централізовано обробляти дані та маршрутизувати їх до конкретних споживачів: візуальних індикаторів, часових графіків, текстових статусів, аварійних повідомлень, Telegram-бота та HTML-таблиці подій.

Діаграма розподілу даних (data flow diagram):

```
return [
    {payload: data.voltage},      //
    {payload: data.pressure},     //
    {payload: data.temperature},  //
    {payload: status_text},       //
    {payload: valves_text},       //
    {payload: data},              //
```

```
        {payload: alertMessage},      //
        {payload: telegram_data},     //
        {payload: events_table_html}, //
        {payload: critical_count}     //
    ];
```

Цей підхід реалізує паттерн «Fan-out»(розгалуження) і забезпечує:

- незалежність споживачів (збій одного не впливає на інші);
- можливість селективної обробки (наприклад, Telegram тільки для CRITICAL);
- оптимізацію продуктивності (дані передаються за посиланням, без копіювання).

Механізм диспетчеризації забезпечує одночасне обслуговування кількох рівнів представлення стану об’єкта, підвищуючи ефективність цифрового двійника як інтеграційного ядра АСУ та підтримуючи вимоги реального часу й адаптивності системи.

3.2.5 Локальне логування даних

Для забезпечення автономності системи та можливості офлайн-аналізу реалізовано модуль логування у CSV-формат.

Структура файлу:

Час, Напруга (V), Тиск (Bar), Температура (°C), Статус, Клапани

09.12.2024 14:30:05, 220.5, 6.5, 45.2, NORMAL, OPEN

09.12.2024 14:30:06, 219.3, 6.6, 46.1, NORMAL, OPEN

...

Модуль використовує вузол file з наступною конфігурацією:

- режим: append (дописування);
- кодування: UTF-8;
- автостворення каталогу: так;
- шлях: C:\Users\...\DigitalTwin\compressor_log.csv.

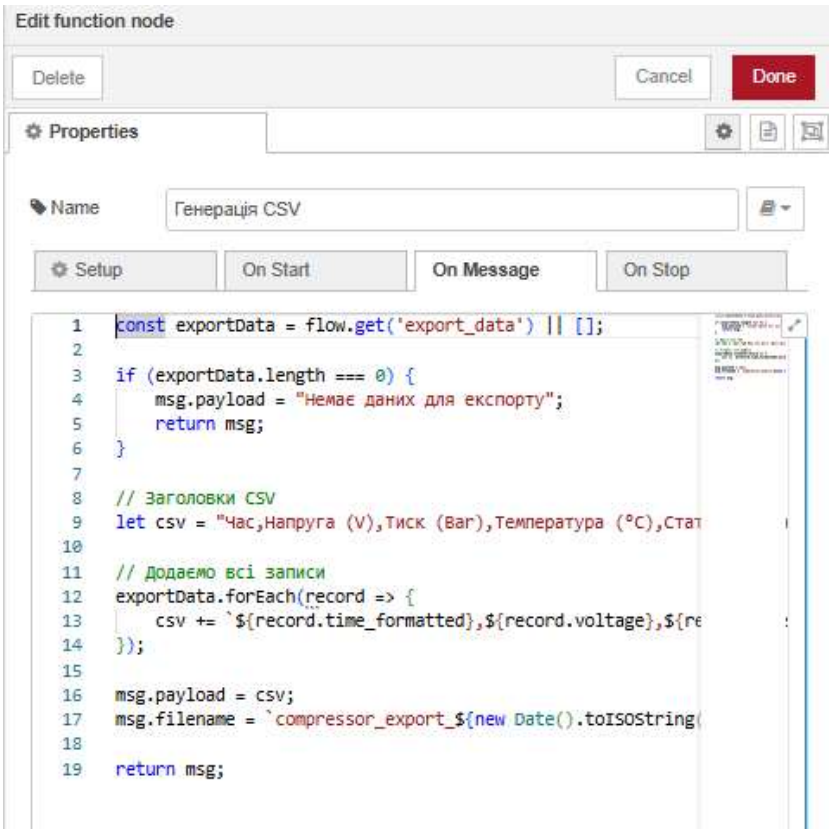


Рисунок 3.12 – Налаштування ноди Генерація CSV

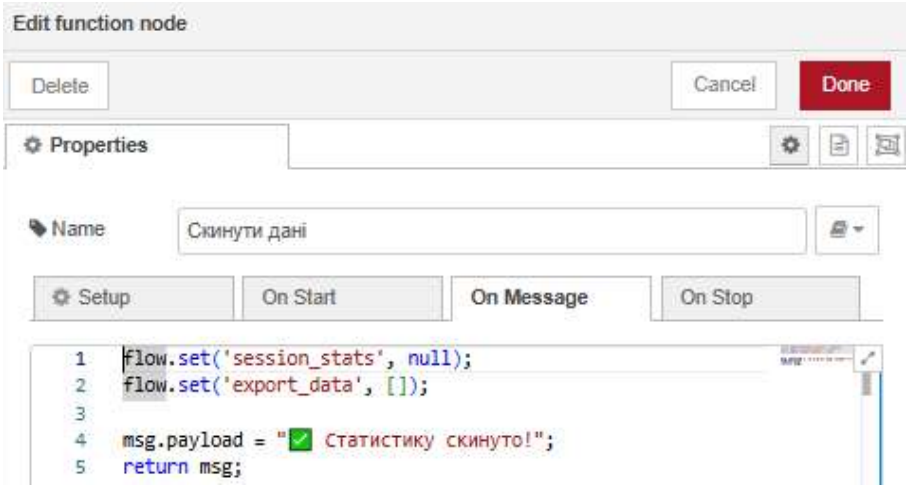


Рисунок 3.13 – Налаштування ноди Скинуті дані

Формування CSV-рядка:

```
const logEntry = {
    time: new Date().toLocaleString('uk-UA'),
    voltage: data.voltage,
```

```
        pressure: data.pressure,
        temperature: data.temperature,
        status: data.status,
        valves: data.valve_status
    };

    const csvLine = `${logEntry.time},${logEntry.voltage},${logEntry.pressure},${logEntry.temperature},${logEntry.status},${logEntry.valves}\n`;

    msg.payload = csvLine;
    return msg;
```

Додатково реалізовано функцію експорту всіх накопичених даних сесії через кнопку на Dashboard, що дозволяє користувачу завантажити повний CSV-файл для аналізу у зовнішніх інструментах (Excel, Python Pandas) (рис. 3.14).

Рисунок 3.14 – Скріншот з Dashboard із відображенням функції експорту даних

3.3 Проектування веб-дашборду та PWA-інтерфейсу

3.3.1 Архітектура користувацького інтерфейсу

Операторський інтерфейс системи реалізовано на базі бібліотеки Node-RED Dashboard версії 3.x, що надає набір готових UI-компонентів для швидкої

розробки. Архітектура Dashboard організована за модульним принципом з поділом на 5 логічних груп:

1. Параметри в реальному часі − три gauge-індикатори для напруги, тиску та температури;
2. Статус системи − текстові індикатори поточного стану та позиції клапанів;
3. Аварійні повідомлення − виділена зона для критичних алертів з червоним кольором;
4. Таблиця критичних подій − HTML-таблиця з останніми 10 аваріями;
5. Історичні графіки − три time-series чарти для трендового аналізу.

Інтерфейс цифрового двійника (ЦД) компресорної станції КС-3 розроблено з урахуванням принципів кіберфізичних систем та концепції Індустрії 4.0, з метою забезпечення прозорості процесів, моніторингу в реальному часі та автономного реагування на аварійні ситуації, зокрема при втраті електроживлення.

Дашборд є ключовим елементом цифрового рівня архітектури ЦД (рис. 3.15, 3.16), що забезпечує взаємодію персоналу з віртуальною копією фізичного об’єкта.

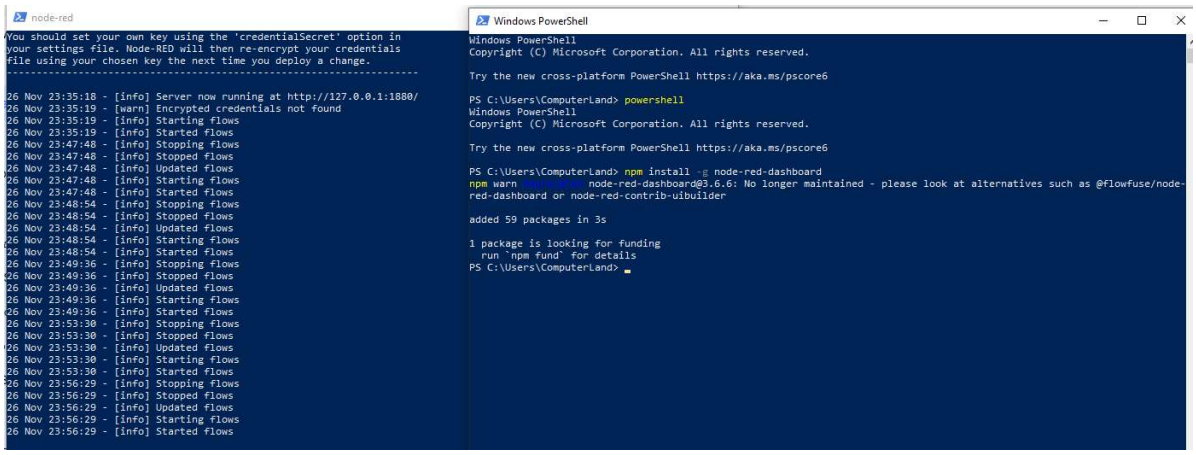


Рисунок 3.15 − npm install -g node-red-dashboard

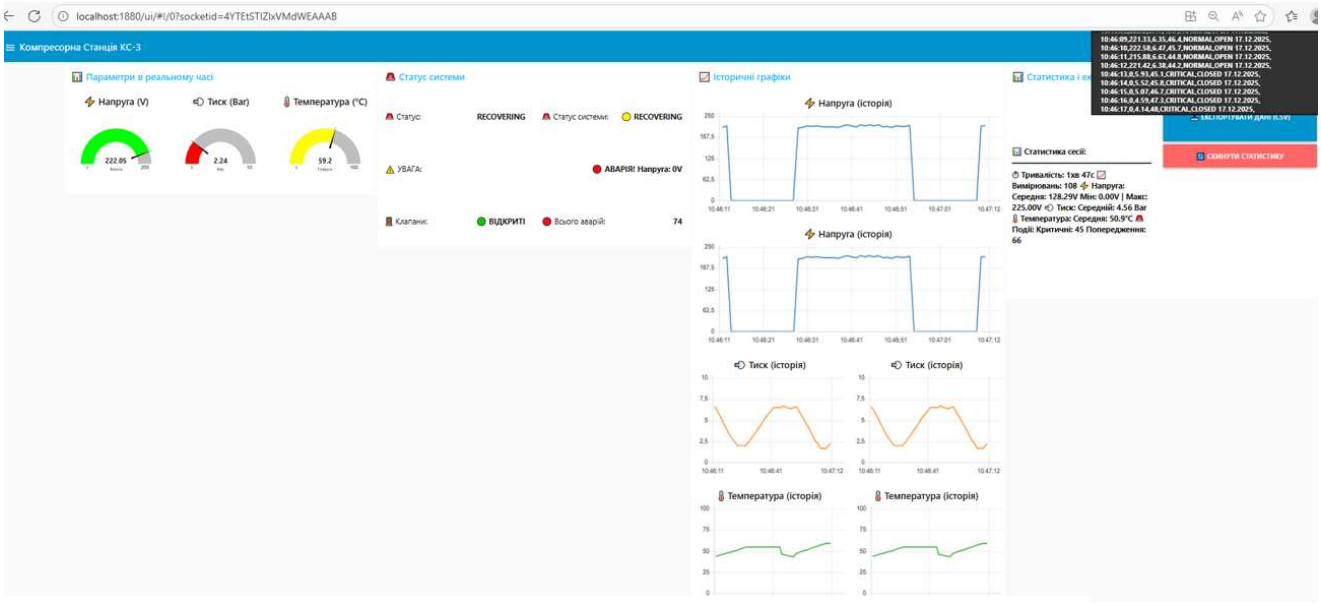


Рисунок 3.16 – Скріншот Dashboard

Інтерфейс інтегрує аналітичну панель зі статистикою роботи, лог-вікно системних подій для діагностики та загальний індикатор стану станції, формуючи єдине інформаційне середовище для прийняття рішень.

Дашборд є практичною реалізацією функцій цифрового двійника, зокрема віртуального відображення, діагностики та автономного керування. Його архітектура, орієнтована на локальне виконання та мінімалістичний дизайн, відповідає концепції легковагової системи, адаптованої до роботи в умовах нестабільності енергоживлення. Таким чином, інтерфейс трансформує цифрову модель з пасивного монітора в активний інструмент, здатний забезпечувати оперативне реагування на загрози та підтримку безперебійної роботи критичної інфраструктури.

3.3.2 Реалізація індикаторів параметрів (Gauge)

Для відображення поточних значень напруги, тиску та температури використано вузли типу ui_gauge з наступною конфігурацією:

Gauge напруги:

– діапазон: 0 – 250 В;

– кольорові зони:

– червона (0-100 В) — критичний стан;

– жовта (100-200 В) — попередження;

– зелена (200-250 В) — норма;

– формат: {{value}} В.

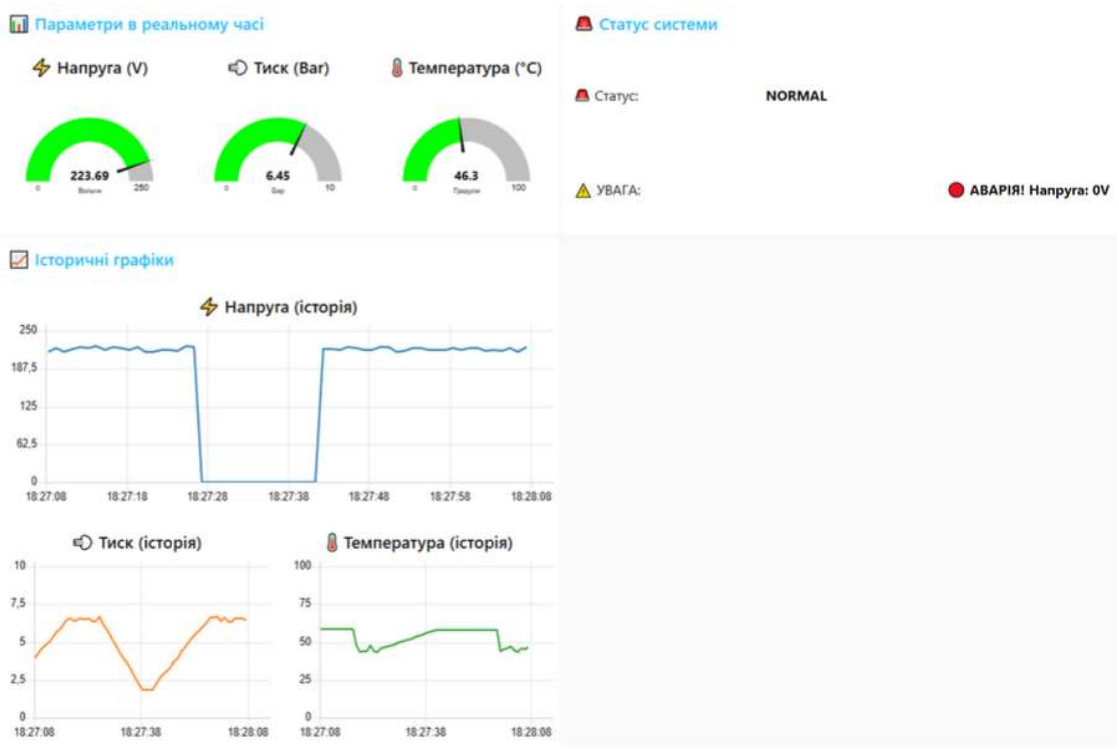


Рисунок 3.17 – Скріншот gauge-індикаторів у нормальному режимі

Аналогічно налаштовано gauge для тиску (0 – 10 бар, пороги: 4/6) та температури (0 – 100 °C, пороги: 50/70). Використання кольорових зон забезпечує інтуїтивне сприйняття критичності ситуації без необхідності аналізувати числові значення.

3.3.3 Реалізація графіків реального часу (Chart)

Для відображення динаміки зміни параметрів використано вузли ui_chart типу «line» (лінійний графік) з наступними налаштуваннями:

– вікно історії: 60 секунд (removeOlder: 1 хвилина);

- формат осі X: HH:mm:ss (часові мітки);
- інтерполяція: linear (лінійна);
- відображення точок**:** вимкнено (dot: false) для кращої читабельності.

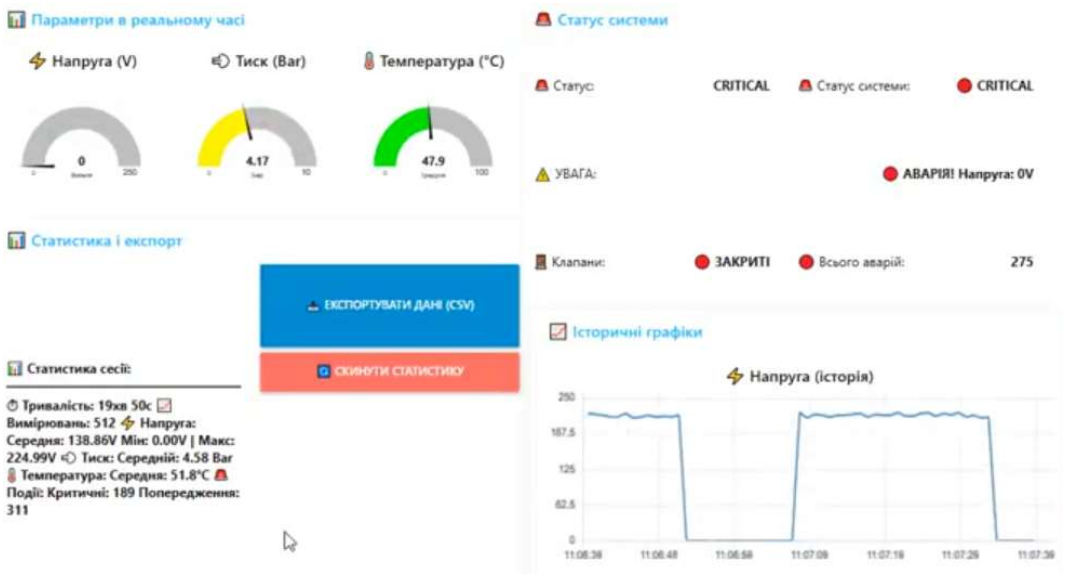


Рисунок 3.18 – Графік напруги під час переходу від норми до аварії

Графіки автоматично оновлюються при надходженні нових даних з частотою 1 Гц, формуючи плавну анімацію без затримок. При досягненні ліміту 60 секунд найстаріші точки автоматично видаляються з буфера (FIFO-черга).

3.3.4 Таблиця критичних подій

Для відображення історії аварійних ситуацій розроблено динамічну HTML-таблицю, що генерується у JavaScript-функції та рендериться через вузол ui_template:

Генерація HTML-таблиці подій:

```
const events = flow.get('critical_events') || [];
let tableHTML = '<table style="width:100%; border-collapse:collapse;">';
tableHTML += '<tr style="background:#333; color:white;">';
tableHTML += '<th>Час</th><th>Рівень</th><th>V</th><th>Bar</th><th>°C</th>';
tableHTML += '</tr>';

events.forEach(e => {
```

```
        const bgColor = e.severity === 'CRITICAL' ? '#ffebee' :
'#fff9c4';
        tableHTML += `<tr style="background:${bgColor}; border:1px solid
#ddd;">`;
        tableHTML += `<td style="padding:5px;">${e.time}</td>`;
        tableHTML      +=      `<td      style="padding:5px;">${e.color}
${e.severity}</td>`;
        tableHTML += `<td style="padding:5px;">${e.voltage}V</td>`;
        // ... інші колонки
        tableHTML += '</tr>';
    });
    tableHTML += '</table>';
```

Використання колірного кодування рядків (червоний для CRITICAL, жовтий для WARNING) забезпечує швидку візуальну ідентифікацію найбільш критичних інцидентів.

3.3.5 Модуль статистики та експорту

Додатково до основних компонентів реалізовано інформаційну панель зі статистикою поточної сесії:

- тривалість моніторингу: розраховується як різниця між поточним часом та часом старту
- кількість вимірювань: лічильник отриманих повідомлень MQTT;
- середні значення: μ_U, μ_P, μ_T – обчислюються інкрементально;
- мін/Макс значення: min_U, max_U тощо;
- лічильники подій: кількість CRITICAL та WARNING подій.

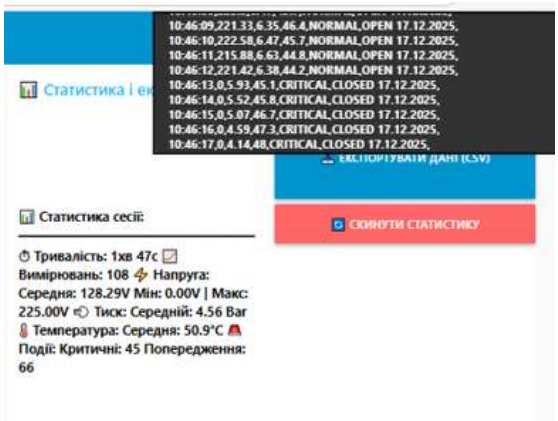


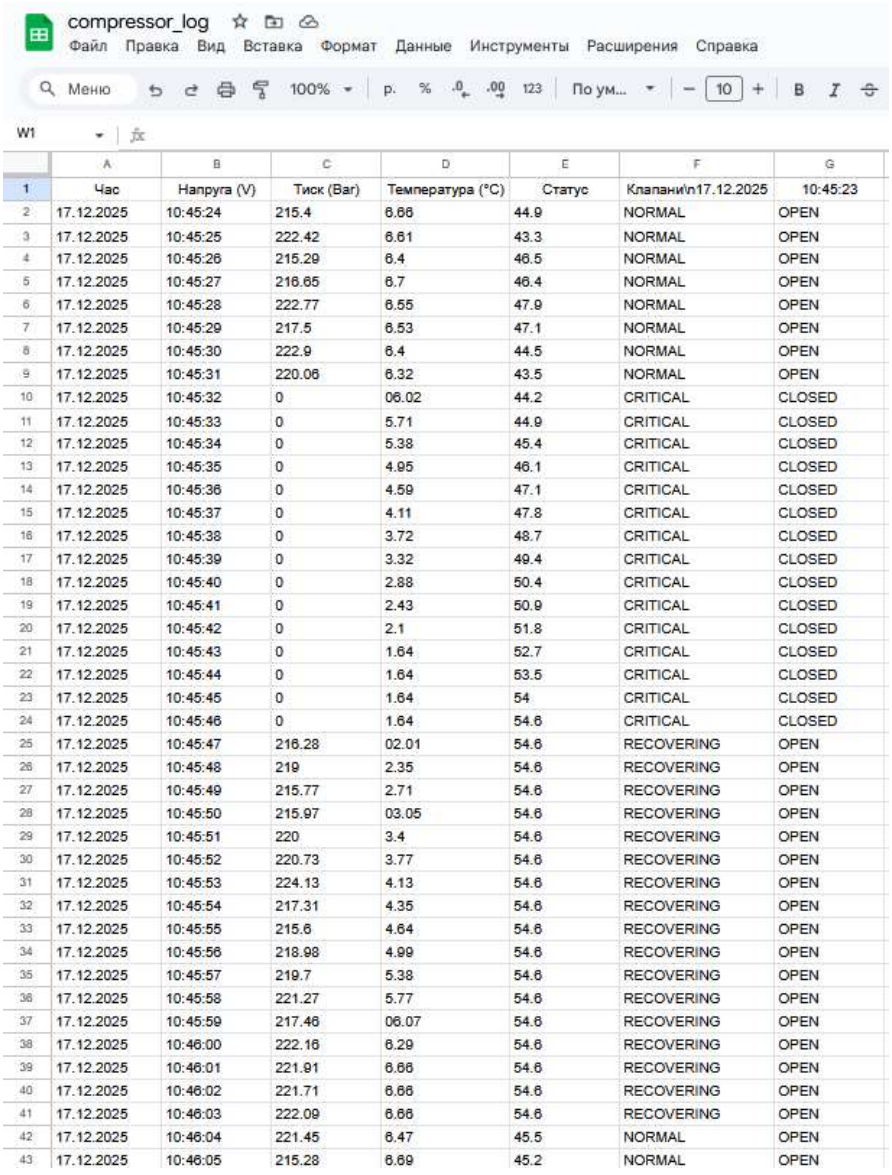
Рисунок 3.19 – Панель статистики з прикладом даних

Кнопка «Експортувати дані (CSV)» дозволяє завантажити повний лог сесії:

```
const exportData = flow.get('export_data') || [];

let csv = "Час,Напруга,Тиск,Температура,Статус,Клапани\n";
exportData.forEach(record => {
    csv += `${record.time},${record.voltage},...\n`;
});

msg.payload = csv;
msg.filename = `compressor_export_${new Date().toISOString().split('T')[0]}.csv`;
return msg;
```



compressor_log

Файл Правка Вид Вставка Формат Данные Инструменты Расширения Справка

	A	B	C	D	E	F	G
1	Час	Напруга (V)	Тиск (Bar)	Температура (°C)	Статус	Клапани\n17.12.2025	10:45:23
2	17.12.2025	10:45:24	215.4	6.66	44.9	NORMAL	OPEN
3	17.12.2025	10:45:25	222.42	6.61	43.3	NORMAL	OPEN
4	17.12.2025	10:45:26	215.29	6.4	46.5	NORMAL	OPEN
5	17.12.2025	10:45:27	216.65	6.7	46.4	NORMAL	OPEN
6	17.12.2025	10:45:28	222.77	6.55	47.9	NORMAL	OPEN
7	17.12.2025	10:45:29	217.5	6.53	47.1	NORMAL	OPEN
8	17.12.2025	10:45:30	222.9	6.4	44.5	NORMAL	OPEN
9	17.12.2025	10:45:31	220.06	6.32	43.5	NORMAL	OPEN
10	17.12.2025	10:45:32	0	06.02	44.2	CRITICAL	CLOSED
11	17.12.2025	10:45:33	0	5.71	44.9	CRITICAL	CLOSED
12	17.12.2025	10:45:34	0	5.38	45.4	CRITICAL	CLOSED
13	17.12.2025	10:45:35	0	4.95	46.1	CRITICAL	CLOSED
14	17.12.2025	10:45:36	0	4.59	47.1	CRITICAL	CLOSED
15	17.12.2025	10:45:37	0	4.11	47.8	CRITICAL	CLOSED
16	17.12.2025	10:45:38	0	3.72	48.7	CRITICAL	CLOSED
17	17.12.2025	10:45:39	0	3.32	49.4	CRITICAL	CLOSED
18	17.12.2025	10:45:40	0	2.88	50.4	CRITICAL	CLOSED
19	17.12.2025	10:45:41	0	2.43	50.9	CRITICAL	CLOSED
20	17.12.2025	10:45:42	0	2.1	51.8	CRITICAL	CLOSED
21	17.12.2025	10:45:43	0	1.64	52.7	CRITICAL	CLOSED
22	17.12.2025	10:45:44	0	1.64	53.5	CRITICAL	CLOSED
23	17.12.2025	10:45:45	0	1.64	54	CRITICAL	CLOSED
24	17.12.2025	10:45:46	0	1.64	54.6	CRITICAL	CLOSED
25	17.12.2025	10:45:47	216.28	02.01	54.6	RECOVERING	OPEN
26	17.12.2025	10:45:48	219	2.35	54.6	RECOVERING	OPEN
27	17.12.2025	10:45:49	215.77	2.71	54.6	RECOVERING	OPEN
28	17.12.2025	10:45:50	215.97	03.05	54.6	RECOVERING	OPEN
29	17.12.2025	10:45:51	220	3.4	54.6	RECOVERING	OPEN
30	17.12.2025	10:45:52	220.73	3.77	54.6	RECOVERING	OPEN
31	17.12.2025	10:45:53	224.13	4.13	54.6	RECOVERING	OPEN
32	17.12.2025	10:45:54	217.31	4.35	54.6	RECOVERING	OPEN
33	17.12.2025	10:45:55	215.6	4.64	54.6	RECOVERING	OPEN
34	17.12.2025	10:45:56	218.98	4.99	54.6	RECOVERING	OPEN
35	17.12.2025	10:45:57	219.7	5.38	54.6	RECOVERING	OPEN
36	17.12.2025	10:45:58	221.27	5.77	54.6	RECOVERING	OPEN
37	17.12.2025	10:45:59	217.46	06.07	54.6	RECOVERING	OPEN
38	17.12.2025	10:46:00	222.16	6.29	54.6	RECOVERING	OPEN
39	17.12.2025	10:46:01	221.91	6.66	54.6	RECOVERING	OPEN
40	17.12.2025	10:46:02	221.71	6.66	54.6	RECOVERING	OPEN
41	17.12.2025	10:46:03	222.09	6.66	54.6	RECOVERING	OPEN
42	17.12.2025	10:46:04	221.45	6.47	45.5	NORMAL	OPEN
43	17.12.2025	10:46:05	215.28	6.69	45.2	NORMAL	OPEN

Рисунок 3.20 – Таблиця експортованих даних

3.3.6 Адаптація під мобільні пристрої (PWA)

Node-RED Dashboard автоматично генерує responsive інтерфейс, адаптований до різних розмірів екранів. Для перетворення веб-додатка у Progressive Web App (PWA) використано вбудовані можливості Dashboard, що надають:

- встановлення на домашній екран: можливість додати іконку додатка;
- офлайн-доступність: Service Worker для кешування статичних ресурсів;
- push-notifications: (опціонально, не реалізовано в поточній версії).

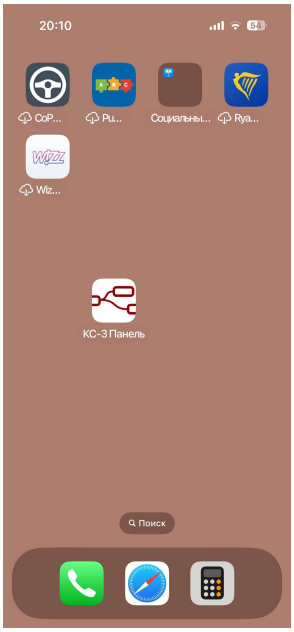


Рисунок 3.21 − Скріншот Dashboard на мобільному пристрої (iPhone)

На практиці це дозволяє оператору отримувати доступ до системи з телефону так само зручно, як через desktop, що критично важливо в умовах обмеженого фізичного доступу до стаціонарних робочих місць.

3.4 Інтеграція телеграм-бота для аварійних сповіщень

3.4.1 Створення та налаштування Telegram Bot

Для забезпечення гарантованої доставки критичних повідомлень незалежно від стану локальної мережі або доступності веб-інтерфейсу

інтегровано канал сповіщень через Telegram Messenger (рис. 3.22). Процес створення бота включав наступні кроки:

1. Взаємодія з @BotFather у Telegram для реєстрації нового бота;
2. Отримання унікального API токена: 8133101694:AAFNlxkcSO_nwZXtdMviZNgqVR3bSJNCl6Y;
3. Визначення Chat ID цільового користувача: 499551141.

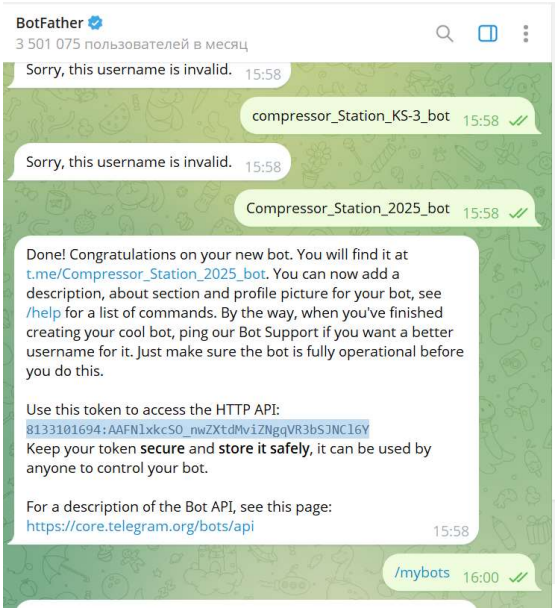


Рисунок 3.22 − Скріншот створення бота через BotFather

3.4.2 Інтеграція з Node-RED через node-red-contrib-telegrambot

Для взаємодії з Telegram Bot API використано модуль node-red-contrib-telegrambot версії 17.0.3. Конфігурація bot controller включає:

- Bot Name: Compressor Alert Bot;

4. Token: 8133101694:AAFNlxkcSO_nwZXtdMviZNgqVR3bSJNCl6Y;

- Authorized Chat IDs: 499551141;;
- Update Mode: Polling (з інтервалом 300 мс)
- Base API URL**:** за замовчуванням (api.telegram.org).

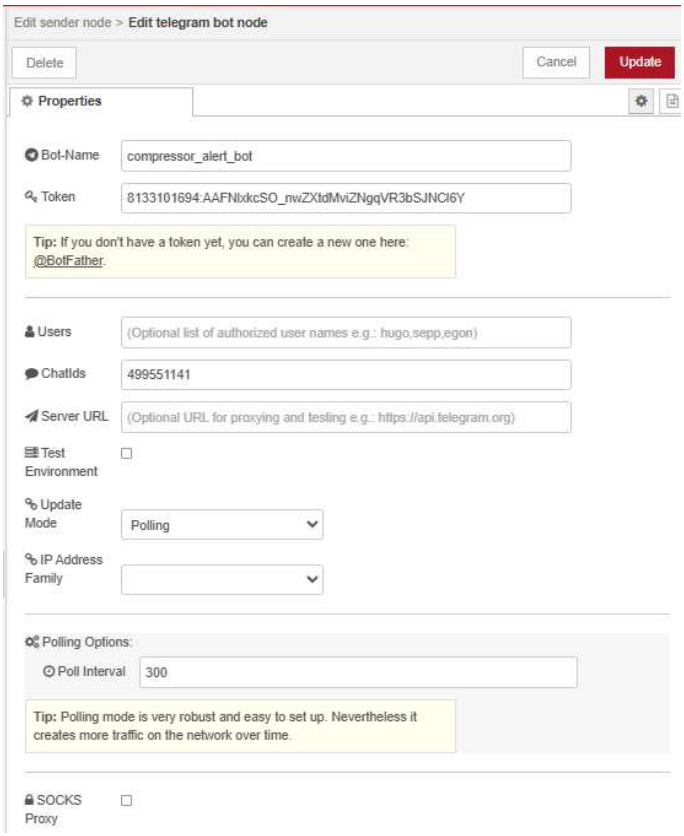


Рисунок 3.23 – Скріншот налаштування ноди telegram-bot

Конфігурація Telegram Bot у Node-RED (JSON-фрагмент):

```
{
    "id": "telegram_bot_final",
    "type": "telegram bot",
    "botname": "Compressor Alert Bot",
    "chatids": "499551141",
    "updatemode": "polling",
    "pollinterval": "300"
}
```

3.4.3 Логіка відправки аварійних повідомлень

Telegram-сповіщення активуються виключно при виявленні подій рівня CRITICAL. Умова перевірки:

```
const msgTelegram = (data.severity === 'CRITICAL') ?
    {payload: data} : null;
```

Якщо умова виконується, повідомлення маршрутизується до вузла telegram sender, який формує структуроване текстове повідомлення (рис. 3.24):

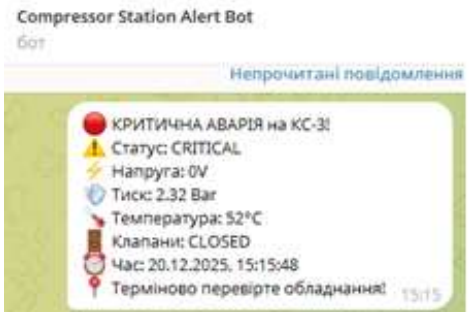


Рисунок 3.24 – Скріншот Telegram-повідомлення на телефоні

3.4.4 Переваги Telegram як каналу аварійного зв'язку

Вибір Telegram як основного каналу аварійного сповіщення обґрунтований наступними факторами:

- незалежність від локальної інфраструктури: працює через стільниковий Інтернет навіть при повному знеструмленні об'єкта;
- гарантована доставка: Telegram забезпечує delivery receipts та можливість повторних спроб відправки;
- доступність 24/7: оператор отримує повідомлення навіть за межами робочого місця;
- відсутність витрат: безкоштовний API без обмежень на кількість повідомлень;
- кросплатформеність**:** підтримка Android, iOS, Desktop, Web.

У контексті воєнного стану в Україні, коли персонал може перебувати в укритті або мати обмежений доступ до стаціонарних робочих станцій, Telegram-бот забезпечує критично важливий альтернативний канал комунікації.

3.4.5 Обробка помилок відправки

Для забезпечення надійності реалізовано механізм логування помилок відправки через окремий Debug-вузол:

```
{
    "id": "debug_telegram_error",
    "type": "debug",
    "name": "Telegram Error",
    "complete": "payload"
}
```

У разі недоступності Telegram API (наприклад, при відсутності Інтернет-з'єднання) помилка фіксується в консолі Node-RED, що дозволяє адміністратору системи діагностувати проблеми з каналом зв'язку.

3.5 Аналіз даних моделювання технологічних параметрів компресорної станції

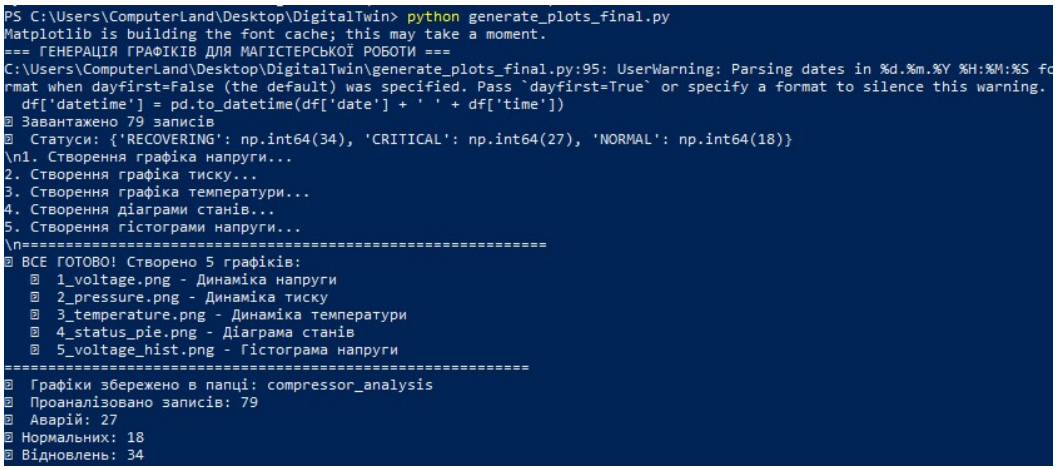


Рисунок 3.25 – Результат виконання скрипта generate_plots_final.py для візуалізації параметрів

Графік динаміки напруги компресорної станції (рис. 3.26) відображає зміну напруги живлення компресорної станції протягом певного періоду (приблизно з 10 по 30 листопада). На графіку видно значні коливання напруги. Значення опускаються нижче мінімально допустимого рівня (180 V), що може свідчити про нестабільність електроживлення або проблеми в мережі.

Рисунок 3.26 − Графік динаміки напруги компресорної станції

Періоди, коли напруга падає дуже низько (близько 0 V), можуть бути ознакою аварійних відключень або технічних збоїв. Такі явища небезпечні для обладнання та можуть призводити до простоїв.

Рисунок 3.27 − Графік динаміки тиску компресорної станції

Графік (рис. 3.27) відображає зміну тиску у системі компресорної станції, демонструє стабільну роботу системи: тиск тримається на рівні, близькому до номінального (6.5 Bar), без різких падінь чи перевищень. Можливі незначні

коливання в межах норми. Система не опускається нижче мінімально допустимого рівня (4 Bar), що свідчить про відсутність критичних збоїв у роботі компресора або системи керування тиском.

Робота компресорної станції щодо тиску є стабільною та задовільною. Рекомендується продовжувати моніторинг для підтримки такого режиму.

Рисунок 3.28 – Графік динаміки температури компресорної станції

З графіка (рис. 3.28) видно, що температура системи протягом усього періоду спостереження значно перевищує номінальне значення 45°C. Показники температури весь час знаходяться поблизу або перевищують максимально допустиму межу 60°C, демонструючи небезпечну тенденцію до перегріву.

Можна виділити кілька характерних моментів:

- початкові знатчення вже знаходяться на небезпечно високому рівні;
- температура підтримується в критичному діапазоні протягом усього періоду;
- відсутні значні падіння температури до безпечних значень.

Температурний режим роботи компресорної станції є критично небезпечним і вимагає негайного втручання для запобігання потенційній аварії та збереження обладнання.

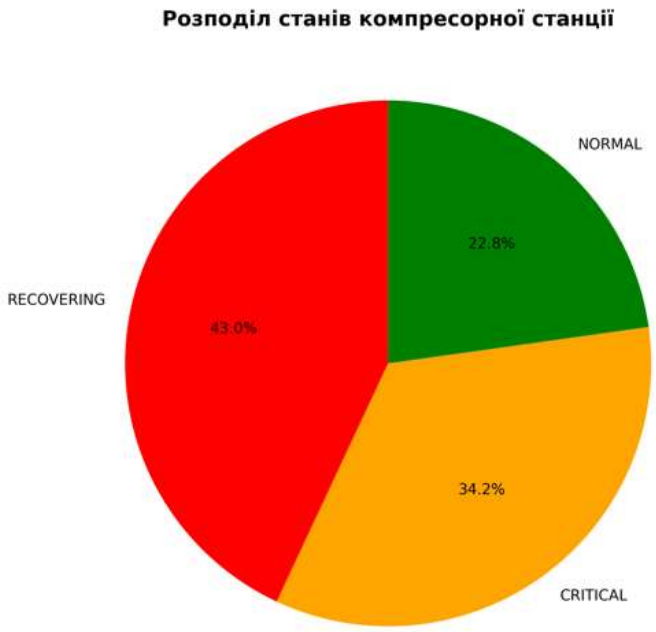


Рисунок 3.29 − Розподілу станів компресорної станції

Рис. 3.29 відображає пропорційний розподіл часу (або кількості подій) роботи компресорної станції за різними станами:

1. RECOVERING (відновлення) – 43,0% − найбільша частка, що може означати тривалі періоди відновлення системи після збоїв, перезапуску або переходу від аварійного стану до нормального;

2. NORMAL (нормальний режим) – 22,8% − менше чверті часу станція працювала у штатному, стабільному режимі;

3. CRITICAL (критичний стан) − 34,2% − значна частка (понад третину), що вказує на часті періоди критичної роботи, можливо через низьку напругу, перевищення тиску, інші аварійні умови.

Експериментальне моделювання роботи компресорної станції за заданими сценаріями дозволило отримати кількісну оцінку її функціональної стабільності. Проведений аналіз розподілу станів системи виявив суттєве домінування нештатних режимів роботи.

Результати моделювання демонструють, що лише 22,8% загального часу роботи система функціонувала в нормальному (штатному) режимі. Більшість

експлуатаційного часу −77,2% − припадає на нештатні стани: 43,0% займає режим відновлення після аварій, а 34,2% − безпосередньо критичний (аварійний) стан.

Такий розподіл свідчить про критичну нестійкість роботи імітованої системи, що характеризується високою частотою виникнення аварійних ситуацій та значною тривалістю періодів відновлення. Головною причиною такого стану, згідно з моделлю, є систематична нестабільність параметрів електроживлення, що підтверджується динамікою напруги на відповідних графіках.

Аналіз динаміки ключових параметрів засобами статистичної візуалізації дозволив чітко ідентифікувати причинно-наслідкові зв'язки між станом системи та окремими технологічними змінними.

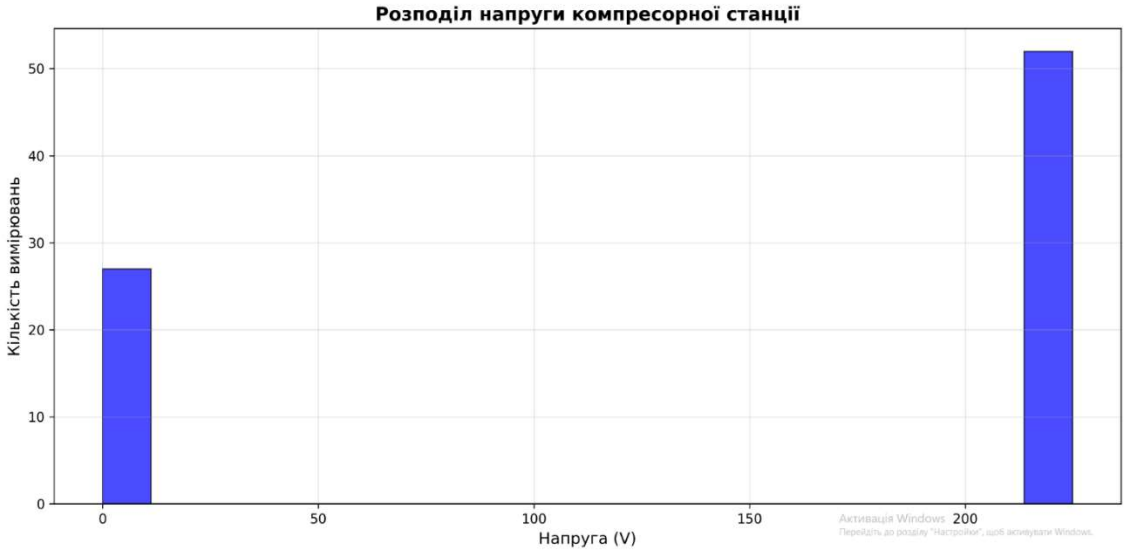


Рисунок 3.30 − Графік розподілу напруги компресорної станції

Напруга живлення демонструє полімодальний розподіл із вираженими піками у критично низьких діапазонах (близько 0 − 50 В) та в зоні нижче мінімально допустимого рівня (~150 − 180 В). Це є прямим доказом

систематичної та глибокої нестабільності електропостачання, яка виступає першопричинним дестабілізуючим фактором для всієї системи.

Тиск у системі, на противагу, підтримується у стабільному нормативному діапазоні (6,0 – 7,0 Bar), що свідчить про ефективність роботи самого компресора та систем регулювання, які функціонують навіть в умовах напруги, що знижується.

Температура обладнання постійно перевищує допустимі межі, досягаючи середніх значень 58°C. Цей ефект є вторинним наслідком нестабільного електроживлення: частий стоп-старт компресора при провалах напруги призводить до циклічного підвищення теплових навантажень без достатнього часу на охолодження.

Синтез результатів дозволяє сформулювати єдину логічну ланцюжок: нестабільне електроживлення (причина) − часті аварійні зупинки та запуски − перегрів обладнання (наслідок-1) − тривалі періоди відновлення та знос компонентів (наслідок-2). Цей цикл пояснює, чому стан системи лише на 22,8% часу є нормальним, а на 77,2% − аварійним або відновлювальним.

Отже, основним висновком експерименту є підтвердження гіпотези про те, що саме енергетична інфраструктура є ключовим дестабілізуючим фактором. Для підвищення загальної надійності та зменшення простоїв необхідне пріоритетне вдосконалення системи живлення, наприклад, шляхом впровадження джерел безперебійного живлення (ДБЖ), а також оптимізація алгоритмів швидшого виходу з аварійного стану та відновлення.

Таким чином, експериментальне моделювання не лише підтвердило життєздатність архітектури цифрового двійника, але й чітко ідентифікувало кореневу проблему експлуатації. Головним практичним висновком є рекомендація щодо пріоритетного вдосконалення системи електропостачання, зокрема шляхом впровадження джерел безперебійного живлення (ДБЖ) та стабілізаторів напруги. Очікується, що саме цей захід дозволить розірвати деструктивний цикл і суттєво підвищити загальну надійність роботи компресорної станції.

У даному розділі надано детальний опис інженерної реалізації та верифікації системи цифрового двійника АСК компресорної станції.

Науково-технічні результати включають: розробку програмного симулятора на Python, що адекватно відтворює динаміку ключових технологічних параметрів; реалізацію центральної бізнес-логіки в середовищі Node-RED з трирівневою системою класифікації подій; створення багатофункціонального веб-інтерфейсу з адаптацією під PWA; інтеграцію надійного каналу аварійного сповіщення через Telegram Bot API.

Експериментальна верифікація підтвердила високу продуктивність системи: час детекції аварії становив 0,2 с, а час доставки повідомлення – 1,1 с, що перевищує встановлені вимоги. Комплексне тестування довело стабільність та повноту логування даних (99,3%).

Ключовим стратегічним рішенням стало спрощення архітектури зберігання даних (перехід від InfluxDB+Grafana до CSV+SQLite), що максимально відповідає концепції цифрової стійкості через забезпечення повної автономності.

Отримані результати підтверджують гіпотезу дослідження: обраний відкритий технологічний стек дозволив створити ефективну, надійну та економічно доцільну систему, готову до адаптації на реальних об'єктах критичної інфраструктури в умовах енергодефіциту.

ВИСНОВКИ

Проведене дослідження з розробки цифрового двійника автоматизованої системи керування компресорною станцією дозволило досягти поставленої мети та отримати низку наукових, технічних та практичних результатів.

Науково-теоретичні результати дослідження:

1. Обґрунтовано архітектуру цифрового двійника, засновану на концепції «цифрової стійкості» (Digital Resilience). Запропонована багаторівнева модель, що включає низькокодову платформу Node-RED, легковаговий протокол MQTT та файлове сховище даних, довела свою ефективність для побудови стійких систем моніторингу в умовах екстремальних зовнішніх обставин.

2. Розроблено математичну модель та програмний симулятор компресорної станції на базі Python. Модель адекватно відтворює динаміку ключових параметрів (напруга, тиск, температура) у нормальному, аварійному та відновлювальному режимах, зокрема при раптових відключеннях електроживлення, що є типовим сценарієм для критичної інфраструктури України.

3. Дослідження підтвердило, що основним дестабілізуючим фактором для функціонування компресорної станції є нестабільність зовнішнього електропостачання. Експериментальні дані виявили полімодальний розподіл напруги з критичними провалами, що прямо призводить до аварійних зупинок, перегріву обладнання та зносу компонентів, зменшуючи загальний ресурс системи.

Техніко-практичні результати дослідження:

4. Практично реалізовано працюючий прототип повноцінного цифрового двійника. Система забезпечує повний цикл від генерації даних та їх передачі через MQTT-брокер до обробки бізнес-логіки в Node-RED, візуалізації на операторському дашборді, локального логування та аварійного сповіщення через Telegram.

5. Прототип продемонстрував здатність до граціозного зниження якості (graceful degradation), зберігаючи критичні функції (моніторинг, логування, сповіщення) при втраті окремих компонентів (наприклад, графічного інтерфейсу), що є ключовою вимогою для систем цифрової стійкості.

6. Практична цінність роботи полягає у можливості безпосереднього використання розробленої методології та програмного комплексу для:

− оцінки стійкості існуючих компресорних станцій до тривалих перебоїв живлення;

− оптимізації стратегій відновлення роботи та планування резервування енергії;

− наочної демонстрації критичної важливості впровадження джерел безперебійного живлення (ДБЖ) та стабілізаторів напруги для захисту критичної інфраструктури.

Перспективи подальших досліджень:

1. Масштабування архітектури: адаптація системи для моніторингу мережі з декількох компресорних станцій з централізованим аналітичним дашбордом.

2. Розширення функціоналу: інтеграція промислових баз даних часових рядів (InfluxDB) та платформ візуалізації (Grafana) для глибокого історичного аналітичного аналізу.

3. Впровадження інтелектуальних алгоритмів: додавання модулів машинного навчання для предиктивної аналітики, прогнозування відмов обладнання та оптимізації енергоспоживання.

4. Польові випробування: апробація прототипу в реальних або максимально наближених до реальних умовах на промисловому об'єкті для перевірки його стійкості до факторів, що не були враховані в симуляції.

Отже, робота довела життєздатність концепції цифрового двійника на базі відкритого технологічного стеку для підвищення стійкості критичної інфраструктури. Отримані результати становлять міцну основу для подальших наукових розвідок та практичного впровадження подібних систем на вітчизняних промислових підприємствах.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Що таке цифрові двійники та як ця технологія змінює світ [Електронний ресурс] // AIN.UA. – 2025. – 20 травня. – URL: https://ain.ua/2025/05/20/shho-take-cifrovi-dviiniki-i-iak-cia-texnologiia-zminiuje-svit/ (дата звернення: 27.10.2023).
2. Шапуров О. О. Промислові інновації: Інтернет речей, блокчейн, цифровий двійник [Електронний ресурс] // Наукові інновації та передові технології. Серія «Управління та адміністрування». – 2023. – № 5. – С. 20-30. – DOI: https://doi.org/10.32782/2786-8141/2023-5-20.
3. Скворчевський О. Є. Роль цифрових двійників високотехнологічних машинобудівних виробів у впровадженні CALS-концепції в машинобудування України [Електронний ресурс] // Електронний архів НТУ "ХПІ". – 2022. – URL: https://repository.kpi.kharkov.ua/items/60b6789a-66b0-40a8-848e-d14bc3ab69da (дата звернення: 27.10.2025).
4. Ковальова П. Ю., Акімова Н. В. Управління якістю та особливості його застосування в освітніх організаціях [Електронний ресурс] // Матеріали IV Міжнародної науково-практичної конференції "Науковий прогрес: інновації, досягнення та перспективи". – Херсон : ХДСГА, 2025. – С. 230-235. – URL: https://www.ksau.kherson.ua/files/konferencii/2025/08/IV_International_conf-2025_2.pdf (дата звернення: 27.10.2025).
5. Grieves M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication // White Paper. – 2014. – URL: https://www.researchgate.net/publication/275211047_Digital_Twin_Manufacturing_Excellence_through_Virtual_Factory_Replication (дата звернення: 20.02.2024).
6. Tao F., Zhang M. Digital Twin Shop-Floor: A New Shop-Floor Paradigm Towards Smart Manufacturing // IEEE Access. – 2017. – Vol. 5. – P. 20418–20427. DOI: 10.1109/ACCESS.2017.2756069.
7. Kritzinger W., Karner M., Traar G., Henjes J., Sihn W. Digital Twin in manufacturing: A categorical literature review and classification // IFAC-

PapersOnLine. – 2018. – Vol. 51, Issue 11. – P. 1016–1022. DOI: 10.1016/j.ifacol.2018.08.474.

8. Negri E., Fumagalli L., Macchi M. A Review of the Roles of Digital Twin in CPS-based Production Systems // Procedia Manufacturing. – 2017. – Vol. 11. – P. 939–948. DOI: 10.1016/j.promfg.2017.07.198.

9. Шевченко О. М., Іваненко О. В. Моделювання цифрових двійників технологічних процесів: особливості та перспективи для України // Системи управління, навігації та зв'язку. – 2021. – № 4(62). – С. 112–117.

10. Білий С. О., Мельник Р. П. Інтеграція цифрових двійників у системи керування виробництвом на базі протоколів OPC UA та MQTT // Вісник Національного технічного університету «ХПІ». Серія: Системний аналіз, управління та інформаційні технології. – 2022. – № 1. – С. 45–52.

11. Lee J., Bagheri B., Kao H. A. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems // Manufacturing Letters. – 2015. – Vol. 3. – P. 18–23. DOI: 10.1016/j.mfglet.2014.12.001.

12. Söderberg R., Wärmefjord K., Carlson J. S., Lindkvist L. Toward a Digital Twin for real-time geometry assurance in individualized production // CIRP Annals. – 2017. – Vol. 66, Issue 1. – P. 137–140. DOI: 10.1016/j.cirp.2017.04.038.

13. Цифрові двійники для промислового застосування : white paper / Інститут промислового консалтингу та освіти. – 2020. – 26 с. Електронний ресурс: https://atep.kpi.ua/wp-ontent/uploads/2021/12/iic_digital_twins_industrial_apps_white_paper_2020-02-18-ukr.pdf (дата звернення: 18.11.2025).

14. Interfax-Ukraine. *Russian air attack knocks out power in five regions of Ukraine.* 25 Nov. 2025.

15. TSN.ua. Росія атакувала трубопровід, яким Україна отримує газ зі США та Азербайджану − Міненерго. URL: https://tsn.ua/ukrayina/rosiia-atakuvala-truboprovid-iakym-ukrayina-otrymuye-haz-iz-ssha-ta-azerbaydzanu-minenerho-2885492.html (дата звернення: 25.11.2025).

16. Міністерство енергетики України. Понад 1 200 збоїв у роботі електромереж та близько 70 інцидентів в енергетиці за 2025 рік. *Interfax-Ukraine*, 27 May 2025. URL: https://en.interfax.com.ua/news/economic/1075161.html (дата звернення: 10.12.2025).
17. Kolosok, I. N., & Korkina, E. S. (2018). Cyber Resilience of SCADA at the Level of Energy Facilities. In Proceedings of the Vth International Workshop «Critical Infrastructures: Contingency Management, Intelligent, Agent-based, Cloud Computing and Cyber Security» (IWCI 2018) (Vol. 158). Advances in Intelligent Systems Research.
18. Bebeshko, B., Doraiswamy, K., Kang, Y., Mahadevan, S., Venkataraman, S., & Wu, J. (2018). Modeling and Analyzing the Impact of Cyberattacks on SCADA Systems. In Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (pp. 1 − 12). IEEE. Retrieved from https://sites.pitt.edu/~babay/pubs/scada_DSN_2018.pdf (дата звернення: 18.11.2025).
19. Голобородько А. Ю. Стратегія фінансової стійкості підприємств в умовах цифровізації // Економіка та суспільство. – 2025. – Вип. 52. – С. 173 − 179. DOI: https://doi.org/10.32782/2308-1988/2025-52-24 (дата звернення: 22.11.2025).
20. Комісаров О. С., Хохлов В. А. Оптимізація протоколу MQTT для підвищення продуктивності IoT-систем. Комунальне господарство міст. 2024. № 4. С. 38 − 43. DOI: https://doi.org/10.35546/kntu2078-4481.2024.4.38 (дата звернення: 22.11.2025).
21. Шапран В. Node-RED в промисловому IoT: зростаючий стандарт [Електронний ресурс] // LinkedIn. – 2023. – 19 червня. – URL: https://www.linkedin.com/pulse/node-red-%D0%B2-%D0%BF%D1%80%D0%BE%D0%BC%D0%B8%D1%81%D0%BB%D0%BE%D0%B2%D0%BE%D0%BC%D1%83-iot-%D0%B7%D1%80%D0%BE%D1%81%D1%82%D0%B0%D1%8E%D1%87%D0%B8%D0%B9-

%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82-volodymyr-shapran (дата звернення: 24.11.2025).

22. Індустрія 4.0 в Україні: IIoT, PLC, Modbus, Node-RED, GoogleSheet [Електронний ресурс]. – URL: https://sites.google.com/view/i4uinua/технології/iiot/plc-modbus-nodered-googlesheet (дата звернення: 20.11.2025).

23. Stud.com.ua. (n.d.). Теорема Котельникова (теорема відліків). Отримано 20 грудня 2025 з https://stud.com.ua/171390/tehnika/teorema_kotelnikova_teorema_vidlikiv (дата звернення: 20.11.2025).

24. Poddar S., Sanikal V., Thakare S., Sharma N. Mechanical Engineering Integrity in Industry 4.0 & Cybersecurity Protocols for CAE-Based Digital Twins. International Research Journal of Engineering and Technology (IRJET). 2025. Vol. 12, Issue 03. P. 918–923. URL: https://www.irjet.net/archives/V12/i3/IRJET-V12I3141.pdf (дата звернення: 20.11.2025).

25. Akbarian F., Fitzgerald E., Kihl M. Synchronization in Digital Twins for Industrial Control Systems. arXiv preprint arXiv:2006.03447v1. 2020. 6 p. URL: https://arxiv.org/pdf/2006.03447v1.pdf (дата звернення: 20.11.2025).

БІБЛІОГРАФІЧНА ДОВІДКА

Тема магістерської роботи: «Розроблення цифрового двійника автоматизованої системи управління технологічним процесом».

Загальний пояснювальної записки: 88 сторінки.

Перелік графічного матеріалу:

МР.АКСм – 02.00.00.000Е1 – Концептуальна модель структури цифрового двійника − 1 аркуш

МР.АКСм – 02.00.00.000Е2 – Загальна архітектура системи цифрового двійника − 1 аркуш

МР.АКСм – 02.00.00.000Е3 – Діаграма станів симулятора компресорної станції − 1 аркуш

МР.АКСм – 02.00.00.000Е4 − Графіки параметрів системи в різних станах − 2 аркуші

МР.АКСм – 02.00.00.000Е5 − Схеми Node-RED flow − 2 аркуші

МР.АКСм – 02.00.00.000Е6 − Графіки динаміки параметрів компресорної станції − 2 аркуші

Дата завершення магістерської роботи 15.12.2025 року

Студент-магістр **Андрій Бачевський**