КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

кібербезпеки та комп'ютерної інженерії

(назва випускової кафедри)

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему:

«Архітектура цифрової тіні для автоматизованої лінії складання»

Гапончук Ярослав Валерійович

(прізвище, ім’я та по батькові здобувача повністю)

Київ 2026 р

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

кібербезпеки та комп'ютерної інженерії

(назва кафедри)

ЗАТВЕРДЖУЮ

Завідувач кафедри

к.т.н., доц. Делембовський М.М.

«___» _____ 2026 року

КВАЛІФІКАЦІЙНА РОБОТА
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

На тему: «Архітектура цифрової тіні для автоматизованої лінії складання»

(Назва)

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач Гапончук Ярослав Валерійович
(прізвище, ім’я та по батькові повністю)
123 «Комп'ютерна інженерія»
(спеціальність)
Комп'ютерні системи і мережі
(освітня програма)
Група КСМс-23
Керівник Гуменний Д.О.
(прізвище та ініціали)
к.т.н. доц.
(вчене звання, науковий ступінь)
Рецензент Шимкович.В.М.
(прізвище та ініціали)
Ідентичність підтверджую

Київ 2026р

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: Автоматизації і інформаційних технологій

Випускова кафедра: Кібербезпеки та комп'ютерної інженерії

Ступінь вищої освіти: бакалавр

Спеціальність: 123 «Комп'ютерна інженерія»

Освітня програма: «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Делембовський М.М.

«___» _____ 2026 р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Гапончук Ярослав Валерійович

(прізвище, ім'я та по батькові здобувача)

1. Тема роботи: «Архітектура цифрової тіні для автоматизованої лінії складання», затверджена наказом ректора КНУБА № 5775-15/13.2/26 від «14» травня 2026 року.

2. Керівник роботи: к.т.н. доц. Гуменний Д.О.

3. Термін подання здобувачем роботи до захисту: «___» _____2026 р.

4. Зміст пояснювальної записки за розділами:

Розділ 1. Теоретичні основи цифрової тіні в контексті Industry 4.0 та комп'ютерних систем.

Розділ 2. Аналіз вимог та проєктування архітектури цифрової тіні для автоматизованої лінії складання.

Розділ 3. Практична реалізація системи цифрової тіні автоматизованої лінії складання.

5. Графічний матеріал за розділами: 12 слайдів презентації формату А4 (альбомний).

6. Календарний план виконання роботи:

Види робіт та їх зміст	Дата виконання
Розділ 1	13.05.2026
Розділ 2	21.05.2026
Розділ 3	31.05.2026
Остаточне оформлення роботи	03.06.2026
Направлення роботи для перевірки на плагіат	09.06.2026
Попередній захист роботи	09.06.2026
Направлення роботи на рецензування	04.06.2026

7. Дата видачі завдання: «___» _____ 2026 р.

Керівник: _____ _____

Здобувач: _____ _____

РЕЗЮМЕ (SUMMARY) до кваліфікаційної випускової роботи здобувача	ПІБ здобувача українською та англійською мовами Гапончук Ярослав Валерійович Haponchuk Yaroslav Valeriyovich		
ЗВО	Київський національний університет будівництва і архітектури		
Тема (українською та англійською)	Архітектура цифрової тіні для автоматизованої лінії складання Architecture of a Digital Shadow for an Automated Assembly Line		
Освітній ступінь	Бакалавр		
Факультет	Автоматизації і інформаційних технологій		
Випускова кафедра	Кібербезпеки та комп'ютерної інженерії		
Спеціальність	123 Комп'ютерна інженерія		
Освітня програма	Комп'ютерні системи і мережі		
Керівник	Гуменний Дмитро Олександрович, к.т.н., доцент		
Обсяг роботи:	Поснювальна записка, стор.	Розділів	Презентація, кількість слайдів
	108, разом з додатками 132	3	12
Розділ 1	Теоретичні основи цифрової тіні в контексті Industry 4.0 та комп’ютерних систем.		
Розділ 2	Аналіз вимог та проєктування архітектури цифрової тіні для автоматизованої лінії складання.		
Розділ 3	Практична реалізація системи цифрової тіні автоматизованої лінії складання.		
Висновки по роботі	Розроблено архітектуру цифрової тіні автоматизованої лінії складання, реалізовано збір, передачу та візуалізацію виробничих даних, підтверджено працездатність системи на симуляційному стенді.		
Ключові слова: Keywords:	цифрова тінь, Industry 4.0, кіберфізична система, автоматизована лінія складання, MQTT, Node-RED, Grafana, моніторинг виробництва digital shadow, Industry 4.0, cyber-physical system, automated assembly line, MQTT, Node-RED, Grafana, production monitoring		

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці архітектури цифрової тіні для автоматизованої лінії складання та її практичній реалізації на базі відкритого програмного забезпечення. Обсяг пояснювальної записки становить 108 сторінок, загальний обсяг роботи разом із додатками - 132 сторінки, ілюстративні матеріали включають 24 рисунки і 22 таблиці у основній частині та 4 додатки.

Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел інформації та додатків. У першому розділі проаналізовано концепцію Industry 4.0, кіберфізичні системи та три типи цифрового представлення фізичних об’єктів; обґрунтовано доцільність використання цифрової тіні для задач моніторингу автоматизованих ліній складання та сформовано багаторівневу архітектуру системи з фізичного, комунікаційного, обробного, зберігаючого та візуалізаційного рівнів.

У другому розділі проаналізовано об’єкт автоматизації, виявлено критичні точки контролю, сформульовано функціональні та нефункціональні вимоги до системи, обґрунтовано вибір технологічного стеку (CoppeliaSim, Eclipse Mosquitto, Node-RED, InfluxDB, Grafana) і розроблено детальну архітектуру з моделями даних та діаграмою потоків. Виконано розрахунок необхідної пропускної здатності мережі та інтегрального показника OEE.

У третьому розділі виконано розгортання та налаштування всіх компонентів технологічного стеку, побудовано симуляційну модель лінії з шести станцій, реалізовано публікатор MQTT, потоки обробки Node-RED, схеми бакетів InfluxDB та дашборди Grafana з системою сповіщень. Здійснено тестування системи за п’ятьма сценаріями, що підтвердило її функціональність, продуктивність та масштабованість в умовах симуляційного моделювання.

Ключові слова: цифрова тінь, Industry 4.0, автоматизована лінія складання, кіберфізична система, MQTT, Node-RED, InfluxDB, Grafana, CoppeliaSim, моніторинг виробничих процесів.

ABSTRACT

The qualification work is devoted to the development of a digital shadow architecture for an automated assembly line and to its practical implementation based on open source software. The explanatory note consists of 108 pages, while the total volume of the work including appendices is 132 pages, 24 figures and 22 tables in the main part, and 4 appendices.

The work consists of an introduction, three chapters, conclusions, a list of references and appendices. The first chapter analyses the Industry 4.0 concept, cyber-physical systems and three types of digital representation of physical objects; substantiates the feasibility of using a digital shadow for monitoring tasks of automated assembly lines; and proposes a multi-layered architecture of the system that includes physical, communication, processing, storage and visualization layers.

The second chapter analyses the automation object, identifies critical control points, formulates functional and non-functional requirements for the system, justifies the choice of the technology stack (CoppeliaSim, Eclipse Mosquitto, Node-RED, InfluxDB, Grafana) and develops a detailed architecture with data models and a data flow diagram. Calculations of the required network bandwidth and the OEE integral indicator are performed.

The third chapter describes the deployment and configuration of all components of the technology stack, the construction of a simulation model of a six-station line, the implementation of an MQTT publisher, Node-RED processing flows, InfluxDB bucket schemas and Grafana dashboards with an alert system. Testing of the system was carried out under five scenarios, which confirmed its functionality, performance and scalability under simulation conditions.

Keywords: digital shadow, Industry 4.0, automated assembly line, cyber-physical system, MQTT, Node-RED, InfluxDB, Grafana, CoppeliaSim, production process monitoring.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Скорочення	Розшифрування
API	Application Programming Interface (програмний інтерфейс)
CAD	Computer-Aided Design (системи автоматизованого проєктування)
CPS	Cyber-Physical System (кіберфізична система)
DFD	Data Flow Diagram (діаграма потоків даних)
ERP	Enterprise Resource Planning
HMI	Human Machine Interface
IIoT	Industrial Internet of Things
IoT	Internet of Things
ISA	International Society of Automation
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
MES	Manufacturing Execution System
MQTT	Message Queuing Telemetry Transport
OEE	Overall Equipment Effectiveness
OPC UA	Open Platform Communications Unified Architecture
PLC	Programmable Logic Controller
QoS	Quality of Service
SCADA	Supervisory Control and Data Acquisition
TLS	Transport Layer Security

ЗМІСТ

ВСТУП

Сучасне промислове виробництво перебуває на етапі глибокої трансформації, пов’язаної з впровадженням концепції четвертої промислової революції (Industry 4.0). Ця концепція передбачає глибоку інтеграцію інформаційних та операційних технологій, формування кіберфізичних систем і побудову розумних фабрик з безперервним обміном даними в реальному часі. У такому контексті особливого значення набувають технології цифрового представлення фізичних об’єктів, серед яких цифрові моделі, цифрові тіні та цифрові двійники забезпечують моніторинг, аналіз і прогнозування поведінки виробничих систем.

Автоматизовані лінії складання є одним із найпоширеніших об’єктів промислового виробництва, для якого впровадження цифрової тіні дає найбільший економічний ефект. Циклічний характер процесу, велика кількість сенсорів і виконавчих механізмів, висока вартість простоїв та критичні вимоги до якості формують природну потребу в безперервному моніторингу та аналізі виробничих параметрів. Сучасні дослідження свідчать про те, що впровадження цифрових представлень виробничих ліній здатне підвищити загальну ефективність обладнання, помітно скоротити час простоїв та зменшити кількість дефектної продукції.

Актуальність роботи зумовлена кількома взаємопов’язаними чинниками. По-перше, активним поширенням принципів Industry 4.0 серед вітчизняних промислових підприємств, які потребують доступних і масштабованих рішень для цифровізації. По-друге, появою у 2021 році стандарту ISO 23247, що формалізує референсну архітектуру цифрового двійника у виробництві та потребує практичної перевірки на реальних об’єктах. По-третє, доступністю якісних відкритих компонентів програмного забезпечення (Eclipse Mosquitto, Node-RED, InfluxDB, Grafana, CoppeliaSim), що дозволяють побудувати повноцінну систему цифрової тіні без значних капітальних витрат на ліцензії.

Метою кваліфікаційної роботи є розробка архітектури цифрової тіні для автоматизованої лінії складання та її практична реалізація на базі відкритого

програмного забезпечення з демонстрацією повного циклу від генерації телеметричних даних до візуального представлення оператору.

Для досягнення поставленої мети у роботі вирішено такі завдання:

- проаналізовано теоретичні основи концепції Industry 4.0, кіберфізичних систем та різних типів цифрового представлення фізичних об’єктів;
- обґрунтовано доцільність використання цифрової тіні для задач моніторингу автоматизованих ліній складання;
- розроблено багаторівневу архітектуру системи цифрової тіні;
- обґрунтовано вибір технологічного стеку та виконано порівняльний аналіз;
- реалізовано симуляційну модель автоматизованої лінії складання у середовищі CoppeliaSim;
- розроблено програмні модулі публікатора MQTT, потоків обробки Node-RED та схем бакетів InfluxDB;
- побудовано інтерактивні дашборди в Grafana з системою сповіщень;
- проведено тестування системи та оцінено її продуктивність і масштабованість.

Критеріями успішності реалізації системи цифрової тіні визначено такі кількісні та якісні показники:

Таблиця В.1 – Критерії приймання системи цифрової тіні

№	**Критерій успішності**	**Цільове значення**	**Фактичне значення (за результатами тестування)**	**Статус**
1	Медіанна end-to-end затримка синхронізації	≤ 2,0 секунди	0,85 секунди	Виконано
2	95-й перцентиль затримки	≤ 5,0 секунд	2,1 секунди	Виконано
3	Максимальна пропускна здатність	≥ 200 повідомлень/с	800 повідомлень/с	Виконано
4	Втрати пакетів MQTT (QoS 1)	≤ 0,1 %	0 %	Виконано

Продовження таблиці В.1

5	Точність розрахунку OEE	± 2 % від реального значення	± 0 % (повна відповідність)	Виконано
6	Час виявлення критичних аномалій	≤ 30 секунд	7–13 секунд	Виконано
7	Масштабованість (кількість паралельних ліній)	≥ 4 лінії	4 лінії (тестовий сценарій)	Виконано
8	Доступність дашбордів для оператора	99,5 %	100 % (в умовах симуляції)	Виконано

Джерело: складено автором за результатами тестування системи (розділ 3).

Об’єктом дослідження є процес цифровізації автоматизованих ліній складання дискретних виробів. Предметом дослідження є архітектура, методи та програмні засоби побудови системи цифрової тіні для зазначеного класу об’єктів.

Методологічною основою роботи є системний підхід до аналізу складних кіберфізичних систем, методи об’єктно-орієнтованого та потокового програмування, методи теорії масового обслуговування для оцінки пропускної здатності систем зв’язку, а також методи імітаційного моделювання та статистичного аналізу для оцінювання продуктивності.

Практичне значення отриманих результатів полягає у можливості використання розробленої архітектури та її програмної реалізації як шаблону для подальших проєктів цифровізації автоматизованих ліній складання різного призначення. Усі компоненти системи побудовані на відкритих ліцензіях, що знімає бар’єр входу для підприємств малого та середнього бізнесу і дозволяє пристосовувати рішення під специфічні умови експлуатації.

Структура роботи відповідає затвердженому Положенню про кваліфікаційну роботу здобувачів вищої освіти КНУБА та включає вступ, три розділи основної частини, висновки, список використаних джерел інформації та чотири додатки. У першому розділі розглянуто теоретичні основи цифрової тіні у контексті Industry 4.0 та комп’ютерних систем. Другий розділ присвячений аналізу вимог та проєктуванню архітектури цифрової тіні для автоматизованої лінії складання. Третій розділ містить опис практичної реалізації системи, тестування та оцінювання її ефективності. Основна частина пояснювальної

записки становить 108 сторінок, загальний обсяг роботи разом із додатками - 132 сторінки з 24 рисунками та 22 таблицями у основній частині, а також чотирма додатками.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ЦИФРОВОЇ ТІНІ В КОНТЕКСТІ INDUSTRY 4.0 ТА КОМП'ЮТЕРНИХ СИСТЕМ

1.1 Концепція Industry 4.0 та цифровізації виробництва

1.1.1 Історія промислових революцій

Промислові революції представляють собою ключові етапи трансформації суспільно-економічних відносин, пов'язані з радикальними змінами в технологіях виробництва, енергетиці та організації праці. Кожна наступна революція не лише підвищувала продуктивність, але й кардинально змінювала структуру економіки, соціальні відносини та повсякденне життя людини. У сучасній науковій літературі традиційно виділяють чотири основні промислові революції.

Перша промислова революція (приблизно 1760–1840 рр.) розпочалася у Великій Британії і ознаменувала перехід від ручної ремісничої праці до машинного фабричного виробництва. Центральним технологічним проривом стало вдосконалення парової машини Джеймсом Ваттом. Парові двигуни дозволили замінити силу води, вітру та м'язів людини на надійне механічне джерело енергії. Це призвело до механізації текстильної промисловості, розвитку металургії, будівництва залізниць і пароплавів. Виробництво вперше стало концентруватися на великих фабриках, що спричинило швидку урбанізацію, зростання міст і формування промислового пролетаріату. Водночас з'явилися гострі соціальні проблеми: експлуатація дитячої праці, погіршення умов життя робітників і екологічне забруднення. Перша промислова революція заклала основу сучасного індустріального суспільства та капіталістичної економіки.

Друга промислова революція (кінець XIX – початок XX ст., приблизно 1870–1914 рр.) часто називають технологічною або електричною. Її головними рушіями стали електрифікація виробництва та впровадження конвеєрного (масового) виробництва. Електродвигуни дозволили значно підвищити гнучкість розміщення обладнання, забезпечити безперервну роботу підприємств і

поліпшити освітлення. Яскравим прикладом нової організації праці став конвеєр Генрі Форда (1913 р.), завдяки якому автомобіль Model T став доступним для широких верств населення. Розвиток нафтової, хімічної та сталеливарної промисловостей, поява двигунів внутрішнього згоряння та засобів телекомунікацій суттєво прискорили глобальну торгівлю й транспорт. Цей період характеризувався стандартизацією продукції, науковою організацією праці (тейлоризм) і швидким зростанням продуктивності. Однак посилення концентрації капіталу та монополізації призвело до загострення соціальних конфліктів.

Третя промислова революція (друга половина XX ст., з 1960–1970-х рр.) відома як цифрова або комп'ютерна. Вона базувалася на впровадженні електроніки, комп'ютерів і автоматизації виробничих процесів. Виникнення напівпровідникових технологій, мікропроцесорів і програмного керування верстатами з числовим програмним керуванням (ЧПК) дозволило перейти від механічного до електронного управління виробництвом. З'явилися промислові роботи, автоматизовані лінії та перші системи CAD/CAM. У кінці періоду почав формуватися глобальний Інтернет, що радикально змінив обмін інформацією. Третя революція сприяла переходу до постіндустріального (інформаційного) суспільства, де знання та технології стали головним фактором конкурентоспроможності. Виробництво стало гнучкішим, а якість продукції – вищою, але водночас зросла потреба в кваліфікованій робочій силі та виникли виклики щодо перерозподілу робочих місць.

Четверта промислова революція (Industry 4.0, з початку 2010-х рр.) – це поточний етап розвитку, який радикально відрізняється від попередніх за швидкістю, масштабом і системним впливом.

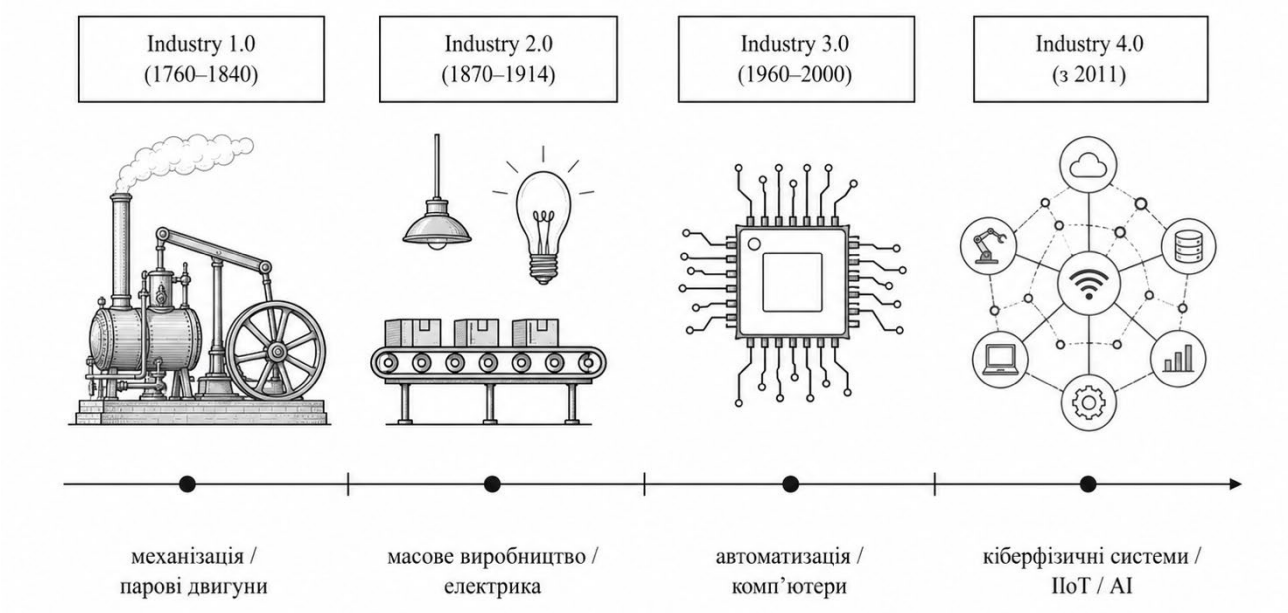


Рисунок 1.1 – Еволюція промислових революцій

Джерело: розроблено автором.

На думку Клауса Шваба, засновника Всесвітнього економічного форуму, четверта промислова революція розвивається експоненціально, а не лінійно, і має глибокий системний вплив на всі сфери життя – від економіки й управління до людської ідентичності та етики. Серед головних викликів – кібербезпека, трансформація ринку праці (зникнення багатьох рутинних професій і потреба в нових компетенціях), етичні питання застосування штучного інтелекту та посилення соціальної нерівності.

Таким чином, кожна промислова революція не лише підвищувала продуктивність праці, але й радикально трансформувала суспільство. Якщо перші три революції поступово формували індустріальне та постіндустріальне суспільство, то Industry 4.0 створює умови для переходу до повністю інтегрованої, інтелектуальної та гнучкої економіки, де межі між виробництвом, послугами та інформацією стають усе більш розмитими.

1.1.2 Поняття Industry 4.0

Термін Industry 4.0 (Industrie 4.0) вперше з'явився у 2011 році в Німеччині. Його офіційно представили на Ганноверському ярмарку (Hannover Messe) в рамках національної стратегії «High-Tech Strategy 2020». Ініціатива була

спрямована на підтримку конкурентоспроможності німецької промисловості через глибоку цифровізацію та інтеграцію сучасних інформаційних технологій у виробничі процеси. Згодом концепція Industry 4.0 набула широкого міжнародного поширення і сьогодні розглядається як основа четвертої промислової революції.

Сутність Industry 4.0 полягає в глибокій інтеграції інформаційних технологій (IT) з операційними технологіями виробництва (OT). На відміну від попередніх промислових революцій, четверта революція створює кіберфізичні системи, у яких фізичні об’єкти (машини, обладнання, продукти) постійно взаємодіють із цифровим середовищем. Це забезпечує принципово новий рівень гнучкості, ефективності та індивідуалізації виробництва.

Ключовою характеристикою Industry 4.0 є автоматичний обмін даними в режимі реального часу між усіма учасниками виробничого процесу. Завдяки вбудованим сенсорам, обчислювальним пристроям та надійним мережам зв’язку машини, системи та люди отримують актуальну інформацію миттєво, що дозволяє реалізовувати самоорганізацію та самокорекцію виробничих процесів.

Важливу роль у концепції Industry 4.0 відіграють різні види інтеграції: вертикальна (від сенсорів і PLC до MES та ERP), горизонтальна (по всьому ланцюжку створення вартості) та наскрізне інженерне забезпечення (end-to-end engineering). Реалізація цих принципів стала можливою завдяки комплексу ключових технологій Industry 4.0, які розглянуто в наступному підрозділі.

1.1.3 Основні технології Industry 4.0

Концепція Industry 4.0 базується на комплексі сучасних інформаційних технологій, що забезпечують інтеграцію фізичних виробничих процесів з цифровими системами. Основні технології Industry 4.0 показані на рисунку 1.2.

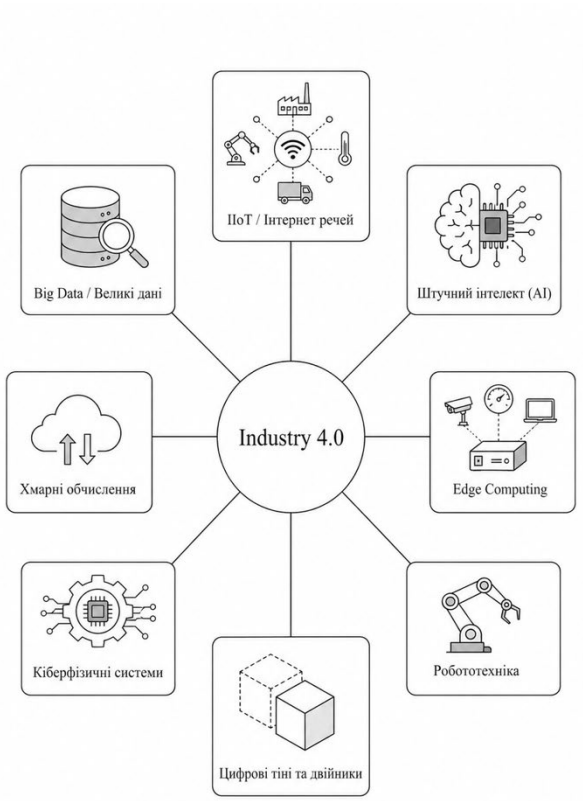


Рисунок 1.2 – Основні технології Industry 4.0

Джерело: розроблено автором.

До ключових технологій належать:

Інтернет речей (IoT) та Промисловий Інтернет речей (IIoT) Інтернет речей являє собою мережу фізичних об’єктів, оснащених сенсорами та підключених до Інтернету для збору й обміну даними. IIoT є промисловою версією IoT, адаптованою до жорстких умов виробництва з підвищеними вимогами до надійності та безпеки.

Великі дані (Big Data) Великі дані включають технології збору, зберігання та аналізу великих обсягів різнорідної інформації, що надходить від тисяч сенсорів. Основні характеристики – обсяг, швидкість, різноманітність і цінність даних.

Штучний інтелект (AI) Штучний інтелект використовується для аналізу даних, машинного навчання, прогнозування несправностей обладнання та підтримки прийняття рішень.

Хмарні обчислення та Edge Computing Хмарні обчислення забезпечують віддалене зберігання та обробку даних, тоді як edge computing дозволяє обробляти інформацію безпосередньо біля джерела, зменшуючи затримки та підвищуючи надійність системи.

Таким чином, інтеграція цих технологій створює основу для функціонування кіберфізичних систем і розумних фабрик.

1.1.4 Кіберфізичні системи (Cyber-Physical Systems, CPS)

Кіберфізична система – це інтегрована система, що поєднує фізичні об’єкти, сенсори, обчислювальні ресурси та комп’ютерні мережі для моніторингу, аналізу та управління фізичними процесами в режимі реального часу.

CPS забезпечує глибоку інтеграцію фізичних і комп’ютерних систем, де фізичні процеси контролюються та керуються за допомогою обчислювальних алгоритмів. Основними елементами такої інтеграції є сенсори для збору даних про стан фізичного об’єкта, контролери (PLC), що виконують первинну обробку сигналів, мережі зв’язку для надійної передачі даних, а також спеціалізоване програмне забезпечення для аналізу та прийняття рішень. Завдяки цьому CPS дозволяє здійснювати управління в реальному часі, що є критично важливим для сучасного виробництва. Кіберфізичні системи є ключовим елементом Industry 4.0 і становлять технологічну основу функціонування Smart Factory (розумних фабрик). Кіберфізична система має багаторівневу структуру, яка забезпечує взаємодію між фізичним середовищем, обчислювальними системами та користувачем. Така структура дозволяє організувати збір даних, їх передачу, обробку, аналіз та управління фізичними процесами.

Схематично потік інформації в CPS представлено на рисунку 1.3

Рисунок 1.3 – Потік інформації в CPS

Джерело: розроблено автором.

Принцип роботи CPS

Робота кіберфізичної системи базується на замкненому циклі управління, який забезпечує постійну взаємодію між фізичним і цифровим середовищем. Основний цикл включає чотири ключові етапи:

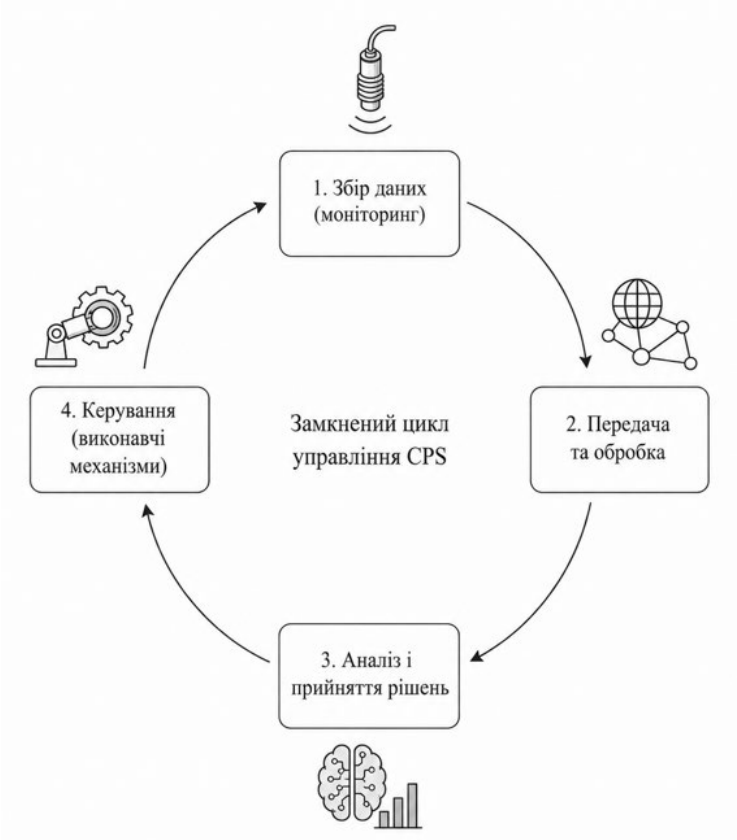


Рисунок 1.4 – Цикл управління

Джерело: розроблено автором.

Однією з найпоширеніших моделей архітектури кіберфізичних систем є 5C-архітектура. Вона включає п'ять послідовних рівнів, що забезпечують

поступовий перехід від простого збору даних до автономного інтелектуального управління (див. рис. 1.5):

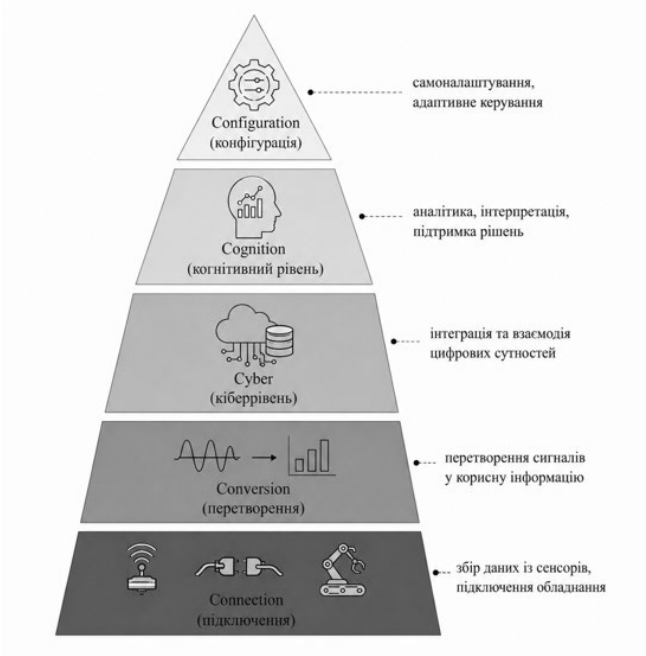


Рисунок 1.5 – 5C-архітектура кібер-фізичних систем (CPS) у рамках Industry 4.0

Джерело: розроблено автором.

Таким чином, кіберфізичні системи є технологічною основою сучасних інтелектуальних виробничих систем. Одним із ключових напрямків їх застосування є концепція розумної фабрики (Smart Factory).

1.1.5 Smart Factory (Розумна фабрика)

Smart Factory (розумна фабрика) є практичною реалізацією концепції Industry 4.0 на рівні підприємства. Це високоавтоматизоване середовище, в якому фізичне обладнання, сенсори та інформаційні системи об’єднані в єдину цифрову екосистему для автономного збору, аналізу та використання даних.

Розумна фабрика здатна самостійно адаптуватися до змін попиту, оптимізувати процеси та прогнозувати збої з мінімальним втручанням людини.

Основні характеристики:

- Повна цифрова інтеграція всіх рівнів виробництва;
- Моніторинг і управління в реальному часі;
- Самооптимізація та адаптивність процесів;
- Прогнозне технічне обслуговування (predictive maintenance);

- Можливість масової кастомізації без втрати продуктивності;
- Висока гнучкість і стійкість до порушень.

Функціонування Smart Factory забезпечують такі технології Industry 4.0, як IIoT і сенсорні мережі, кіберфізичні системи, великі дані та штучний інтелект, edge- і хмарні обчислення, сучасна робототехніка, а також цифрові двійники й цифрові тіні.

Архітектура Smart Factory базується на класичній піраміді автоматизації з повною вертикальною та горизонтальною інтеграцією. Типова п’ятирівнева модель автоматизації представлена на рисунку 1.6.

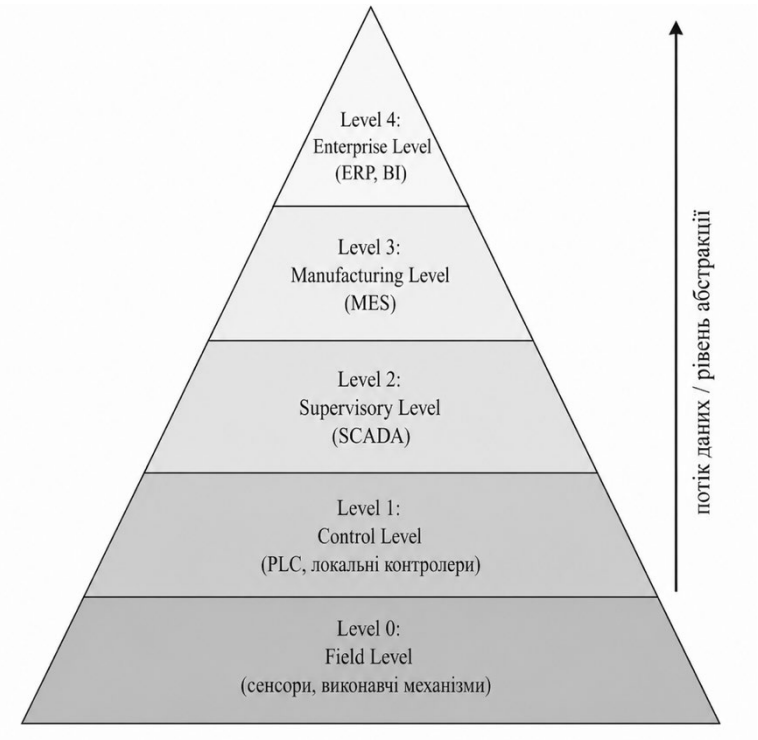


Рисунок 1.6 – Піраміда автоматизації Smart Factory (ISA-95)

Джерело: розроблено автором.

Як показано на рисунку 1.6, архітектура включає такі рівні:

- Field level – сенсори, виконавчі механізми та обладнання;
- Control level – PLC та локальні контролери;
- Supervisory level – SCADA та MES-системи;
- Enterprise level – ERP та бізнес-аналітика;
- Cloud/Analytics level – хмарні платформи, Big Data та AI-аналітика.

Усі рівні працюють як єдина система з обміном даними в реальному часі.

Впровадження розумної фабрики дозволяє підвищити продуктивність, скоротити простої обладнання завдяки прогнозному обслуговуванню, покращити якість продукції, зменшити брак і оптимізувати використання ресурсів.

Таким чином, Smart Factory виступає логічним втіленням принципів Industry 4.0 у промисловості. Саме в такому середовищі цифрові тіні та цифрові двійники стають потужним інструментом моніторингу та оптимізації виробничих процесів.

1.1.6 Цифрова тінь у контексті автоматизованих ліній складання

Автоматизовані лінії складання (assembly lines) є одним із найбільш поширених об’єктів цифровізації в рамках Industry 4.0, оскільки саме вони визначають ефективність сучасного дискретного виробництва. На відміну від традиційних конвеєрних систем масового виробництва, лінії Industry 4.0 характеризуються високою гнучкістю, здатністю до масової кастомізації та інтеграцією колаборативної робототехніки. Основними особливостями таких ліній є можливість швидкої переналадки під індивідуальні замовлення клієнта без значних зупинок, використання промислових роботів та автоматизованих систем подачі компонентів, оптимізація часу циклу (cycle time) як ключового показника продуктивності, інтеграція систем контролю якості в реальному часі (машинний зір, сенсори, RFID), а також горизонтальна та вертикальна інтеграція з ланцюгами постачань для забезпечення just-in-time доставки комплектуючих. Такі характеристики дозволяють поєднувати високу продуктивність з індивідуалізацією продукції, що є критично важливим для автомобільної, авіаційної, електронної та машинобудівної галузей.

У контексті цифровізації таких складних циклічних процесів особливого значення набувають різні форми цифрового представлення фізичного об’єкта – від статичної цифрової моделі до динамічної цифрової тіні та повноцінного цифрового двійника. Саме цифрові тіні та цифрові двійники дозволяють

здійснювати безперервний моніторинг стану обладнання, аналізувати виробничі цикли та виявляти вузькі місця без прямого втручання в роботу контролерів.

Перехід до детального розгляду цих понять, їх відмінностей та особливостей застосування в умовах автоматизованих ліній складання здійснено в наступному підрозділі.

1.2. Поняття цифрової моделі, цифрової тіні та цифрового двійника в інформаційних технологіях

1.2.1 Цифрова модель (Digital Model)

Цифрова модель (Digital Model) є віртуальним представленням фізичного об’єкта, процесу або системи, створеним за допомогою спеціалізованого програмного забезпечення. Вона відображає геометрію, структуру, фізичні властивості та поведінку об’єкта в цифровому середовищі.

За своєю сутністю цифрова модель є статичною або динамічною 3D-моделлю, математичною моделлю чи їх комбінацією. Вона не має постійного зв’язку з реальним фізичним об’єктом і не отримує від нього жодних даних у режимі реального часу. Цифрова модель існує незалежно від фізичного прототипу і використовується переважно на етапах проєктування, розробки та аналізу.

Основним інструментом створення цифрових моделей є системи автоматизованого проєктування CAD (Computer-Aided Design). За допомогою CAD-програм інженери створюють детальні тривимірні моделі деталей, вузлів і цілих виробів. Такі моделі дозволяють проводити різноманітні симуляції – аналіз напруг, теплових процесів, аеродинаміки, кінематики механізмів та багато іншого – без необхідності виготовлення фізичного прототипу. Це значно скорочує час розробки, знижує витрати та зменшує кількість помилок на ранніх стадіях проєктування.

Цифрова модель не отримує дані від реального об’єкта в режимі реального часу. Завдяки цьому вона залишається відносно простим і доступним інструментом, який широко застосовується на етапі концептуального та

технічного проєктування. Прикладами використання цифрових моделей є розробка нових деталей автомобілів, проєктування промислового обладнання, моделювання будівельних конструкцій, створення прототипів споживчих товарів та симуляція технологічних процесів на виробництві.

Для побудови цифрових моделей на практиці найбільш часто застосовуються такі інструменти, як SolidWorks, Autodesk Inventor, CATIA, Creo Parametric, Fusion 360, а також спеціалізоване CAE-програмне забезпечення ANSYS, Abaqus і COMSOL Multiphysics для проведення інженерних розрахунків і симуляцій.

Таким чином, цифрова модель виступає важливим, але статичним елементом цифровізації виробництва, який закладає фундамент для подальшого переходу до більш складних форм цифрового представлення – цифрових тіней та цифрових двійників.

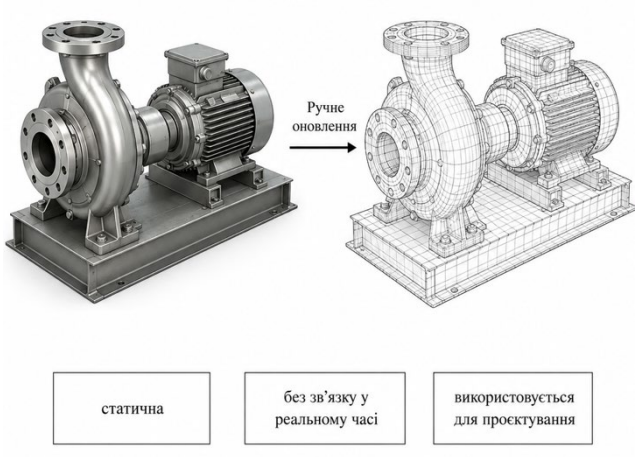


Рисунок 1.7 – Цифрова модель

Джерело: розроблено автором.

1.2.2. Цифрова тінь (Digital Shadow)

Цифрова тінь (Digital Shadow) – це віртуальна модель фізичного об’єкта або процесу, яка отримує дані виключно в одному напрямку – від фізичного світу до цифрового [11; 28; 39]. На відміну від статичної цифрової моделі, цифрова тінь є динамічною і синхронізується з реальним станом об’єкта в режимі near-real-time або реального часу [37].

Основна особливість цифрової тіні полягає в односторонньому потоці даних: фізичний об’єкт (машина, лінія, процес) через сенсори, контролери та промислові протоколи передає інформацію до своєї віртуальної копії, але цифрова тінь не здійснює зворотного впливу на фізичну систему. Таким чином, вона функціонує як точне цифрове відображення (тінь) реального процесу.

У контексті Industry 4.0 цифрова тінь є важливим проміжним етапом цифровізації, оскільки поєднує відносну простоту реалізації з високою практичною цінністю. Вона особливо актуальна для автоматизованих ліній складання, де критичним є надійний моніторинг у реальному часі без ризику втручання у фізичне керування.

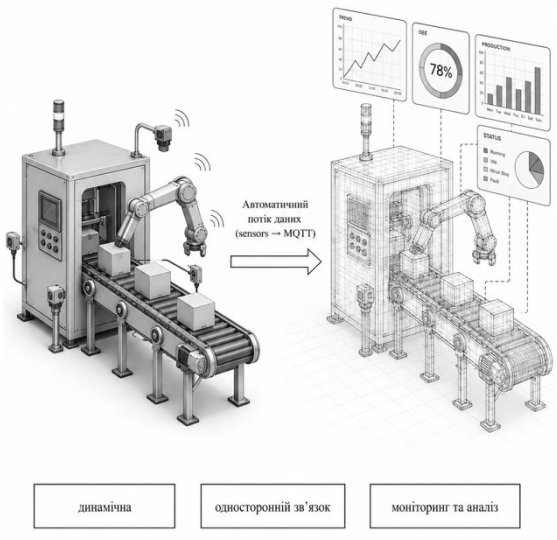


Рисунок 1.8 – Цифрова тінь

Джерело: розроблено автором.

1.2.3. Цифровий двійник (Digital Twin)

Цифровий двійник (Digital Twin) становить найвищий рівень цифрової репрезентації фізичного об’єкта або процесу. На відміну від цифрової тіні, він реалізує повноцінний двосторонній автоматизований обмін даними між фізичним і віртуальним середовищами в режимі реального часу.

Така модель не лише отримує потік даних від сенсорів, контролерів і промислових мереж, але й здатна повертати в фізичну систему результати симуляцій, оптимізовані параметри або керуючі команди. Це дає можливість проводити «what-if» аналіз, прогнозувати поведінку лінії складання та динамічно

коригувати її роботу (наприклад, змінювати швидкість конвеєра, траєкторії роботів чи послідовність операцій) без зупинки виробництва.

Ключовими технічними особливостями цифрового двійника є повна синхронізація в реальному часі, інтеграція з алгоритмами штучного інтелекту для прогнозування та можливість активного впливу на фізичний процес через надійні детерміновані мережі. У контексті комп'ютерної інженерії реалізація двійника вимагає високої пропускної здатності каналів зв'язку, мінімальної латентності та складних механізмів забезпечення безпеки двостороннього обміну.

Для побудови та експлуатації цифрових двійників на практиці широко застосовуються промислові платформи, такі як Siemens MindSphere, PTC ThingWorx, Microsoft Azure Digital Twins, GE Predix, ABB Ability та Bosch IoT Suite. Крім того, для візуалізації, симуляції та взаємодії з моделлю часто використовуються середовища Unity, Unreal Engine та інтегровані рішення на базі MATLAB/Simulink.

Таким чином, цифровий двійник розширює можливості цифрової тіні від режиму спостереження до режиму активного адаптивного управління, завершуючи ієрархію цифрових представлень.

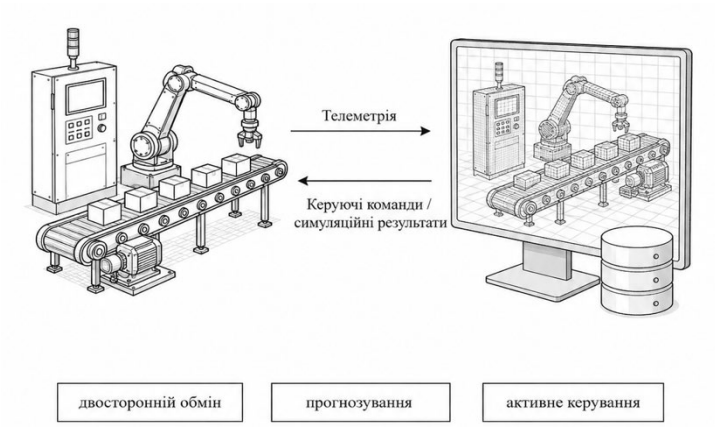


Рисунок 1.9 – Цифровий двійник

Джерело: розроблено автором.

1.2.4. Порівняльний аналіз Digital Model, Digital Shadow та Digital Twin

Для систематизації відмінностей між трьома рівнями цифрової репрезентації фізичного об’єкта проведено порівняльний аналіз за ключовими технічними характеристиками. Результати наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльний аналіз

Критерій порівняння	Digital Model	Digital Shadow	Digital Twin
Потік даних	Відсутній або ручний	Односторонній (фізичний → цифровий)	Двосторонній (↔)
Синхронізація	Статична	Near-real-time / real-time	Повна real-time
Вплив на фізичний об’єкт	Ні	Ні	Так (симуляція + керування)
Джерела даних	CAD/CAE моделі	Сенсори, IoT, PLC, MQTT/OPC UA	Сенсори + симуляція + AI
Застосування в комп’ютерних системах	Проєктування, візуалізація	Моніторинг, аналіз аномалій, дашборди	Прогнозування, оптимізація, автономне керування
Ресурсоємність	Низька	Середня	Висока

Джерело: складено автором за матеріалами розділу 1.

Ієрархічна залежність між цими поняттями наочно демонструється на рисунку нижче.

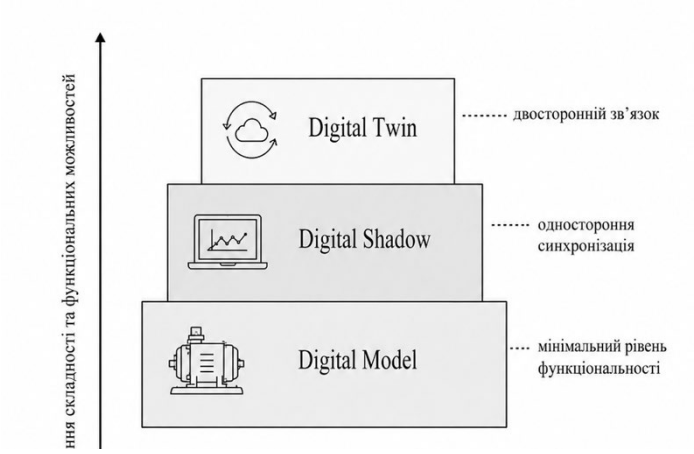


Рисунок 1.10 – Ієрархія Digital Model, Digital Shadow та Digital Twin

Джерело: розроблено автором.

Аналіз таблиці та рисунку виявляє поступове зростання складності та функціональних можливостей. Digital Model виступає статичним віртуальним описом, повністю відокремленим від реального процесу. Digital Shadow переходить до динамічного відображення завдяки односторонньому потоку даних від сенсорів і контролерів через промислові протоколи та edge-обробку. Digital Twin завершує ієрархію, забезпечуючи двосторонній обмін, що дозволяє не лише спостерігати, а й активно впливати на фізичну систему.

У контексті комп'ютерної інженерії перехід між рівнями супроводжується суттєвим збільшенням вимог до надійності мереж, детермінованості затримок передачі даних, механізмів синхронізації та обчислювальних потужностей edge-вузлів. Найвищий рівень вимагає найбільш розвиненої інфраструктури, тоді як середній рівень забезпечує значне розширення аналітичних можливостей при помірних ресурсних витратах.

Таким чином, порівняльний аналіз підкреслює необхідність обґрунтованого вибору рівня цифрової репрезентації залежно від доступної мережевої інфраструктури та завдань автоматизованої лінії складання.

1.2.5. Переваги використання цифрової тіні порівняно з цифровим двійником для задач моніторингу автоматизованих ліній складання

У контексті автоматизованих ліній складання Industry 4.0 вибір між цифровою тінню та цифровим двійником має принципове значення. Такі лінії характеризуються високою циклічністю процесів, жорсткою послідовністю операцій, інтеграцією промислових роботів і конвеєрних систем, а також критичною залежністю від стабільності керування. Будь-яке втручання в фізичний процес може призвести до зупинки всієї лінії, зростання браку або пошкодження обладнання. Саме тому цифровий двійник з двостороннім обміном даними далеко не завжди є оптимальним рішенням.

Цифрова тінь, яка реалізує лише односторонній потік інформації від фізичного об'єкта до віртуальної моделі, пропонує значно кращий баланс для задач моніторингу автоматизованих ліній складання. Основні переваги проявляються саме через специфіку конвеєрного виробництва:

По-перше, цифровий двійник несе суттєві ризики для лінії складання. Можливість надсилати керуючі команди назад у фізичну систему вимагає надзвичайно надійних детермінованих мереж з мінімальною латентністю. У разі помилки в моделі, розсинхронізації даних або кібератаки неправильна команда може зупинити конвеєр, порушити синхронізацію роботів або призвести до зіткнення елементів. На автоматизованій лінії, де зупинка навіть на кілька хвилин спричиняє значні фінансові втрати, такі ризики неприйнятні. Цифрова тінь повністю позбавлена цього недоліку, оскільки функціонує виключно в режимі спостереження та аналізу.

По-друге, для більшості завдань моніторингу лінії складання достатньо саме одностороннього потоку даних [38]. Основними потребами є безперервний контроль часу циклу (cycle time), продуктивності (throughput), виявлення вузьких місць (bottlenecks), аналіз аномалій у роботі роботів і конвеєра, а також можливість відтворення (replay) минулих циклів для ретроспективного аналізу якості складання. Цифрова тінь успішно вирішує всі ці завдання без необхідності активного впливу на процес, що підтверджено у нещодавніх дослідженнях інфраструктури Internet of Production [34] та порівняльних експериментах із цифровим двійником у стохастичних виробничих системах [35].

По-третє, цифровий двійник висуває значно вищі вимоги до обчислювальних ресурсів і мережевої інфраструктури. Він потребує потужних edge- або cloud-обчислень для виконання складних симуляцій «what-if» у реальному часі, високопродуктивних систем синхронізації та надійних двосторонніх каналів зв’язку. У реальних умовах конвеєрного виробництва, де часто присутні обмежені бюджети та вже існуюча інфраструктура на базі PLC і Industrial Ethernet, такі вимоги суттєво ускладнюють і здорожчують проєкт. Цифрова тінь, навпаки, працює з помірними ресурсами: достатньо стабільного одностороннього потоку через MQTT або OPC UA, edge-обробки на рівні Node-RED та бази даних часових рядів InfluxDB.

Крім того, реалізація цифрової тіні дозволяє поступово накопичувати історичні дані про роботу лінії, проводити глибокий аналіз трендів і готувати

ґрунт для майбутнього переходу до повноцінного цифрового двійника, коли з’явиться відповідна мережева інфраструктура та фінансова можливість. Такий еволюційний підхід особливо вигідний для підприємств і навчальних проєктів з обмеженими ресурсами.

Таким чином, для задач моніторингу автоматизованих ліній складання цифрова тінь є більш раціональним і безпечним рішенням порівняно з цифровим двійником. Вона забезпечує необхідний рівень аналітичних можливостей (моніторинг, виявлення аномалій, replay циклів, оптимізація cycle time) при значно нижчих ризиках, менших вимогах до ресурсів і простішій архітектурі.

1.3 Архітектура та принцип функціонування системи цифрової тіні

1.3.1. Загальна концепція архітектури цифрової тіні

Архітектура системи цифрової тіні побудована навколо принципу односторонньої трансляції телеметрії від фізичної автоматизованої лінії складання до її віртуального представлення. Такий підхід дозволяє реалізовувати динамічне відображення стану конвеєра, роботів та технологічних параметрів без будь-якого ризику впливу на реальний цикл управління.

Система цифрової тіні складається з двох фундаментальних частин: фізичного рівня, який включає виробниче обладнання, сенсори, виконавчі механізми та контролери, та цифрового рівня, що відповідає за прийом, передачу, обробку, зберігання та представлення інформації. Дані передаються від фізичного об’єкта до цифрової моделі, формуючи актуальне відображення поточного стану виробничої лінії або технологічного процесу.

Побудова архітектури цифрової тіні базується на принципах Industry 4.0, кіберфізичних систем (Cyber-Physical Systems) та промислового Інтернету речей (IIoT). Для організації обміну інформацією між компонентами системи можуть застосовуватися як класична клієнт-серверна модель, так і архітектура типу publish–subscribe, яка широко використовується в IoT-системах. Такий підхід дозволяє окремим компонентам системи незалежно публікувати дані та

підписуватися на необхідні інформаційні потоки, що підвищує гнучкість, масштабованість та надійність системи.

Основними завданнями архітектури цифрової тіні є збір первинних технологічних даних, їх надійна передача через комп'ютерні мережі, зберігання у базах даних, попередня обробка та аналіз, а також візуалізація у вигляді графіків, панелей моніторингу та цифрових моделей. Таким чином, архітектура цифрової тіні забезпечує безперервний інформаційний потік від фізичної системи до цифрового середовища, створюючи умови для ефективного моніторингу, своєчасного виявлення відхилень та підтримки прийняття рішень щодо оптимізації виробничих процесів.

1.3.2 Багаторівнева архітектура системи цифрової тіні

Архітектура цифрової тіні традиційно будується як ієрархічна багаторівнева система, що є типовим рішенням для кіберфізичних систем, промислового Інтернету речей та концепції Industry 4.0. Кожен рівень виконує чітко визначені функції, пов'язані зі збором, передачею, обробкою, зберіганням та представленням даних. Інформація послідовно проходить через усі шари, зазнаючи при цьому необхідних перетворень, фільтрації та збагачення. Така модульна структура забезпечує високу масштабованість системи, можливість поступового розширення, підвищення надійності та спрощення інтеграції нових компонентів без порушення роботи вже існуючих рівнів.

Схематично багаторівневу архітектуру цифрової тіні можна представити у вигляді послідовної вертикальної ієрархії (рис. 1.11).

Нижче наведено детальний опис кожного рівня багаторівневої архітектури цифрової тіні.

Найнижчим рівнем архітектури є фізичний рівень, який включає безпосередньо виробниче обладнання, сенсори, виконавчі механізми та програмовані логічні контролери. Саме на цьому рівні генеруються первинні технологічні дані, що стають основою для всього подальшого цифрового представлення. Детальний опис компонентів фізичного рівня, наведено в підрозділі 1.3.3.

Рівень збору даних (Data Acquisition) забезпечує захоплення інформації з фізичного середовища. На цьому рівні працюють сенсори, програмовані логічні контролери, модулі аналого-цифрового перетворення та IoT-пристрої. Основними завданнями є зчитування сигналів, їх дискретизація, первинна фільтрація шумів і перетворення фізичних величин у цифрову форму, придатну для подальшої передачі.

Рівень передачі даних (Communication) відповідає за надійну та ефективну доставку інформації між фізичним обладнанням і вищими рівнями системи. Передача здійснюється через промислові мережі (Ethernet, Industrial Ethernet), бездротові канали або спеціалізовані протоколи прикладного рівня. Для організації гнучкого обміну даними широко застосовується архітектура publish–subscribe. Критичними параметрами цього рівня є латентність, пропускна здатність каналів та надійність доставки пакетів у промислових умовах.

Рівень обробки даних (Data Processing) виконує проміжну аналітичну обробку, часто на рівні edge computing. Тут відбувається фільтрація, агрегація, нормалізація даних, виявлення подій та первинне розпізнавання аномалій. Обробка на даному рівні дозволяє значно зменшити обсяг інформації, що передається далі, та підвищити її цінність для наступних шарів системи.

Рівень зберігання даних (Data Storage) призначений для довгострокового накопичення інформації. На цьому рівні використовуються бази даних, орієнтовані на роботу з часовими рядами, що забезпечують зберігання даних разом із точними мітками часу. Це створює надійну основу для історичного аналізу, порівняння циклів виробництва та подальшої поглибленої аналітики.

Рівень візуалізації (Visualization) забезпечує інтерфейс взаємодії оператора з системою. Тут формуються інтерактивні дашборди, графіки, сигналізації та звіти, що дозволяють у реальному часі спостерігати за станом виробничого процесу, аналізувати тенденції та оперативно реагувати на відхилення.

Рівень цифрової тіні (Digital Shadow) є інтеграційним шаром і являє собою результат роботи всієї архітектури. На цьому рівні відбувається синтез усіх зібраних і оброблених даних у єдине динамічне цифрове представлення

виробничої системи. Цифрова тінь поєднує актуальні параметри процесу з віртуальною моделлю, забезпечує можливість відтворення минулих циклів, виявлення аномалій, аналізу ефективності та підтримки прийняття рішень щодо оптимізації виробництва.

Таким чином, багаторівнева архітектура цифрової тіні дозволяє чітко розмежувати функції між рівнями, забезпечуючи високу масштабованість і гнучкість системи. Вона дає змогу легко підключати додаткові джерела даних, розширювати аналітичні можливості, підвищувати обчислювальну потужність на рівні edge та створює технічну основу для подальшого еволюційного переходу від цифрової тіні до повноцінного цифрового двійника з двостороннім обміном даними.

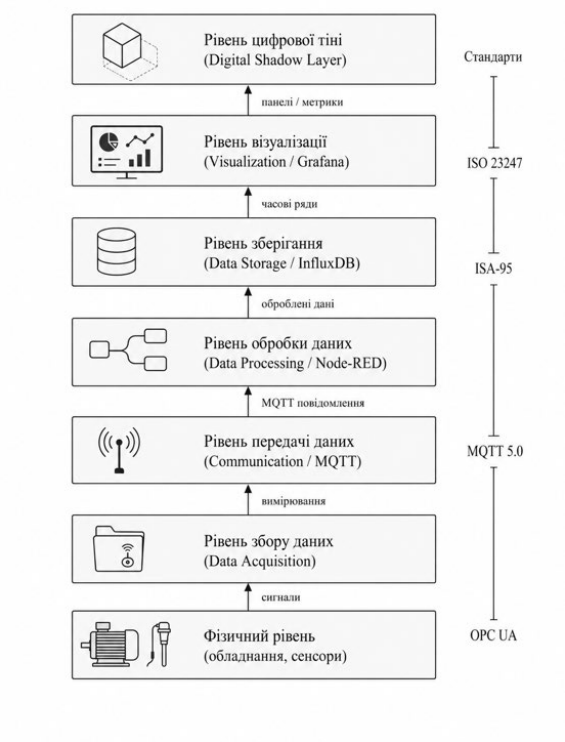


Рисунок 1.11 – Багаторівнева архітектура цифрової тіні

Джерело: розроблено автором.

1.3.3 Фізичний рівень системи цифрової тіні

Фізичний рівень є фундаментальною основою будь-якої системи цифрової тіні. Саме тут відбуваються реальні виробничі процеси, генеруються первинні технологічні дані та формуються умови для подальшого цифрового відображення об’єкта. Без достовірної інформації з фізичного рівня цифрової тіні

як такої не існує, оскільки всі наступні рівні – збір, передача, обробка, зберігання та візуалізація – повністю залежать від даних, отриманих від обладнання або його симуляційної моделі. Сенсори та виконавчі механізми безперервно перетворюють фізичні параметри процесу в цифрові сигнали, які стають основою для створення динамічного віртуального відображення.

Автоматизована лінія складання є одним із найбільш поширених об’єктів цифровізації в сучасній промисловості. Вона являє собою комплекс взаємопов’язаних станцій, призначених для послідовного виконання операцій зі складання виробу: подачі компонентів, позиціонування, кріплення, контролю якості, маркування та вивантаження готової продукції. Лінія працює в циклічному режимі, де ключовими показниками ефективності є час циклу (cycle time), продуктивність (throughput) та наявність буферних зон для компенсації різної швидкості окремих станцій. Основу лінії складають конвеєрні системи, що забезпечують транспортування виробів між станціями, та промислові роботи, які виконують точні маніпуляції з компонентами.

Рисунок 1.12 – Приклад сучасної автоматизованої лінії складання з промисловими роботами та конвеєрною системою

Джерело: розроблено автором.

Важливу роль на фізичному рівні відіграють сенсори та джерела даних. Для забезпечення повноцінного моніторингу використовуються датчики різних типів: датчики положення та близькості, оптичні та індуктивні сенсори, датчики температури, вібрації, струму, швидкості обертання двигунів, а також лічильники виробів і візуальні системи (камери).

Система цифрової тіні збирає широкий спектр параметрів, необхідних для точного відображення стану виробничого процесу. Основні збирані дані вказано в таблиці 1.2.

Таблиця 1.2 – Основні параметри, що збираються на фізичному рівні

Параметр	Опис
Position	Положення виконавчих механізмів та роботів
Speed	Швидкість руху конвеєра та елементів
Cycle time	Час виконання одного виробничого циклу
Operation status	Поточний стан операції (виконується, пауза, помилка)
Temperature	Температура двигунів та критичних вузлів
Vibration	Рівень вібрації обладнання
Motor current	Споживаний струм електродвигунів
Counter	Кількість оброблених виробів
Error states	Стани помилок та аварійних ситуацій
Timestamp	Часова мітка події

Джерело: складено автором за матеріалами розділу 1.

Програмовані логічні контролери (PLC) та інші промислові контролери виступають «мозком» фізичного рівня. Вони не лише збирають дані з численних сенсорів і керують виконавчими механізмами згідно закладеної логіки, але й забезпечують первинну обробку сигналів та підготовку даних для передачі у мережу. Сучасні PLC часто підтримують промислові протоколи обміну даними, що дозволяє ефективно інтегрувати їх у системи цифрової тіні.

Таким чином, фізичний рівень є єдиним джерелом первинних об’єктивних даних для всієї архітектури цифрової тіні. Від якості, повноти та своєчасності інформації, зібраної на цьому рівні, безпосередньо залежить точність і цінність подальшого цифрового представлення виробничого процесу. Саме через

сенсори, контролери та мережі дані з фізичного середовища надходять у цифрову систему, формуючи надійну основу для моніторингу, аналізу аномалій та оптимізації роботи автоматизованої лінії складання.

1.3.4 Рівень передачі даних та комп'ютерні мережі

Рівень передачі даних є критичним елементом архітектури цифрової тіні, оскільки саме він забезпечує надійний і своєчасний обмін інформацією між фізичним обладнанням, контролерами, обчислювальними вузлами та вищими рівнями системи. Мережі з'єднують фізичний і цифровий світи, дозволяючи перетворювати розрізнені сигнали сенсорів у єдиний інформаційний потік. Без ефективної комунікаційної інфраструктури цифрової тіні не може існувати, адже вона залежить від безперервної передачі даних у режимі, наближеному до реального часу. Промислові мережі та IoT-протоколи стають сполучною ланкою в кіберфізичних системах, забезпечуючи інтеграцію обладнання, edge-вузлів і хмарних сервісів у рамках концепції Industry 4.0.

Промислові мережі та Industrial Ethernet складають основу комунікаційної інфраструктури сучасних автоматизованих систем. Класичний Ethernet, адаптований для промислового застосування, отримав розвиток у вигляді Industrial Ethernet. Найпоширенішими протоколами цього класу є PROFINET, EtherCAT та Modbus TCP. Вони забезпечують детерміновану передачу даних на рівні поля (field level) і рівня керування (control level). Для бездротового зв'язку використовуються промислові версії Wi-Fi, а в перспективі – технології 5G, які дозволяють досягти високої пропускної здатності та низької затримки в мобільних і розподілених системах. Мережева топологія включає комутатори (switches), маршрутизатори та сегментацію мережі відповідно до рівнів автоматизації – від польового рівня до рівня підприємства.

Протоколи передачі даних визначають правила обміну інформацією між компонентами системи цифрової тіні. Базовим рівнем залишається стек протоколів TCP/IP, який забезпечує надійну передачу даних у більшості промислових мереж. Для IoT-систем особливо важливим став протокол MQTT (Message Queuing Telemetry Transport) – легкий, ефективний протокол,

призначений для передачі телеметрії в умовах обмежених ресурсів. OPC UA (OPC Unified Architecture) є стандартом промислової взаємодії, що підтримує як клієнт-серверну, так і publish-subscribe модель, забезпечуючи семантичну інтероперабельність і вбудовані механізми безпеки. Додатково можуть застосовуватися протоколи HTTP, WebSocket та REST API для інтеграції з веб-сервісами, а також формат JSON для структурованого представлення повідомлень.

Однією з найбільш ефективних моделей організації обміну даними в системах цифрової тіні є архітектура publish–subscribe. На відміну від традиційної клієнт-серверної моделі, вона дозволяє повністю розв'язати (decouple) виробників даних (publishers) від їх споживачів (subscribers). Центральним елементом архітектури виступає message broker, який маршрутизує повідомлення за темами (topics). Такий підхід забезпечує асинхронну комунікацію, високу масштабованість системи та стійкість до зростання кількості підключених пристроїв.

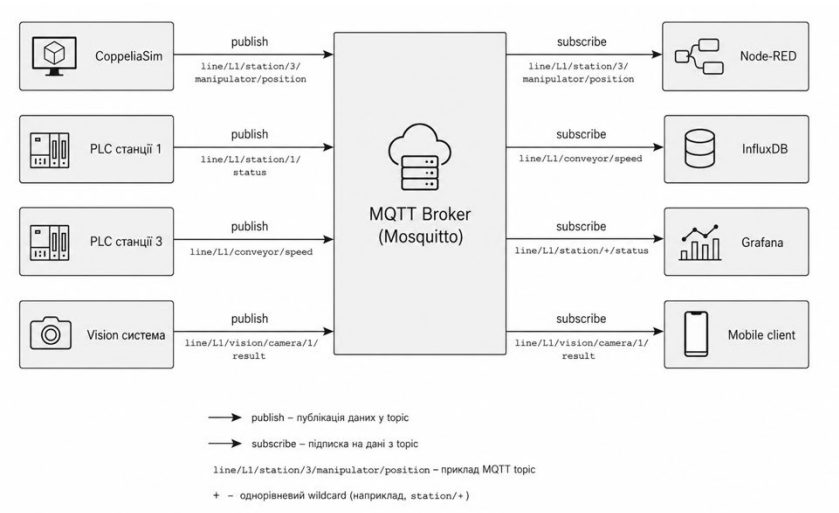


Рисунок 1.13 – Архітектура передачі даних publish–subscribe у системі цифрової тіні

Джерело: розроблено автором.

Промислові мережі повинні забезпечувати стабільну роботу в умовах електромагнітних перешкод, вібрацій та широкого діапазону температур.

Edge computing на рівні передачі даних відіграє важливу роль у підвищенні ефективності системи. Обробка даних безпосередньо біля джерела (на рівні

контролерів або спеціалізованих edge-вузлів) дозволяє значно зменшити обсяг трафіку, що передається в центральну мережу, знизити затримки та підвищити загальну надійність. Попередня фільтрація, агрегація та виявлення аномалій на краю мережі зменшують навантаження на брокер і системи зберігання. У багатьох реалізаціях роль edge-вузла виконує Node-RED, який дозволяє візуально налаштовувати потоки даних, виконувати preprocessing і направляти тільки необхідну інформацію далі.

Таким чином, рівень передачі даних та комп'ютерні мережі є ключовим елементом архітектури цифрової тіні. Використання промислових протоколів MQTT та OPC UA разом з архітектурою publish–subscribe і можливостями edge computing забезпечує ефективну, масштабовану та надійну передачу інформації від фізичного рівня до цифрового представлення. Саме завдяки сучасним комунікаційним технологіям стає можливим створення динамічної цифрової тіні, яка відображає стан автоматизованої лінії складання в режимі, наближеному до реального часу.

1.3.5 Рівень обробки та зберігання даних

Рівень обробки даних відіграє ключову роль у системі цифрової тіні, виступаючи проміжною ланкою між отриманням сирих даних з фізичного рівня та їх подальшим зберіганням, аналізом і візуалізацією. Сирі дані, що надходять від сенсорів і контролерів, зазвичай містять шум, надлишкову інформацію та неструктуровані значення. Тому вони не можуть одразу потрапляти до бази даних або систем візуалізації. Рівень обробки перетворює потік телеметрії в структуровану, очищену та цінну інформацію, придатну для ефективного зберігання та подальшого використання. Саме на цьому рівні формуються data pipelines – конвеєри обробки потокових даних, що забезпечують фільтрацію, агрегацію, нормалізацію та збагачення інформації часовими мітками.

Edge computing суттєво підвищує ефективність рівня обробки. Обробка даних безпосередньо біля джерела (на edge-вузлах або gateway) дає можливість виконувати частину обчислень локально, не відправляючи весь потік сирих даних у центральну мережу чи хмару. Такий підхід зменшує мережеве

навантаження, знижує затримки (latency), підвищує загальну надійність системи та дозволяє реагувати на події в режимі near-real-time. У розподілених IoT-архітектурах edge-вузли стають важливим елементом, який поєднує можливості локальної обробки з подальшою передачею тільки необхідної, вже обробленої інформації.

Одним із найпоширеніших інструментів реалізації edge-обробки в системах цифрової тіні є Node-RED – платформа flow-based programming, яка дозволяє візуально створювати складні потоки обробки даних без глибокого програмування. Завдяки набору готових вузлів (nodes) Node-RED підтримує роботу з протоколом MQTT, виконання функцій JavaScript, маршрутизацію повідомлень, фільтрацію, агрегацію та інтеграцію з різними системами.

Для реалізації edge-обробки також широко застосовуються такі інструменти, як Apache NiFi (потужна платформа для побудови складних data pipelines з візуальним інтерфейсом), Eclipse Streamsheets (інструмент, орієнтований на обробку даних у реальному часі та інтеграцію з IoT-платформами), а також Python-based рішення на базі бібліотек Pandas, Kafka Streams або MQTT-клієнтів. Кожен із цих інструментів має свої переваги щодо продуктивності, масштабованості, зручності розробки та рівня безпеки, що дозволяє обирати оптимальне рішення залежно від конкретних вимог системи.

Типовий потік даних у такій архітектурі виглядає так: дані з MQTT-брокера надходять у інструмент обробки, проходять етапи попередньої обробки (фільтрація, агрегація, нормалізація тощо), після чого спрямовуються до системи зберігання або візуалізації. Такий підхід значно спрощує розробку, налагодження та масштабування конвеєрів обробки даних.

Зберігання даних є наступним критичним етапом після обробки. У промислових системах доводиться працювати з великими обсягами телеметрії, логів подій, історичних записів і результатів моніторингу. Традиційні реляційні бази даних (SQL) часто не найкраще підходять для високошвидкісного запису часових рядів. Тому в системах Industry 4.0 та цифрових тіней широко застосовуються спеціалізовані бази даних часових рядів (time-series databases).

Серед них особливо виділяється InfluxDB – відкрита високопродуктивна time-series database, оптимізована для зберігання промислової телеметрії. Вона використовує концепцію measurement (вимірювання), fields (поля зі значеннями) та tags (мітки для категоризації). InfluxDB характеризується високою швидкістю запису, ефективним стисненням даних і зручною інтеграцією з інструментами візуалізації, зокрема Grafana.

Для зберігання та обробки часових рядів у системах цифрової тіні також широко застосовуються такі інструменти, як TimescaleDB (розширення PostgreSQL з потужними можливостями для time-series даних), Apache Druid (аналітична база даних для великих обсягів даних у реальному часі), Prometheus (система моніторингу та зберігання метрик) та ClickHouse (колонкова база даних з високою продуктивністю для аналітики). Кожен із цих рішень має свої сильні сторони щодо швидкості запису, запиту, масштабованості та інтеграції з іншими компонентами системи.

Така база даних ідеально підходить для зберігання історичних даних, проведення ретроспективного аналізу та розрахунку різноманітних метрик продуктивності.

Історичні дані, накопичені в системі, відкривають широкі можливості для аналітики які показані на таблиці 1.3:

Таблиця 1.3 – Показники та мета аналізу історичних даних цифрової тіні

Показник	Мета аналізу
Cycle time	Оцінка продуктивності станцій
Throughput	Загальна ефективність лінії
Downtime	Виявлення та аналіз простоїв
Error states	Діагностика поломок і збоїв
Temperature	Контроль перегріву та теплового режиму
Vibration	Виявлення зносу обладнання
Trends	Виявлення довгострокових тенденцій
Bottlenecks	Пошук вузьких місць у процесі
Energy consumption	Оптимізація енергоспоживання

Джерело: складено автором за матеріалами розділу 1.

Таким чином, рівень обробки та зберігання даних виконує важливу трансформаційну функцію в архітектурі цифрової тіні. Він перетворює великий

обсяг сирих сенсорних даних у впорядковану, цінну інформацію, готову до зберігання, глибокого аналізу та візуального представлення. Використання edge computing, сучасних інструментів потокової обробки (наприклад, Node-RED) та спеціалізованих баз даних часових рядів (InfluxDB) забезпечує ефективність, масштабованість і готовність системи до подальшого розвитку в напрямку розширеної аналітики.

1.3.6 Рівень візуалізації

Рівень візуалізації є завершальним етапом архітектури цифрової тіні, на якому відбувається перетворення накопичених числових даних і потоків телеметрії у зручну для сприйняття та аналізу форму. Сирі дані самі по собі мають низьку інформативність для людини. Лише через наочне представлення у вигляді графіків, панелей моніторингу та інтерактивних дашбордів вони стають корисною інформацією, що підтримує прийняття оперативних рішень. Саме на цьому рівні оператор отримує цілісне уявлення про поточний стан системи, може швидко виявляти відхилення та оцінювати ефективність виробничого процесу. Фактично, рівень візуалізації забезпечує матеріалізацію цифрової тіні для кінцевого користувача.

Інтерактивні дашборди є основним інструментом візуалізації в сучасних системах цифрової тіні. Вони дозволяють у реальному часі відображати ключові параметри роботи автоматизованої лінії складання, такі як поточний стан обладнання, швидкість руху конвеєра, час виконання циклів (cycle time), продуктивність лінії (throughput), температурний режим, кількість помилок, тривалість простоїв (downtime), а також динаміку ключових показників ефективності (KPI) та виробничих метрик. Промислові дашборди поєднують елементи моніторингу, аналітики та підтримки прийняття рішень, надаючи оператору зручний доступ до актуальної інформації.

Однією з найбільш поширених відкритих платформ для створення таких дашбордів є Grafana – потужна open-source система візуалізації, спеціально адаптована для роботи з часовими рядами. Grafana забезпечує тісну інтеграцію з базами даних типу InfluxDB, дозволяє будувати складні інтерактивні панелі

моніторингу в режимі реального часу, налаштовувати сповіщення (alerts) та проводити базову аналітику. Завдяки великій кількості готових візуальних елементів (лінійні графіки, gauge, таблиці, heatmaps, status-панелі) вона широко застосовується в IoT-системах і промисловому моніторингу.

Крім Grafana, для реалізації рівня візуалізації цифрової тіні широко застосовуються й інші сучасні інструменти. Зокрема, Kibana (частина стеку Elastic Stack) дозволяє створювати гнучкі дашборди з розширеними можливостями пошуку та аналізу логів, що особливо корисно для глибокої діагностики аномалій на автоматизованій лінії складання. Ще одним ефективним рішенням є Power BI (від Microsoft), який забезпечує інтеграцію з різними джерелами даних, побудову інтерактивних звітів і можливість використання штучного інтелекту для автоматичного виявлення трендів у параметрах cycle time, throughput та downtime.

Вибір конкретного інструменту залежить від існуючої IT-інфраструктури підприємства, необхідного рівня складності аналітики та вимог до інтеграції з іншими системами. У навчальних та дослідницьких проєктах перевага часто віддається Grafana завдяки її відкритості, легкості розгортання та відмінній сумісності з InfluxDB.

Схематично взаємозв’язок компонентів рівня візуалізації можна представити наступним чином:

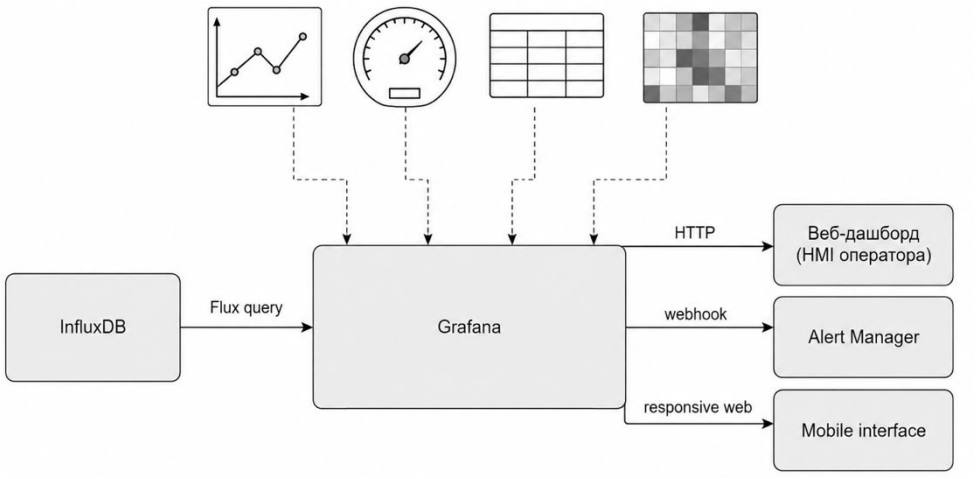


Рисунок 1.14 – Взаємозв’язок компонентів рівня візуалізації цифрової тіні

Джерело: розроблено автором.

Для ефективного представлення інформації на дашбордах використовуються різні типи візуалізацій. Основні з них наведено в таблиці 1.4.

Таблиця 1.4 – Типи графіків та їх призначення в системі цифрової тіні

Тип графіка	Призначення
Line chart	Відображення зміни параметра в часі
Bar chart	Порівняння значень між станціями або періодами
Gauge	Відображення поточного значення відносно норми
Table	Представлення табличних даних і статусів
Heatmap	Візуалізація інтенсивності процесів або аномалій
Status panel	Індикація поточного стану обладнання
Pie chart	Розподіл часу між різними станами (робота, простій тощо)
Alerts panel	Відображення активних попереджень

Джерело: складено автором за матеріалами розділу 1.

Важливим елементом рівня візуалізації є система сповіщень (alerts). Вона забезпечує оперативне реагування на відхилення від нормальних значень шляхом встановлення порогових величин (threshold).

Сповіщення можуть виводитися безпосередньо на дашборді або надсилатися через канали зв’язку, сприяючи ранньому виявленню проблем і реалізації принципів прогнозного обслуговування.

Інтерфейс оператора (Human Machine Interface – HMI) у сучасних системах цифрової тіні часто реалізується саме через веб-дашборди. У цьому контексті Grafana виступає сучасною, гнучкою альтернативою традиційним SCADA-системам, надаючи оператору зручний інструмент моніторингу та підтримки прийняття рішень у виробничому середовищі.

Візуалізація цифрової тіні є найбільш цілісним рівнем системи. Вона об’єднує інтерактивні дашборди, графіки реального часу, історичні тренди, систему сповіщень та інтеграцію з 3D-моделлю лінії. Таким чином, цифрова тінь матеріалізується як єдине віртуальне представлення виробничого процесу, де

дані, аналітика та візуалізація утворюють єдине ціле. Саме на цьому рівні оператор отримує повну картину стану автоматизованої лінії складання.

Таким чином, рівень візуалізації завершує ланцюг перетворення сирих даних у знання, роблячи цифрову тінь практично корисним інструментом для моніторингу, діагностики та оптимізації виробничих процесів.

1.3.7 Потоки даних у системі цифрової тіні

Потоки даних є основою функціонування цифрової тіні. На відміну від статичних цифрових моделей, цифрова тінь існує завдяки постійному односторонньому потоку даних від фізичного об’єкта до цифрового середовища. Цей потік має характер телеметрії та відбувається в режимі реального часу або near-real-time. Дані, збагачені часовими мітками (timestamp), рухаються по чітко визначеному data pipeline – від джерел генерації через етапи збору, передачі, обробки та зберігання до рівня візуалізації. Така безперервна передача потокових даних (streaming data) забезпечує динамічне оновлення віртуального представлення виробничого процесу.

Джерелами даних у системі цифрової тіні виступають різноманітні фізичні та віртуальні компоненти. До них належать сенсори, програмовані логічні контролери, системи машинного зору, лічильники, промислові роботи, конвеєрні системи, а також симуляційне середовище CoppeliaSim.

Таблиця 1.5 – Джерела даних у системі цифрової тіні

Джерело	Тип даних	Приклад даних
Сенсор положення	Position	Координати робота
Сенсор швидкості	Speed	Швидкість конвеєра
PLC	Status	Стан операції
Камера	Image / Data	Контроль якості
Лічильник	Counter	Кількість виробів
CoppeliaSim	Telemetry	Всі параметри лінії

Джерело: складено автором за матеріалами розділу 1.

Передача даних є критичним етапом потоку. На цьому рівні використовуються промислові мережі на базі Industrial Ethernet та протокол TCP/IP. Для ефективної передачі телеметрії в IoT-системах найчастіше

застосовується протокол MQTT, який реалізує архітектуру publish–subscribe. У цій моделі пристрої-видавці (publishers) відправляють повідомлення в брокер за певними темами (topics), а підписники (subscribers) отримують тільки необхідні дані. Такий підхід забезпечує низьке навантаження, масштабованість і підтримку різних рівнів Quality of Service (QoS).

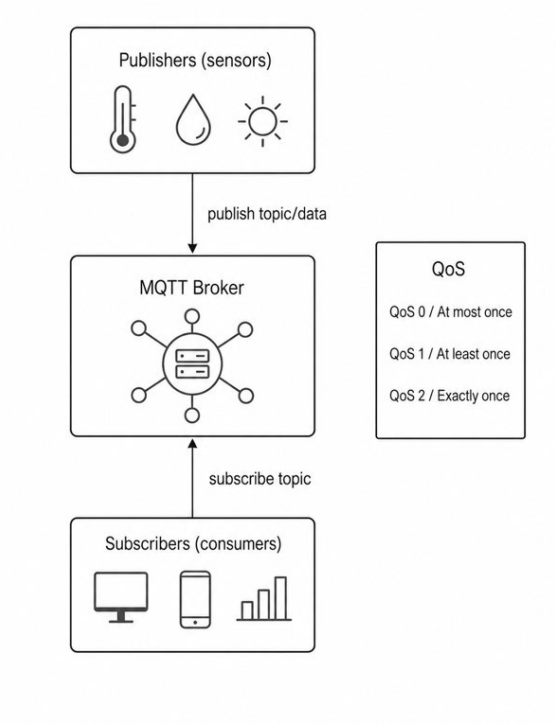


Рисунок 1.15 – Архітектура передачі даних за моделлю publish–subscribe (MQTT)

Джерело: розроблено автором.

Обробка потокових даних виконується на рівні edge computing за допомогою Node-RED, який забезпечує візуальне налаштування фільтрації, агрегації та маршрутизації телеметрії.

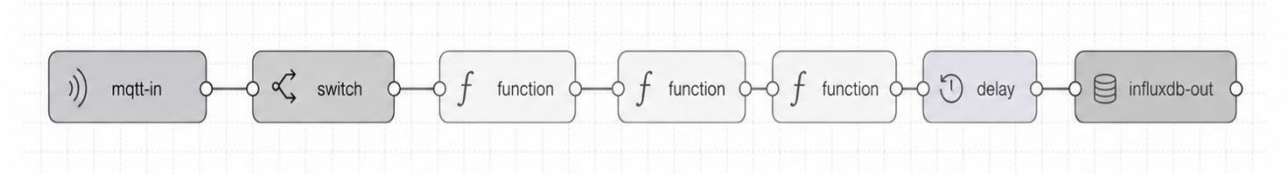


Рисунок 1.16 – Приклад візуального потоку обробки даних у Node-RED

Джерело: розроблено автором.

Зберігання даних здійснюється в спеціалізованій базі даних часових рядів. InfluxDB оптимально підходить для промислової телеметрії завдяки високій швидкості запису та ефективній роботі з даними, позначеними timestamp. У базі зберігаються як поточні, так і історичні значення ключових параметрів.

Таблиця 1.6 – Основні дані, що зберігаються в системі цифрової тіні

Параметр	Тип даних	Опис
Timestamp	Time	Часова мітка події
Position	Float	Координати виконавчих механізмів
Speed	Float	Швидкість руху
Cycle time	Float	Час виконання циклу
Status	String	Поточний стан операції
Counter	Integer	Кількість оброблених виробів
Temperature	Float	Температура вузлів

Джерело: складено автором за матеріалами розділу 1.

Фінальним етапом потоку даних є їх візуалізація та формування цифрової тіні. На цьому рівні Grafana формує інтерактивні дашборди, графіки, панелі моніторингу та систему сповіщень. Одночасно відбувається синхронізація даних з 3D-моделлю лінії, що дозволяє реалізовувати функцію replay (відтворення минулих циклів) та отримувати цілісне віртуальне представлення виробничого процесу.

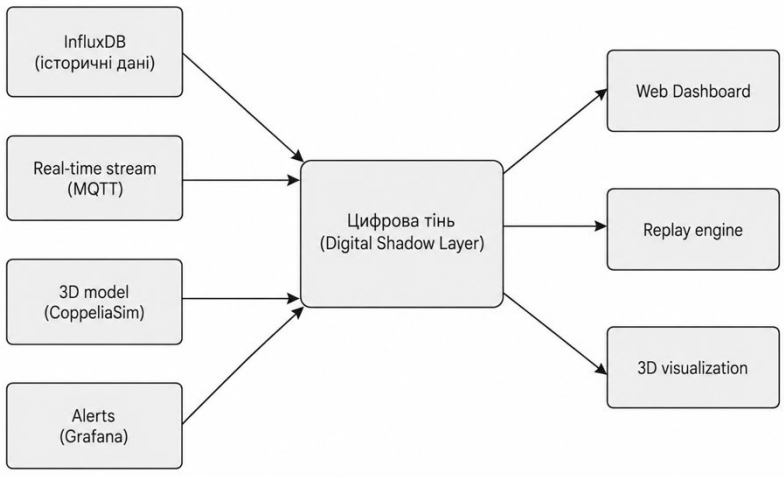


Рисунок 1.17 – Інтеграція даних у візуалізацію цифрової тіні

Джерело: розроблено автором.

Для порівняння основних протоколів передачі даних у системах цифрової тіні можна використовувати наступну таблицю:

Таблиця 1.7 – Порівняльний аналіз протоколів передачі даних

OPC UA	Client-Server / Pub-Sub	Висока безпека, промисловий стандарт	Більш ресурсоємний
HTTP	Request-Response	Простота реалізації	Не підходить для real-time

Протокол		Тип архітектури	Основні переваги	Недоліки
MQTT	Publish-Subscribe	Легкий, масштабований, низьке навантаження		Потребує брокера

Джерело: складено автором за матеріалами розділу 1.

Таким чином, потоки даних у системі цифрової тіні утворюють єдиний безперервний ланцюг від фізичних джерел до віртуального представлення. Ефективна організація цих потоків з використанням сучасних протоколів, інструментів обробки та спеціалізованих баз даних забезпечує актуальність, точність і практичну цінність цифрової тіні як інструменту моніторингу та аналізу автоматизованих виробничих процесів.

1.3.8 Взаємодія компонентів системи цифрової тіні

Система цифрової тіні являє собою складну розподілену архітектуру, що складається з кількох взаємопов’язаних підсистем. Основними компонентами є фізичний рівень (сенсори та симуляційне середовище), мережевий рівень з MQTT-брокером, модуль обробки даних Node-RED, база даних часових рядів InfluxDB, система візуалізації Grafana та 3D-модель, яка формує власне цифрову тінь. Взаємодія між компонентами здійснюється через комбінацію архітектур client-server та publish-subscribe, що забезпечує модульність, масштабованість і гнучкість системи. Така організація дозволяє легко підключати нові джерела даних, розширювати аналітичні можливості та підтримувати стабільну роботу в умовах змінного навантаження.

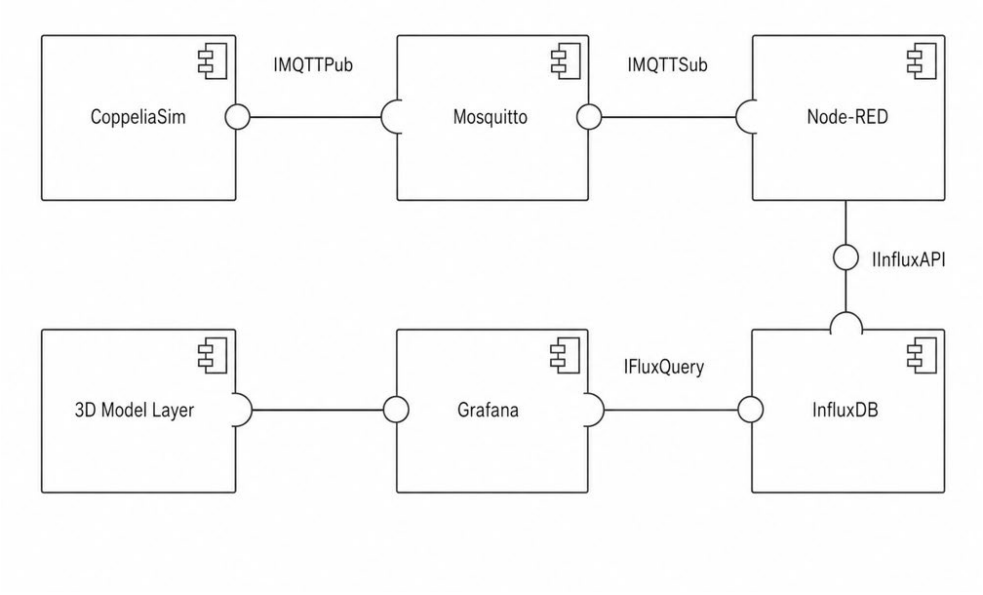


Рисунок 1.18 – Компонентна діаграма системи цифрової тіні

Джерело: розроблено автором.

Взаємодія починається на фізичному рівні, де сенсори та симуляційне середовище CoppeliaSim генерують телеметричні дані. Сенсори фіксують реальні фізичні параметри, а CoppeliaSim імітує поведінку лінії складання. Дані збагачуються часовими мітками та формуються у повідомлення для подальшої передачі. Взаємодія може відбуватися як у режимі polling (опитування), так і event-based (за подією).

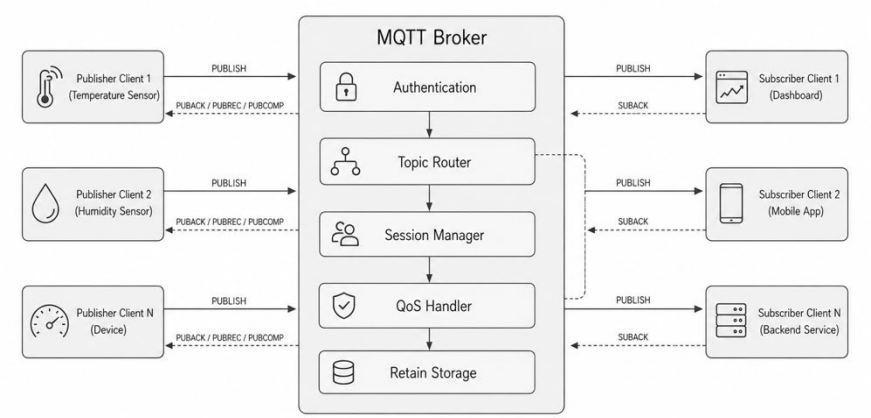
Таблиця 1.8 – Взаємодія компонентів фізичного рівня та рівня збору даних

Компонент	Функція	Тип даних
Сенсори	Збір первинних даних	Position, speed, temperature
PLC	Керування та первинна передача	Status, counter
CoppeliaSim	Симуляція виробничого процесу	Telemetry (всі параметри)
MQTT Publisher	Публікація даних у брокер	JSON-повідомлення

Джерело: складено автором за матеріалами розділу 1.

Центральним елементом комунікації виступає MQTT-брокер, який реалізує модель publish–subscribe. Компоненти-видавці (publishers) надсилають дані за визначеними темами (topics), а підписники (subscribers) отримують тільки потрібну інформацію. Такий підхід забезпечує повне розв’язання компонентів

(decoupling), асинхронну комунікацію, масштабованість та підтримку різних рівнів Quality of Service (QoS).



Стандартний порт	Опис
1883 (TCP)	Стандартний порт MQTT через незахищене TCP-з'єднання
8883 (TLS)	Порт MQTT через захищене TLS-з'єднання
9001 (WebSocket)	Порт MQTT через WebSocket (зручно для веб-клієнтів)

Рисунок 1.19 – Архітектура взаємодії через MQTT-брокер

Джерело: розроблено автором.

Попередня обробка даних здійснюється у Node-RED, який виступає проміжним шаром між MQTT-брокером та базою InfluxDB.Grafana, у свою чергу, читає дані безпосередньо з бази та формує інтерактивні дашборди, графіки та сповіщення.

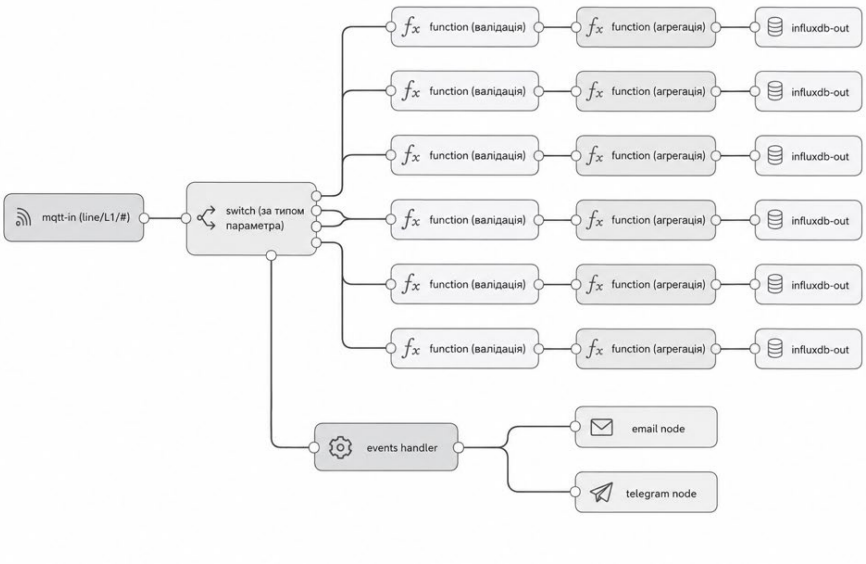


Рисунок 1.20 – Потік даних через Node-RED

Фінальним етапом взаємодії є формування цифрової тіні. На цьому рівні відбувається інтеграція всіх зібраних і оброблених даних із 3D-моделлю лінії.

Синхронізація даних у реальному часі дозволяє створювати динамічне віртуальне представлення системи, відтворювати минулі цикли (replay), проводити аналітику та забезпечувати моніторинг поточного стану.

Таблиця 1.9 – Роль компонентів у формуванні цифрової тіні

Компонент	Роль у цифровій тіні
Сенсори / CoppeliaSim	Джерело первинних даних
MQTT	Надійна передача даних
Node-RED	Обробка та маршрутизація потоків
InfluxDB	Зберігання історичних даних
Grafana	Візуалізація та моніторинг
3D-модель	Віртуальне представлення системи

Джерело: складено автором за матеріалами розділу 1.

Таким чином, взаємодія компонентів системи цифрової тіні побудована як єдиний, добре структурований інформаційний ланцюг. Модульна архітектура з використанням MQTT-брокера та проміжного шару обробки забезпечує високу гнучкість, надійність і можливість подальшого масштабування системи в напрямку повноцінного цифрового двійника.

Висновки до розділу 1

У розділі розглянуто теоретичні основи цифрової тіні в контексті Industry 4.0 та комп’ютерних систем. Проаналізовано еволюцію промислових революцій, сутність концепції Industry 4.0, ключові технології (IIoT, Big Data, штучний інтелект, edge computing) та роль кіберфізичних систем як технологічної основи сучасного виробництва.

Проведено порівняльний аналіз трьох рівнів цифрової репрезентації фізичного об’єкта: цифрової моделі, цифрової тіні та цифрового двійника. Встановлено, що цифровий двійник забезпечує двосторонній обмін даними та активне керування фізичним об’єктом, тоді як цифрова тінь реалізує лише односторонній потік телеметрії від фізичної системи до її цифрового представлення.

Обґрунтовано, що цифрова тінь є більш доцільним рішенням для задач моніторингу автоматизованих ліній складання, оскільки її односторонній

характер забезпечує високий рівень безпеки, мінімальні ризики втручання в реальний цикл управління та помірні вимоги до обчислювальних і мережевих ресурсів. Це дозволяє ефективно реалізовувати моніторинг cycle time, виявлення вузьких місць, аналіз аномалій та відтворення виробничих циклів.

Розглянуто багаторівневу архітектуру системи цифрової тіні, визначено функції кожного рівня (фізичний рівень, рівень збору даних, передача даних, обробка, зберігання та візуалізація) та проаналізовано сучасні інструменти їх реалізації (Node-RED, MQTT, InfluxDB, Grafana, CoppeliaSim). Показано, що багаторівнева архітектура забезпечує масштабованість, гнучкість і можливість подальшого розвитку системи в напрямку цифрового двійника.

Таким чином, проведений теоретичний аналіз підтверджує актуальність та доцільність застосування технології цифрової тіні для автоматизованих ліній складання в умовах Industry 4.0. Отримані результати створюють теоретичну та методологічну основу для подальшої розробки архітектури та практичної реалізації системи цифрової тіні, що буде розглянуто в наступних розділах дипломної роботи.

РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПРОЄКТУВАННЯ АРХІТЕКТУРИ ЦИФРОВОЇ ТІНІ ДЛЯ АВТОМАТИЗОВАНОЇ ЛІНІЇ СКЛАДАННЯ

2.1 Аналіз об’єкта автоматизації

2.1.1 Загальна характеристика автоматизованої лінії складання

Об’єктом, для якого проєктується система цифрової тіні, виступає автоматизована лінія складання дискретних виробів середнього розміру, побудована за модульним принципом з чітко визначеною послідовністю технологічних операцій. Такі лінії широко поширені у машинобудуванні, приладобудуванні та електроніці, де вимоги до повторюваності циклів, точності позиціонування та контролю якості мають критичне значення. Лінія складання розглядається як послідовність взаємопов’язаних робочих станцій, з’єднаних транспортною системою на базі конвеєра, які виконують операції подачі, фіксації, з’єднання, контролю та маркування компонентів виробу. Структура лінії передбачає поєднання механічних, електричних та інформаційних підсистем, що працюють синхронно під керуванням розподіленої системи контролерів.

З погляду цифровізації автоматизована лінія складання являє собою сукупність дискретних процесів, які генерують потоки телеметричних даних у режимі, наближеному до реального часу. Кожна станція оснащена сенсорами стану, лічильниками виробів, датчиками положення, температури та вібрації, а конвеєрні приводи мають окремі сенсори швидкості та споживання струму. Відповідно до концепції Industry 4.0 та архітектурної моделі ISA-95, така лінія належить до польового рівня (field level) автоматизації, що дозволяє коректно вписати її у багаторівневу архітектуру цифрової тіні, розглянуту в підрозділі 1.3.2.

Для подальшого проєктування ухвалено базовий варіант лінії, що включає шість послідовних станцій: завантаження заготовок, попереднє позиціонування, основну збірну операцію за участю промислового маніпулятора, проміжний візуальний контроль засобами машинного зору, фінальне з’єднання та станцію

маркування й вивантаження. Такий склад відповідає поширеним конфігураціям дискретних збірних ліній і дозволяє продемонструвати весь спектр функцій цифрової тіні без надмірного ускладнення моделі. Опис компонентів лінії наведено в підрозділі 1.3.3, а їхня візуалізація передбачена у Додатку А.

Таблиця 2.1 – Базовий склад станцій автоматизованої лінії складання

№	Назва станції	Тип операції	Основні сенсори
1	Завантаження заготовок	Подача матеріалу	Лічильник, оптичний
2	Попереднє позиціонування	Орієнтація деталі	Датчик положення
3	Основна збірна операція	Маніпулятор 6 DoF	Енкодери, струм
4	Візуальний контроль	Машинний зір	Камера, освітлення
5	Фінальне з’єднання	Кріплення	Датчик моменту
6	Маркування та вивантаження	Лазерне маркування	Лічильник, температура

Джерело: складено автором за матеріалами [13; 17; 24].

Аналіз структури, наведеної в таблиці 2.1, показує, що шість обраних станцій формують технологічно завершений ланцюг від подачі заготовки до вивантаження готового виробу. Найбільшу інтенсивність генерації телеметричних даних формують станції 3 і 4, оскільки промисловий маніпулятор та модуль машинного зору працюють з високою частотою опитування сенсорів, а їхні відмови мають найбільший вплив на загальну продуктивність. Саме ці станції розглядаються як критичні точки моніторингу під час подальшого проєктування.

2.1.2 Опис технологічного процесу складання та структура станцій

Технологічний процес складання організований за принципом потокового виробництва, де кожна станція виконує суворо визначений набір операцій у межах загального такту лінії. Заготовки надходять на першу станцію за допомогою вібролоткового подавача, що формує впорядкований потік одиничних виробів. Друга станція забезпечує точне позиціонування деталі у фіксованому положенні, контрольованому датчиками положення та оптичним сенсором, після чого деталь передається транспортною системою на третю

станцію. Транспортування здійснюється стрічковим конвеєром із сервоприводом, що дозволяє регулювати швидкість руху залежно від поточного стану наступної станції.

Центральним елементом лінії є третя станція, де промисловий маніпулятор виконує основну збірну операцію. Після виконання збірної операції деталь надходить на станцію візуального контролю, потім - на станцію фінального з'єднання та маркування.

Для чіткого розуміння розподілу часу виконання операцій у симуляційній моделі в таблиці 2.2 наведено номінальні значення cycle time для кожної станції та їхню частку в загальному такті лінії.

Таблиця 2.2 – Параметри станцій симуляційної моделі автоматизованої лінії складання

№	**Назва станції**	**Номінальний cycle time, с**	**Частка в загальному такті, %**	**Основні параметри моніторингу**	**Критичність**
1	Завантаження заготовок	2,0	20 %	counter, presence sensor, position	Низька
2	Попереднє позиціонування	1,5	15 %	position, alignment sensors	Середня
3	Основна збірна операція (маніпулятор 6 DoF)	4,5	45 %	joint positions, motor current, temperature, torque	Висока
4	Візуальний контроль (машинний зір)	1,8	18 %	inspection result (OK/NOK), processing time	Висока
5	Фінальне з'єднання	3,0	30 %	torque, force, duration	Висока
6	Маркування та вивантаження	1,2	12 %	counter, marking quality, temperature	Низька
Разом	Загальний такт лінії	10,0	100 %	-	-

Джерело: розроблено автором.

Загальний такт лінії в симуляційній моделі становить 10,0 секунд на один виріб, що відповідає продуктивності 360 виробів за годину. Найбільш тривалими та критичними є операції на станціях 3 і 5, які разом займають 75 % часу циклу і визначають загальну продуктивність лінії.

На рисунку 2.1 продемонстровано загальну логіку руху виробу від першої станції до останньої й вказано перелік основних сенсорів, що генерують телеметричні дані. Наявність єдиного контролера PLC, який координує роботу всіх станцій, формує природне джерело даних для першого рівня архітектури цифрової тіні.

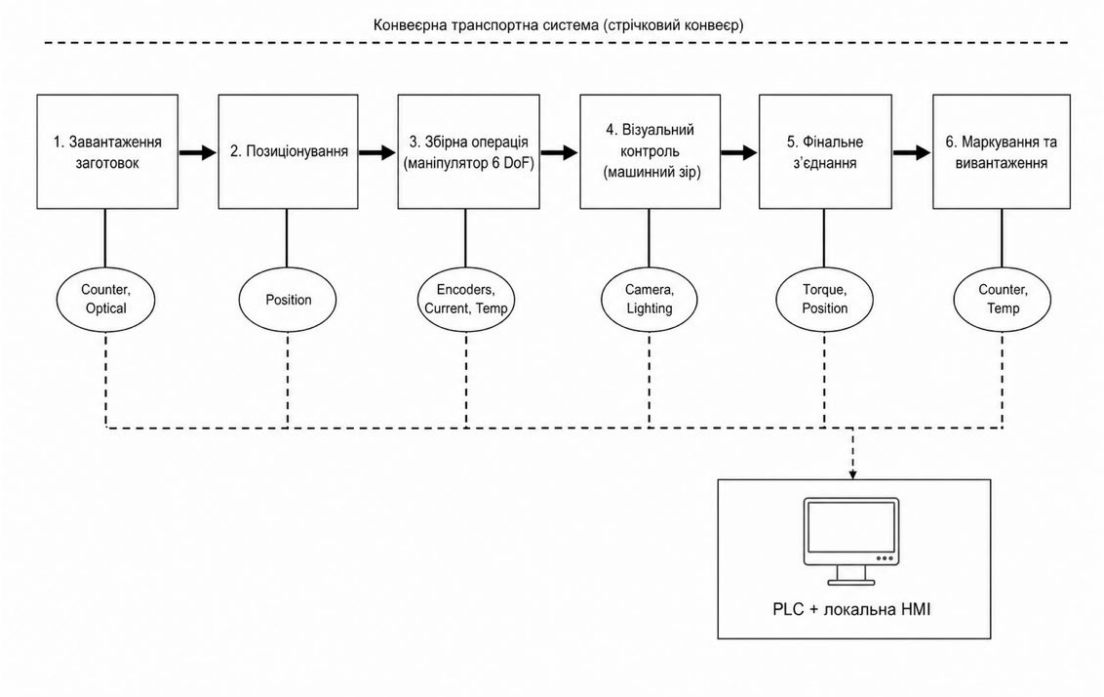


Рисунок 2.1 – Структурна схема технологічного процесу складання

Джерело: розроблено автором.

2.1.3 Виявлення критичних точок контролю та параметрів моніторингу

Виявлення критичних точок контролю є відправною точкою для формування переліку параметрів моніторингу, оскільки безпосередньо визначає, які саме дані повинні надходити з фізичного рівня до системи цифрової тіні. Критичними вважаються ті точки технологічного процесу, у яких відмова обладнання або відхилення параметрів від норми спричиняє максимальний вплив на загальну продуктивність лінії, якість продукції або безпеку роботи. Відповідно до методології ISO 23247, ці точки відображаються у системі цифрової тіні як спостережувані виробничі елементи (observable manufacturing elements), що мають власні цифрові ідентифікатори та набори характеристик.

Аналіз обраної лінії дозволив виокремити чотири категорії критичних точок: точки з високим ризиком зупинки (станція маніпулятора та конвеєр), точки з впливом на якість (станції візуального контролю та фінального з'єднання), точки з впливом на енергоспоживання (приводи всіх станцій) та точки безпеки (вхід і вихід зон роботи маніпулятора). Для кожної категорії визначено мінімально необхідний набір параметрів моніторингу. Таблиця 2.3 узагальнює результати такого аналізу.

Таблиця 2.3 – Критичні точки контролю та параметри моніторингу

Точка контролю	Категорія	Параметри моніторингу	Частота опитування
Маніпулятор станції 3	Ризик зупинки	Position, current, temperature	10 Гц
Конвеєр	Ризик зупинки	Speed, current	5 Гц
Станція 4 (зір)	Якість	Результат класифікації	За подією
Станція 5 (момент)	Якість	Torque, position	20 Гц
Усі приводи	Енергоспоживання	Current, voltage	1 Гц
Зони безпеки	Безпека	Status датчиків	За подією

Джерело: складено автором на основі [13; 24; 27].

Дані таблиці 2.3 формують основу подальшого проєктування рівня збору даних: загальна сумарна частота надходження повідомлень у пікові моменти сягає близько 80–100 повідомлень на секунду, що значно нижче гранично допустимих можливостей сучасних брокерів MQTT і підтверджує обґрунтованість вибору протоколу для подальшої передачі.

2.1.4 Аналіз показників продуктивності (cycle time, throughput, downtime)

Для оцінювання ефективності автоматизованої лінії складання використовуються три ключові показники, що формують основу подальшого аналізу та візуалізації у системі цифрової тіні. Перший показник, час циклу (cycle time), визначає тривалість обробки одного виробу на лінії від моменту початку першої операції до моменту виходу готового виробу. Цей показник є найбільш чутливим до зміни умов роботи окремих станцій і саме він виступає основою для виявлення вузьких місць. Другий показник, продуктивність (throughput), описує

кількість виробів, оброблених лінією за одиницю часу, та обчислюється як відношення кількості виробів до інтервалу спостереження. Третій показник, час простою (downtime), фіксує тривалість періодів, коли лінія не виконує корисну роботу через аварійну зупинку, очікування або переналадку.

Зв'язок між наведеними показниками описується формулою розрахунку загальної ефективності обладнання OEE (Overall Equipment Effectiveness), яка інтегрує доступність, продуктивність та якість. Формула (2.1) подає узагальнений вигляд цього показника:

$$\mathrm{OEE} = A \cdot P \cdot Q \tag{2.1}$$

де A – коефіцієнт доступності обладнання; P – коефіцієнт продуктивності; Q – коефіцієнт якості.

Коефіцієнт доступності розраховується через відношення фактичного часу роботи до планового, коефіцієнт продуктивності – через відношення фактичної кількості виробів до теоретично максимальної, а коефіцієнт якості – через частку придатних виробів у загальному обсязі. Така декомпозиція дозволяє системі цифрової тіні автоматично формувати інтегральну оцінку стану лінії на основі первинних телеметричних даних. Для обраної лінії складання базові значення показників прийнято на рівні: A = 0,92, P = 0,88, Q = 0,97, що відповідає типовому рівню автоматизованих ліній середньої складності за класифікацією Японської асоціації обслуговування заводів (JIPM) [33]. Розрахунок підсумкового значення наведено у формулі (2.2):

$$\mathrm{OEE} = 0{,}92 \cdot 0{,}88 \cdot 0{,}97 \approx 0{,}785 \tag{2.2}$$

де $\mathrm{OEE} \in [0; 1]$ – інтегральний показник загальної ефективності обладнання.

Отриманий показник OEE приблизно 0,785, або 78,5 відсотка, відповідає категорії «прийнятна ефективність» за класифікацією Японської асоціації

обслуговування заводів [33] і визначає мету подальшого моніторингу – постійне відстеження кожного з компонентів OEE з метою виявлення відхилень і поступового підвищення показника. Саме ці три первинні коефіцієнти стають основними обчислюваними метриками рівня візуалізації, що описаний у підрозділах 3.6.2 та 3.6.3, а перевірка коректності їх обчислення на даних симуляції наведена у підрозділі 3.7.5.

2.2 Формування вимог до системи цифрової тіні

2.2.1 Функціональні вимоги до системи

Функціональні вимоги визначають перелік дій, які система цифрової тіні повинна виконувати в межах своєї цільової області, та сформовані на основі попереднього аналізу об’єкта автоматизації. Першою функціональною вимогою є забезпечення безперервного збору телеметричних даних із усіх станцій лінії складання у режимі, наближеному до реального часу. Друга вимога стосується надійної передачі зібраних даних через мережу до центрального обчислювального вузла з мінімальною затримкою та збереженням послідовності повідомлень. Третя вимога передбачає проміжну обробку потокових даних з можливістю фільтрації, агрегації та виявлення простих аномалій без значних обчислювальних витрат.

Четверта функціональна вимога визначає необхідність довгострокового зберігання телеметрії у спеціалізованій базі даних часових рядів, що забезпечує ретроспективний аналіз і відтворення минулих циклів. П’ята вимога стосується побудови інтерактивних панелей моніторингу з відображенням ключових показників ефективності, поточного стану обладнання та активних попереджень. Шоста вимога передбачає реалізацію системи сповіщень з можливістю налаштування порогових значень для критичних параметрів і автоматичним надсиланням повідомлень оператору. Перелік функціональних вимог узагальнено у таблиці 2.4.

Таблиця 2.4 – Функціональні вимоги до системи цифрової тіні

№	Функціональна вимога	Пріоритет
FR-1	Збір телеметрії з фізичного рівня в режимі, наближеному до реального часу	Високий
FR-2	Передача даних до центрального вузла за моделлю publish–subscribe	Високий
FR-3	Проміжна обробка потокових даних (фільтрація, агрегація)	Середній
FR-4	Довгострокове зберігання у БД часових рядів	Високий
FR-5	Побудова інтерактивних дашбордів моніторингу	Високий
FR-6	Система сповіщень із порогами	Середній
FR-7	Можливість відтворення минулих циклів	Середній
FR-8	Інтеграція 3D-моделі для віртуального представлення	Низький

Джерело: складено автором.

Розподіл пріоритетів у таблиці 2.4 відображає логіку поступового впровадження: спочатку реалізуються вимоги високого пріоритету, що формують мінімально життєздатну версію цифрової тіні, після чого додаються можливості середнього та низького пріоритету. Така стратегія дозволяє отримати робочу систему вже на ранніх етапах і поступово розширювати її функціонал.

2.2.2 Нефункціональні вимоги (продуктивність, надійність, безпека)

Нефункціональні вимоги визначають якісні характеристики системи цифрової тіні та сформульовані з огляду на специфіку промислової експлуатації. За параметром продуктивності встановлено, що система має забезпечувати обробку до двохсот повідомлень на секунду без накопичення черги, а затримка від моменту генерації даних до їх відображення на дашборді не повинна перевищувати п’яти секунд для звичайних параметрів і однієї секунди для аварійних подій. Вимога до надійності передбачає безперервну роботу системи протягом не менше двадцяти чотирьох годин у робочому циклі та автоматичне

відновлення після короткочасних мережевих збоїв через механізми повторної доставки повідомлень.

Безпекові вимоги ґрунтуються на принципах захисту OPC UA та MQTT 5.0, що передбачають автентифікацію клієнтів, шифрування трафіку на основі TLS та контроль доступу за принципом мінімальних привілеїв. Доступ до бази даних обмежується окремим обліковим записом з правом лише запису для модуля обробки та правом лише читання для системи візуалізації, що зменшує ризик несанкціонованого втручання. Окремо передбачено ведення журналу подій для всіх дій оператора у середовищі Grafana, що відповідає принципам аудиту, описаним у ISO/IEC 27001.

2.2.3 Вимоги до якості та часу обміну даними (latency, bandwidth, QoS)

Вимоги до якості обміну даними сформульовані у термінах трьох ключових характеристик: затримки (latency), пропускної здатності (bandwidth) та рівня обслуговування повідомлень (QoS). Для звичайних телеметричних даних встановлено допустиму односторонню затримку до двохсот мілісекунд, що відповідає категорії near-real-time згідно з рекомендаціями ISO 23247-4. Аварійні повідомлення мають доставлятися з затримкою не більше п’ятдесяти мілісекунд та з гарантованою доставкою, що досягається використанням QoS рівня два за специфікацією MQTT 5.0.

Базовою (первинною) вимогою щодо пропускної здатності системи цифрової тіні встановлено пікову частоту повідомлень $N = 200$ msg/s, що відповідає сумарній телеметрії шести робочих станцій і маніпулятора з шістьма ступенями свободи при робочому такті лінії в межах від восьми до дванадцяти секунд. Це єдине вихідне значення є відправною точкою для всіх подальших оцінок: розрахунку необхідної смуги каналу (формула 2.3), обґрунтування вибору рівня QoS, оцінювання запасу за пропускною здатністю та побудови сценаріїв навантажувального тестування у підрозділі 3.7.4. При середньому розмірі повідомлення близько ста двадцяти байтів у форматі JSON підсумкова оцінка трафіку наведена у формулі (2.3):

$$B = N \cdot S = 200 \cdot 120 = 24\,000 \text{ байт/с} \approx 24 \text{ КБ/с} \qquad (2.3)$$

де B – пропускна здатність каналу; N – частота повідомлень; S – середній розмір повідомлення.

Отримане значення приблизно двадцять чотири кілобайти на секунду перебуває значно нижче типової пропускної здатності промислової мережі Ethernet навіть стандарту Fast Ethernet (100 Мбіт/с), що підтверджує мережеву реалізованість системи за базовою вимогою N = 200 msg/s. Запас за смугою каналу свідомо передбачено з ураху-ванням прогнозованого розширення навантаження до 4× від базового значення (тобто приблизно 800 msg/s ≈ 96 КБ/с), що відповідає сценарію одночасного підключення чотирьох ліній складання до спільної системи цифрової тіні. Цей запас буде верифікований експериментально у підрозділі 3.7.4 (сценарій 5).

2.2.4 Вимоги до візуалізації та інтерфейсу оператора

Вимоги до візуалізації спрямовані на забезпечення зручного та інформативного інтерфейсу оператора, що відповідає принципам ергономіки SCADA-систем. Головна панель моніторингу повинна одночасно відображати поточний час циклу, продуктивність, кількість оброблених виробів від початку зміни та статус кожної станції з кольоровим індикатором стану. Деталізовані панелі повинні містити графіки часових рядів для всіх ключових параметрів з можливістю масштабування часового вікна від останніх п’яти хвилин до останньої доби. Для аварійних подій передбачено окрему панель з хронологічним переліком і фільтрацією за рівнем критичності.

Усі панелі повинні бути доступні через веб-браузер без необхідності встановлення додаткового клієнтського програмного забезпечення, що забезпечує гнучкість використання з різних робочих місць. Передбачено підтримку мобільних пристроїв з адаптивним розташуванням елементів, що дозволяє оператору контролювати стан лінії під час обходу цеху. Опис

конкретних дашбордів та їхньої структури наведено у підрозділах 3.6.2 і 3.6.3, а скріншоти готових панелей розміщено в Додатку В.

2.2.5 Матриця простежуваності вимог

Для забезпечення прозорості та контролю відповідності між сформульованими вимогами, обраними технологічними рішеннями та подальшою реалізацією розроблено матрицю простежуваності вимог (Traceability Matrix). Матриця відображає зв'язок функціональних (FR) та нефункціональних (NFR) вимог з компонентами реалізації, тестовими сценаріями та отриманими результатами тестування.

Таблиця 2.5 – Матриця простежуваності вимог до системи цифрової тіні

ID	Тип	Опис вимоги (коротко)	Компонент реалізації	Тестовий сценарій	Результат тестування	Статус
FR-01	FR	Збір телеметрії від усіх станцій лінії в реальному часі	CoppeliaSim + MQTT Publisher (mqtt_bridge.py)	1, 5	~200 msg/s (до 800 msg/s)	Виконано
FR-02	FR	Надійна передача даних з підтримкою QoS	Eclipse Mosquitto + MQTT	1, 5	Втрати пакетів 0% при QoS 1	Виконано
FR-03	FR	Фільтрація, агрегація та нормалізація даних	Node-RED flows	1, 4	Успішна обробка всіх типів даних	Виконано
FR-04	FR	Довгострокове зберігання часових рядів	InfluxDB (2 buckets + downsampling)	1	Дані зберігаються коректно	Виконано
FR-05	FR	Візуалізація KPI (cycle time, throughput, OEE)	Grafana дашборди	1	Оновлення кожні 5 секунд	Виконано
FR-06	FR	Виявлення аномалій та система сповіщень	Node-RED + Grafana Alerts	2, 3, 4	Виявлення за 7–13 секунд	Виконано
FR-07	FR	Автоматичний розрахунок показника OEE	Grafana (Flux) + InfluxDB	1	OEE = 78,5% (повна відповідність)	Виконано
NFR-01	NFR	Медіанна end-to-end затримка ≤ 2,0 с	Весь стек	1–5	0,85 с	Виконано
NFR-02	NFR	Масштабованість (≥ 200 msg/s, тест ≥ 4 лінії)	Mosquitto + Node-RED + InfluxDB	5	800 msg/s (4 лінії)	Виконано

Продовження таблиці 2.5

NFR-03	NFR	Втрати пакетів ≤ 0,1%	MQTT QoS 1	1, 5	0% втрат	Виконано
NFR-04	NFR	Доступність дашбордів ≥ 99,5%	Docker + Grafana	1–5	100% (в умовах симуляції)	Виконано

Джерело: розроблено автором на основі даних власного дослідження.

Матриця простежуваності вимог забезпечує чіткий зв'язок між сформульованими функціональними та нефункціональними вимогами, обраними компонентами технологічного стеку та проведеним тестуванням системи. Вона буде використана в розділі 3 для підтвердження відповідності реалізації поставленим завданням.

2.3 Обґрунтування вибору технологічного стеку

2.3.1 Обґрунтування вибору середовища симуляції CoppeliaSim

Для побудови симуляційної моделі автоматизованої лінії складання обрано середовище CoppeliaSim, що розробляється компанією Coppelia Robotics. Цей інструмент є наступником V-REP і має тривалу історію використання у промислових, освітніх та дослідницьких проєктах [27]. CoppeliaSim побудоване на принципі розподіленого керування об'єктами, де кожна модель може мати власний вбудований скрипт мовою Lua або Python, що забезпечує високу гнучкість і реалістичність моделювання поведінки робототехнічних систем. Згідно з документацією Coppelia Robotics, остання стабільна версія програми (4.10) підтримує безпосередню інтеграцію з Python-бібліотеками, що значно спрощує побудову зовнішніх інтерфейсів для передачі телеметрії [27].

Серед альтернативних рішень розглядалися Webots, Gazebo та MATLAB/Simscape. Webots орієнтований переважно на освітнє моделювання та має обмежені можливості розширення сторонніми протоколами. Gazebo тісно пов'язаний з екосистемою ROS і потребує значних ресурсів для роботи у поєднанні з повноцінними промисловими сценаріями. MATLAB/Simscape – комерційне рішення з високою вартістю ліцензії, що не відповідає вимогам відкритості та доступності, поставленим перед цим проєктом. Натомість

CoppeliaSim має академічну ліцензію для безкоштовного некомерційного використання, що робить його оптимальним вибором для дипломної роботи.

2.3.2 Обґрунтування вибору протоколу MQTT та брокера повідомлень

Як основний протокол передачі даних обрано MQTT версії 5.0, специфікація якого затверджена консорціумом OASIS у 2019 році як офіційний стандарт [22]. Протокол реалізує модель publish–subscribe з трьома рівнями QoS, мінімальним накладними витратами на передачу заголовків та підтримкою механізмів збереження сесій. Згідно з офіційною специфікацією, MQTT був первісно спроєктований для роботи в обмежених мережевих умовах з малою пропускною здатністю та нестабільним зв'язком, що повністю відповідає потенційним сценаріям промислової експлуатації [22].

Для розгортання брокера повідомлень обрано Eclipse Mosquitto – відкрите програмне забезпечення під ліцензією Eclipse Public License, що розробляється у межах Eclipse Foundation. Mosquitto має невеликий розмір виконуваного файла, низькі вимоги до ресурсів і добре документовані механізми налаштування автентифікації та контролю доступу. На відміну від комерційних альтернатив, таких як HiveMQ Enterprise або IBM MQ, Mosquitto не накладає обмежень на кількість підключень у безкоштовній версії, що важливо для масштабованості системи цифрової тіні.

Порівняльний аналіз основних брокерів MQTT наведено у таблиці 2.6. Він базується на даних офіційних сайтів проєктів та незалежних оглядів у академічній літературі [22; 24].

Таблиця 2.6 – Порівняння брокерів MQTT для системи цифрової тіні

Брокер	Ліцензія	Підтримка MQTT 5.0	Кластеризація
Eclipse Mosquitto	EPL 2.0	Так	Bridging
HiveMQ CE	Apache 2.0	Так	Так (Enterprise)
EMQX	Apache 2.0	Так	Так
VerneMQ	Apache 2.0	Так	Так
RabbitMQ (MQTT plugin)	MPL 2.0	Частково	Так

Джерело: складено автором на основі офіційних сайтів проєктів.

Аналіз даних таблиці 2.6 показує, що Eclipse Mosquitto забезпечує мінімальний поріг входу та достатній функціонал для типових сценаріїв цифрової тіні, тоді як EMQX і HiveMQ є кращими варіантами у разі потреби в горизонтальній масштабованості. Для цієї роботи обрано Mosquitto як оптимальне рішення з огляду на компактність і простоту розгортання.

2.3.3 Обґрунтування вибору Node-RED для обробки потокових даних

Для організації проміжного шару обробки даних обрано Node-RED – платформу flow-based programming, розроблену та підтримувану OpenJS Foundation. Node-RED дозволяє створювати потоки обробки даних шляхом візуального з'єднання готових вузлів (nodes), що значно пришвидшує розробку та робить її доступною без глибокого програмування мовою JavaScript. Платформа має офіційні вузли для роботи з MQTT, InfluxDB, файлами, HTTP-запитами та функціями власної логіки, що повністю покриває потреби промислової обробки телеметрії [16].

Альтернативними варіантами для рівня обробки розглядалися Apache NiFi, Apache Kafka Streams та власна реалізація на Python з використанням бібліотеки paho-mqtt. Apache NiFi орієнтований на великі обсяги даних і вимагає значних ресурсів, що є надмірним для лінії середньої складності. Kafka Streams потребує розгортання повноцінного кластера Apache Kafka, що ускладнює архітектуру. Власна реалізація на Python потребує тривалої розробки та супроводу, що знижує оперативність впровадження. Node-RED натомість поєднує простоту візуального конструктора з можливістю використання JavaScript-функцій для нестандартної логіки.

2.3.4 Обґрунтування вибору InfluxDB для зберігання часових рядів

Як систему зберігання часових рядів обрано InfluxDB версії 2.x, розроблену компанією InfluxData. InfluxDB є спеціалізованою базою даних, оптимізованою для роботи з даними, прив'язаними до мітки часу, та широко використовується у промислових IoT-системах. Згідно з матеріалами InfluxData, InfluxDB використовує концепцію вимірювань (measurements), полів зі

значеннями (fields) та категоризуючих міток (tags), що дозволяє ефективно структурувати телеметричні дані [16]. База даних підтримує мову запитів Flux, що забезпечує гнучкі агрегації, перетворення та трансформації даних у реальному часі.

Альтернативними варіантами для рівня зберігання розглядалися TimescaleDB, Prometheus та класична реляційна база PostgreSQL. TimescaleDB є розширенням PostgreSQL і має сильні позиції в SQL-екосистемі, проте поступається InfluxDB у спеціалізованих функціях стиснення даних. Prometheus орієнтований переважно на моніторинг IT-інфраструктури та має обмеження щодо тривалості зберігання даних. PostgreSQL у чистому вигляді не оптимізований для часових рядів і потребує додаткових індексів та секцій для забезпечення прийнятної продуктивності.

2.3.5 Обґрунтування вибору Grafana для рівня візуалізації

Для рівня візуалізації обрано Grafana – відкриту платформу аналітики та моніторингу, розроблену Grafana Labs під ліцензією AGPL. Grafana має широкі можливості з побудови інтерактивних дашбордів, підтримує понад тридцять різних джерел даних, серед яких InfluxDB є одним із найкраще інтегрованих. Платформа надає набір готових візуальних компонентів, серед яких лінійні графіки, графіки розподілу, теплові карти, індикатори типу gauge, таблиці та статус-панелі, що дозволяє побудувати повноцінну систему моніторингу без розробки власного фронтенду.

Окремою перевагою Grafana є вбудована система сповіщень з можливістю надсилання повідомлень через електронну пошту, Slack, Telegram, Microsoft Teams та інші канали. Сповіщення налаштовуються через візуальний інтерфейс і ґрунтуються на запитах до бази даних, що інтегрує візуалізацію та аналітику в єдиний шар. Альтернативами розглядалися Kibana та Apache Superset; Kibana міцно пов’язана з Elasticsearch і не має нативної інтеграції з InfluxDB, а Apache Superset зорієнтований переважно на бізнес-аналітику з відомими наборами даних, а не на потоковий моніторинг промислового обладнання.

2.3.6 Порівняльний аналіз альтернативних технологій

Для обґрунтування остаточного вибору технологічного стеку проведено порівняння обраних рішень з основними альтернативами, які часто використовуються в промислових IoT- та Industry 4.0 проєктах. Результати порівняння наведено в таблиці 2.7.

Таблиця 2.7 – Порівняльний аналіз альтернативних технологій

Компонент	**Обране рішення**	**Альтернативи**	**Переваги обраного рішення**	**Недоліки альтернатив**
Середовище симуляції	CoppeliaSim	Gazebo, Webots, Siemens Plant Simulation	Безкоштовний, потужний Lua API, зручний для робототехніки	Gazebo - складніший у налаштуванні
Протокол передачі даних	MQTT (Mosquitto)	OPC UA, Kafka, AMQP	Легкий, низьке навантаження, ідеальний для IoT	OPC UA - важчий, Kafka - надлишковий
Обробка потокових даних	Node-RED	Apache NiFi, Apache Storm, Python	Візуальне програмування, швидка розробка	NiFi - складніший для невеликих проєктів
База даних часових рядів	InfluxDB	TimescaleDB, Prometheus, Cassandra	Висока швидкість запису, Flux query, open-source	Prometheus - більше для метрик, ніж телеметрії
Візуалізація	Grafana	Ignition SCADA, Power BI, Kibana	Відмінна інтеграція з InfluxDB, open-source, alerts	Ignition - платний, Power BI - менш гнучкий для IoT

Джерело: складено автором.

Вибір саме запропонованого стеку (CoppeliaSim + Mosquitto + Node-RED + InfluxDB + Grafana) обумовлений оптимальним співвідношенням функціональності, простоти розгортання, відкритості коду та відповідністю завданням моніторингу цифрової тіні.

2.4 Проєктування архітектури системи цифрової тіні

2.4.1 Загальна структурна схема системи

Загальна архітектура системи цифрової тіні побудована як п’ятирівнева ієрархія, що відповідає теоретичній моделі, описаній у підрозділі 1.3.2, та реалізує принципи стандарту ISO 23247 щодо референсної архітектури цифрового двійника у виробництві [9; 10]. Нижній рівень утворює симуляційна

модель лінії складання у CoppeliaSim, що генерує телеметричні дані; над ним розташований шар комунікацій на базі брокера Mosquitto; далі йде шар обробки на основі Node-RED; четвертий шар забезпечується базою даних InfluxDB; верхній рівень займає платформа візуалізації Grafana. Всі рівні з'єднані стандартизованими інтерфейсами, що дозволяє замінювати окремі компоненти без зміни загальної архітектури.

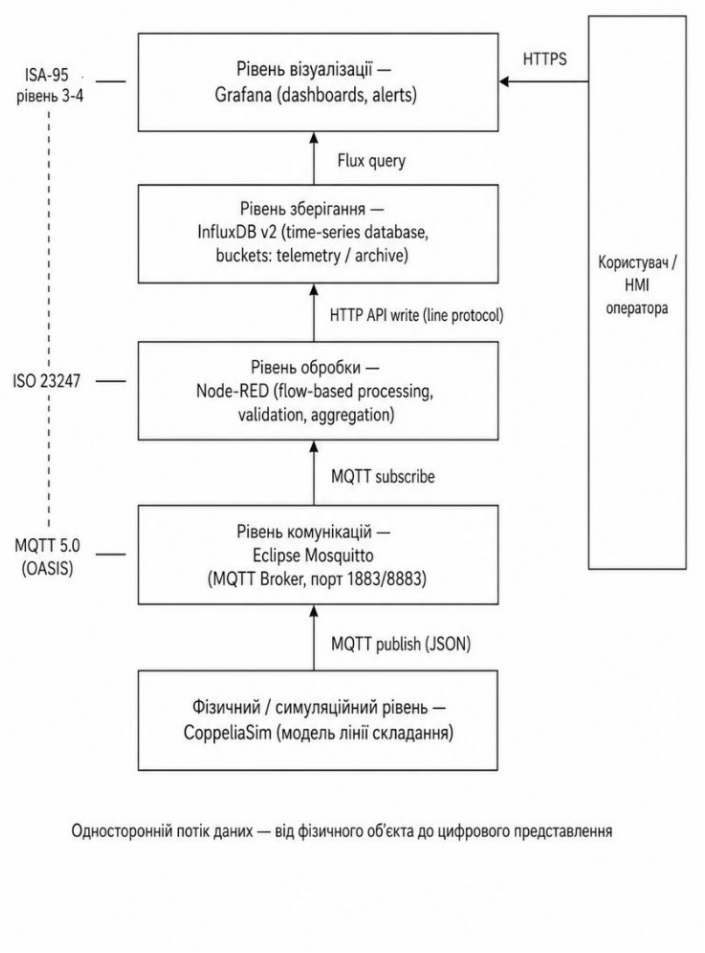


Рисунок 2.2 – Загальна структурна схема системи цифрової тіні

Джерело: розроблено автором за моделлю ISO 23247 [9].

На рисунку 2.2 показано вертикальну послідовність потоку даних, що рухаються знизу вгору від моменту їх генерації до візуального представлення. Стрілки на схемі позначають напрямок одностороннього потоку, що відповідає визначенню цифрової тіні згідно з [3], і виключають можливість зворотного впливу системи на фізичний рівень.

2.4.2 Проєктування фізичного рівня та симуляційної моделі

Проєктування фізичного рівня здійснюється у середовищі CoppeliaSim шляхом побудови тривимірної моделі лінії складання, що включає шість робочих станцій, ділянки конвеєра між ними та промислового маніпулятора з шістьма ступенями свободи. Кожен елемент моделі має власний вбудований скрипт мовою Lua, який реалізує поведінку у межах загального циклу симуляції. Скрипти забезпечують періодичне опитування віртуальних сенсорів положення, швидкості та лічильників виробів, формують структуровані повідомлення та публікують їх до брокера MQTT через інтерфейс bridge-плагіна, описаного у Додатку А.

Особливу увагу при проєктуванні приділено забезпеченню реалістичної динаміки руху виробів конвеєрною стрічкою та коректній синхронізації роботи маніпулятора з тактом лінії. Для цього використано вбудовані фізичні рушії CoppeliaSim – Bullet або ODE – з налаштованою точністю інтегрування. Параметри симуляції встановлено таким чином, щоб один такт симуляції відповідав п’ятдесяти мілісекундам реального часу, що дозволяє досягти необхідної частоти оновлення сенсорів.

2.4.3 Проєктування рівня збору та передачі даних

Рівень збору та передачі даних будується навколо брокера Eclipse Mosquitto, який виконує роль централізованого хабу для всіх повідомлень у системі. Брокер розгортається у Docker-контейнері з налаштованою системою автентифікації за принципом username/password та обмеженням доступу до тем за допомогою механізму ACL. Усі публікації від CoppeliaSim надсилаються до брокера через TCP-з’єднання на стандартному порту 1883 (без TLS у локальному середовищі розробки) або 8883 (з TLS у промисловому розгортанні).

Структура тем (topics) MQTT організована за ієрархічним принципом, що відповідає рекомендаціям Sparkplug B Specification: на верхньому рівні розміщується ідентифікатор лінії, на наступному – номер станції, далі тип сенсора, останній рівень містить конкретний параметр. Така ієрархія дозволяє

підписникам гнучко обирати потрібні набори даних за допомогою wildcard-операторів плюс і решетка, передбачених специфікацією MQTT 5.0 [22]. Приклад побудови теми та структури повідомлення наведено у підрозділі 2.5.1.

2.4.4 Проєктування рівня обробки даних

Рівень обробки даних реалізується через серію Node-RED потоків (flows), кожен з яких відповідає за окрему задачу: фільтрацію аномальних значень, обчислення похідних метрик, агрегацію за часовими вікнами та виявлення подій. Потоки організовані таким чином, що вхідні дані з брокера MQTT надходять до spliter-вузла, який розподіляє повідомлення за типом параметра, після чого кожна гілка проходить власну послідовність трансформацій. Завершальним кроком кожної гілки є запис обробленого значення до InfluxDB через офіційний node-red-contrib-influxdb пакет.

Для забезпечення відмовостійкості потоків передбачено механізм буферизації повідомлень у внутрішніх чергах Node-RED з обмеженням часу очікування. У разі недоступності InfluxDB повідомлення тимчасово накопичуються в локальному файлі та автоматично переподаються після відновлення з’єднання. Логіка обробки помилок винесена в окремий subflow, що дозволяє повторно використовувати її в різних гілках.

2.4.5 Проєктування рівня зберігання та візуалізації

Рівень зберігання представлений організацією InfluxDB з двома основними buckets: «assembly_line_telemetry» для оперативних даних з періодом зберігання сім днів і «assembly_line_archive» для агрегованих даних з періодом зберігання двадцять чотири місяці. Така структура поєднує високу швидкість запиту до свіжих даних з можливістю довгострокового ретроспективного аналізу. Перехід даних з оперативного бакета до архівного організовано через task у InfluxDB, який щогодини виконує downsampling за допомогою функцій Flux.

Рівень візуалізації побудовано як набір з чотирьох дашбордів у Grafana: загальна панель моніторингу лінії, детальна панель кожної станції, панель аналізу OEE та панель аварійних подій. Дашборди використовують спільну

палітру кольорів для забезпечення консистентного користувацького досвіду та підтримують механізм time range picker, що дозволяє оператору швидко переходити між реальним часом і будь-яким історичним інтервалом. Перелік панелей з описом їхнього вмісту наведено у підрозділі 3.6 та Додатку В.

2.4.6 Проєктування інтеграційних інтерфейсів між компонентами

Інтеграція компонентів системи відбувається через стандартизовані інтерфейси, що мінімізує жорстку зв'язаність та дозволяє замінювати окремі елементи без зміни архітектури. CoppeliaSim публікує повідомлення до брокера через стандартний MQTT-клієнт; Node-RED підписується на теми за допомогою вбудованого вузла mqtt-in і записує дані до InfluxDB через node-red-contrib-influxdb, що використовує HTTP API InfluxDB версії 2; Grafana звертається до InfluxDB через офіційний плагін з підтримкою Flux. Усі інтерфейси базуються на загальноприйнятих промислових стандартах і не вимагають розробки власних протоколів.

2.5 Розробка моделі даних та інформаційних потоків

2.5.1 Структура повідомлень MQTT та схема topics

Для організації обміну повідомленнями розроблено схему тем MQTT, яка враховує особливості автоматизованої лінії складання та забезпечує гнучкість для подальшого масштабування. Шаблон теми має чотири рівні: ідентифікатор лінії, номер станції, тип сенсора та назва параметра. У базовому випадку шаблон має вигляд line/{line_id}/station/{station_id}/{sensor_type}/{parameter}. Така ієрархія дозволяє підписникам вибірково отримувати дані потрібного рівня деталізації за допомогою wildcard-операторів [22].

Тіло повідомлення формується у форматі JSON та містить чотири обов'язкові поля: timestamp у форматі ISO 8601, значення параметра, одиницю вимірювання та необов'язкове поле з метаданими. Приклад повідомлення з телеметрією маніпулятора станції 3 наведено у лістингу 2.1 (Додаток А). Перелік основних тем і параметрів узагальнено у таблиці 2.8.

Таблиця 2.8 – Структура тем MQTT для цифрової тіні

Тема	Параметр	Тип значення	QoS
line/L1/station/3/manipulator/position	Кутове положення суглобів	JSON [6 float]	1
line/L1/station/3/manipulator/current	Струм двигунів	JSON [6 float]	1
line/L1/station/3/manipulator/temperature	Температура корпусу	float	0
line/L1/conveyor/speed	Швидкість конвеєра	float	0
line/L1/station/4/vision/result	Результат класифікації	string	2
line/L1/station/+/counter/value	Кількість виробів	integer	1
line/L1/+/events/error	Аварійні події	JSON	2

Джерело: розроблено автором на основі специфікації MQTT 5.0 [22].

Дані таблиці 2.8 демонструють, що для критичних подій застосовується QoS рівня два з гарантованою доставкою, для важливої телеметрії – рівень один з не менш як одноразовою доставкою, а для частих оновлень типу температури – рівень нуль, що допускає втрату окремих повідомлень, але забезпечує мінімальне навантаження на мережу.

Для підвищення надійності MQTT-обміну в системі цифрової тіні додатково передбачено використання механізмів retained messages, persistent sessions та автоматичного повторного підключення клієнтів (reconnect). Retained messages дозволяють новим підписникам одразу отримувати останній актуальний стан виробничої лінії після підключення до брокера. Persistent sessions забезпечують збереження підписок і накопичених повідомлень під час короткочасних розривів зв’язку. У разі втрати з’єднання MQTT-клієнти автоматично виконують повторне підключення до брокера, після чого обмін даними продовжується без повторного налаштування topics. Поєднання механізмів QoS, підтвердження доставки повідомлень та повторної передачі пакетів дозволяє мінімізувати втрати даних і забезпечити стабільну роботу системи моніторингу навіть за наявності тимчасових мережевих збоїв.

2.5.2 Схема даних InfluxDB (measurements, fields, tags)

Схема даних InfluxDB побудована з урахуванням рекомендованих практик роботи з часовими рядами та принципів структурування за parts of speech, запропонованих InfluxData. Кожна тема MQTT відображається у відповідне

вимірювання (measurement) бази даних. Tags використовуються для категоризуючих атрибутів, які беруть участь у фільтрації запитів, а fields – для безпосередніх числових значень параметрів. Така структура забезпечує ефективну індексацію та швидкий доступ до даних.

Основними вимірюваннями у схемі є manipulator_telemetry, conveyor_telemetry, vision_results, energy_metrics та production_counters. Як tags виступають line_id, station_id, sensor_type, що дозволяє виконувати агрегацію по будь-якому з цих вимірів. Перелік основних вимірювань та їхньої структури узагальнено у таблиці 2.9.

Таблиця 2.9 – Схема даних InfluxDB для цифрової тіні

Measurement	Tags	Fields	Частота запису
manipulator_telemetry	line_id, station_id	joint_1–6, current, temp	10 Гц
conveyor_telemetry	line_id	speed, current	5 Гц
vision_results	line_id, station_id	result, confidence	За подією
energy_metrics	line_id, station_id	voltage, current, power	1 Гц
production_counters	line_id, station_id	count, defects	За подією

Джерело: розроблено автором.

Структура таблиці 2.9 підтверджує однозначну відповідність між темами MQTT і вимірюваннями InfluxDB, що спрощує налагодження та діагностику потоку даних. Сумарна кількість записів на добу за робочих умов оцінюється на рівні близько п’яти мільйонів, що для сучасних обчислювальних потужностей є помірним обсягом.

2.5.3 Діаграма потоків даних (DFD) системи цифрової тіні

Для формального опису інформаційних потоків розроблено діаграму потоків даних (DFD), що відображає шлях кожного типу телеметричної інформації від моменту її генерації у CoppeliaSim до моменту відображення оператору. На DFD виокремлено зовнішні сутності, процеси, сховища даних та потоки, що дозволяє наочно простежити трансформації даних на кожному кроці. Деталізована DFD наведена на рисунку 2.3.

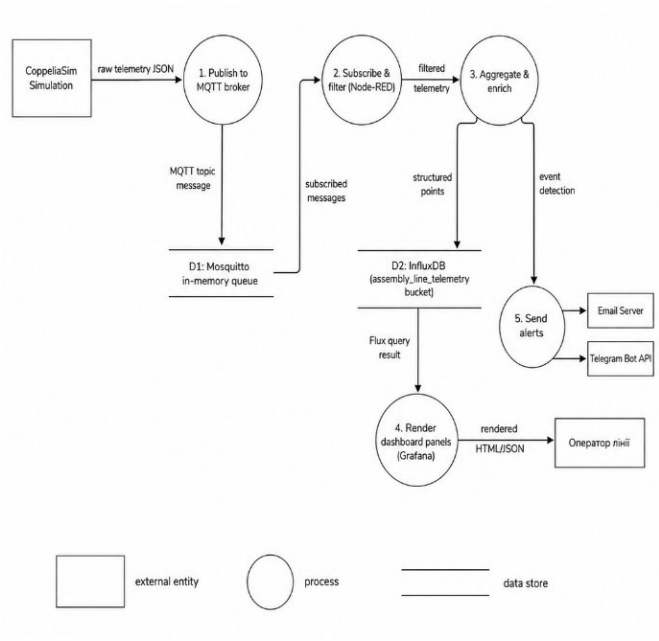


Рисунок 2.3 – Діаграма потоків даних системи цифрової тіні (DFD рівня 1)

Джерело: розроблено автором.

Аналіз рисунка 2.3 показує, що дані проходять п’ять основних трансформацій між моментом генерації та відображенням, причому всі сховища мають чітко визначені інтерфейси читання та запису. Така деталізація потоків формує основу для тестування системи та діагностики потенційних місць виникнення затримок.

Висновки до розділу 2

У розділі проведено системний аналіз об’єкта автоматизації, сформовано вимоги до системи цифрової тіні, обґрунтовано вибір технологічного стеку та спроєктовано загальну архітектуру системи з усіма необхідними інтерфейсами. Виокремлено критичні точки контролю автоматизованої лінії складання, серед яких найвищий пріоритет мають станція маніпулятора та конвеєр; обчислено базове значення показника OEE на рівні 78,5 відсотка, що відповідає категорії прийнятної ефективності.

Сформульовано вісім функціональних і три групи нефункціональних вимог до системи, з-поміж яких ключовими є вимоги до затримки доставки повідомлень не більше двохсот мілісекунд для звичайних даних і п’ятдесяти мілісекунд для аварійних подій. Підтверджено реалізованість системи на наявній

промисловій мережевій інфраструктурі з оцінкою необхідної пропускної здатності на рівні приблизно двадцять чотири кілобайти на секунду.

Обґрунтовано вибір технологічного стеку, до складу якого входять CoppeliaSim, Eclipse Mosquitto, Node-RED, InfluxDB та Grafana, кожний компонент якого використовується під відкритими ліцензіями та має тривалу історію промислової експлуатації. Розроблено структуру тем MQTT за чотирирівневим ієрархічним принципом, схему даних InfluxDB з п’ятьма основними вимірюваннями та діаграму потоків даних, що формалізує всі трансформації між компонентами. Отримана архітектура є основою для практичної реалізації, описаної в розділі 3.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ ЦИФРОВОЇ ТІНІ АВТОМАТИЗОВАНОЇ ЛІНІЇ СКЛАДАННЯ

3.1 Розгортання та налаштування програмного середовища

3.1.1 Встановлення та конфігурація CoppeliaSim

Симуляційне середовище CoppeliaSim розгортається на робочій станції під керуванням операційної системи Ubuntu 22.04 LTS, що відповідає офіційно підтримуваним конфігураціям, перерахованим у документації Coppelia Robotics. Інсталяційний пакет EDU edition завантажується з офіційного сайту проєкту, після чого розпаковується у власну директорію без потреби у системних правах. Запуск середовища виконується командою ./coppeliaSim.sh, що автоматично активує всі вбудовані плагіни, включно з MQTT-клієнтом, реалізованим у simRemoteApi та simZmqRemoteApi.

Для забезпечення відтворюваності конфігурації створено власний файл налаштувань user.cfg, у якому фіксуються параметри сцени за замовчуванням, крок симуляції dt = 50 мс, дозвіл на запуск зовнішніх Python-скриптів і шлях до директорії з користувацькими моделями. Аналогічно конфігурується пайплайн відеозапису симуляції, що використовується для подальшого створення демонстраційних матеріалів. Перелік виконаних кроків інсталяції та значення параметрів узагальнено у Додатку Г.

3.1.2 Встановлення та налаштування MQTT-брокера Eclipse Mosquitto

Брокер Eclipse Mosquitto розгорнуто у Docker-контейнері на основі офіційного образу eclipse-mosquitto. Контейнер запускається з прокинутим томом для зберігання журналів та файлу конфігурації mosquitto.conf. Базова конфігурація передбачає прослуховування порту 1883 для незахищеного локального трафіку та порту 8883 для TLS-з’єднань з використанням самопідписаного сертифіката. Параметр allow_anonymous встановлено у значення false, що змушує всіх клієнтів проходити процедуру автентифікації.

Облікові записи створюються командою mosquitto_passwd та зберігаються у файлі passwd, доступному лише для користувача mosquitto. Контроль доступу

до тем налаштовується у файлі acl.conf, де кожний обліковий запис отримує мінімально необхідні права. Наприклад, скрипт CoppeliaSim отримує право publish до тем line/L1/#, а Node-RED – право subscribe до тих самих тем. Така схема відповідає принципу мінімальних привілеїв і знижує ризик небажаного доступу до даних.

3.1.3 Встановлення та налаштування Node-RED

Node-RED встановлюється на ту саму робочу станцію через офіційний інсталятор за допомогою npm. Версія Node.js обрана 18 LTS як рекомендована розробниками платформи. Після встановлення створюється системний сервіс node-red.service, що забезпечує автоматичний запуск при перезавантаженні. Конфігураційний файл settings.js модифікується для додавання адміністративного пароля до інтерфейсу, активації CORS для домену Grafana та встановлення лімітів пам’яті потоків.

Окремим кроком встановлюються додаткові пакети вузлів: node-red-contrib-influxdb для запису у часові ряди, node-red-contrib-mqtt-broker для діагностики тем, node-red-node-email для надсилання сповіщень. Усі пакети встановлюються через вбудований Palette Manager, що дозволяє відстежувати версії та оновлення централізовано. Після завершення налаштування виконується тестовий потік для перевірки правильності підключення до брокера та запису у базу даних.

3.1.4 Встановлення та налаштування InfluxDB

InfluxDB версії 2.7 розгортається у Docker-контейнері з прокинутими томами для постійного збереження даних і конфігураційних файлів. Початкова ініціалізація бази даних виконується через CLI команду influx setup, у процесі якої створюється організація AssemblyLine, перший адміністративний користувач та основний bucket assembly_line_telemetry. Згенерований токен доступу зберігається у захищеному файлі та використовується для підключення з Node-RED і Grafana.

Окремо створюється другий bucket assembly_line_archive з тривалістю зберігання двадцять чотири місяці для довгострокових даних. Між цими двома bucket налаштовується задача downsampling, реалізована мовою запитів Flux. Задача виконується кожну годину та агрегує сирі дані до п’ятихвилинних середніх значень, що значно зменшує обсяг архівних даних без втрати корисної аналітичної інформації. Текст задачі Flux наведено в лістингу 3.1 у Додатку А.

3.1.5 Встановлення та налаштування Grafana

Grafana встановлюється у власному Docker-контейнері з прив’язкою до томів для постійного збереження конфігурації та плагінів. Версія 10.x обрана як остання стабільна гілка з повною підтримкою Flux. У файлі grafana.ini налаштовуються параметри автентифікації через локальних користувачів і за можливості – через корпоративний LDAP-сервер. Тимчасовий пароль адміністратора замінюється при першому вході відповідно до вимог безпеки.

Після успішного запуску Grafana виконується перше підключення джерела даних InfluxDB через стандартний плагін з підтримкою Flux. Налаштовуються тестові запити, що перевіряють доступність всіх п’яти основних вимірювань, описаних у підрозділі 2.5.2. Після успішного тестування імпортується базовий набір дашбордів з JSON-файлів, описаний у Додатку В.

3.2 Розробка симуляційної моделі автоматизованої лінії складання у CoppeliaSim

3.2.1 Створення 3D-моделі конвеєрної системи

Тривимірна модель конвеєрної системи будується на основі стандартних примітивів CoppeliaSim з подальшим уточненням геометрії під реальні пропорції промислового конвеєра. Стрічковий конвеєр представлено як прямокутну поверхню, що зв’язана з невидимою кінематичною віссю; рух стрічки моделюється шляхом анімації текстури вздовж осі та одночасного руху об’єктів, що перебувають на поверхні, з тією самою швидкістю. Такий підхід реалізує реалістичну поведінку конвеєра без значного навантаження на фізичний рушій.

Геометричні параметри конвеєра обрано виходячи з типових промислових значень: довжина однієї секції – два метри, ширина стрічки – триста міліметрів, висота розташування над підлогою – дев’ятсот міліметрів, що відповідає ергономічним вимогам. Між шістьма станціями розташовано п’ять секцій конвеєра, з’єднаних разом у єдину систему. Для забезпечення можливості незалежного керування кожна секція має власний віртуальний привод з налаштовуваною швидкістю.

3.2.2 Моделювання промислових роботів та робочих станцій

Промисловий маніпулятор станції 3 моделюється на основі типової кінематичної схеми вертикально-шарнірного робота з шістьма ступенями свободи, що відповідає поширеним моделям, таким як KUKA KR6 або ABB IRB 1200. Інверсна кінематика реалізується через вбудовану в CoppeliaSim бібліотеку Reflexxes IK, що дозволяє задавати траєкторії у декартовому просторі та автоматично обчислювати кутові положення суглобів. Захватний пристрій представлено як двопальцевий пневматичний грипер з керованою силою стиснення.

Інші станції моделюються у спрощеному вигляді як комбінації статичних і динамічних об’єктів, що виконують необхідні маніпуляції з виробом у потрібні моменти часу. Станція машинного зору представлена як куб з прикріпленою камерою-сенсором; станція моменту – як пневматичний ударний механізм з датчиком сили; станція маркування – як випромінювач зі змінним кольором, що імітує лазерну мітку. Усі станції додаються до моделі через інтерфейс drag-and-drop та конфігуруються через панель Object Properties.

3.2.3 Розробка скриптів керування циклом складання

Логіка керування циклом складання реалізується через головний дочірній скрипт сцени мовою Lua, який працює як кінцевий автомат із п’ятьма станами: ініціалізація, очікування заготовки, виконання операції, передача наступної станції, обслуговування. Перехід між станами відбувається на основі подій від

сенсорів та лічильників часу. Такий підхід забезпечує детермінованість поведінки моделі та її відповідність реальній логіці PLC промислової лінії.

Параметри циклу винесено у конфігураційну таблицю на початку скрипта, що дозволяє швидко змінювати такт лінії, послідовність операцій та пороги аварійних подій без модифікації основної логіки. Базовий такт лінії встановлено на рівні десять секунд на виріб, що відповідає середньому темпу виробництва трьохсот шістдесят виробів на годину. Повний код керуючого скрипта наведено у лістингу А.1 (Додаток А).

3.2.4 Реалізація віртуальних сенсорів та лічильників

Віртуальні сенсори реалізуються через комбінацію вбудованих об’єктів CoppeliaSim – proximity sensors, vision sensors, force sensors – та власних обчислюваних змінних, що оновлюються кожен крок симуляції. Сенсори положення суглобів маніпулятора повертають кутові значення з точністю до одного сантирадіана; сенсор швидкості конвеєра обчислюється через диференціювання положення; сенсор температури моделюється як емпірична функція струму та часу неперервної роботи.

Лічильники виробів реалізовано через підрахунок подій перетину контрольних площин у початковій та кінцевій точках лінії. Різниця між цими лічильниками визначає поточну кількість виробів на лінії, а добові значення – основну метрику продуктивності. Усі значення сенсорів експортуються до асоціативної таблиці sensor_state, яка зчитується з періодом сто мілісекунд окремою функцією-публікатором MQTT, описаною в наступному підрозділі.

3.3 Реалізація рівня збору та передачі даних

3.3.1 Розробка MQTT-publisher на базі CoppeliaSim

Публікатор MQTT реалізовано у вигляді окремого Python-модуля mqtt_bridge.py, що працює у одному процесі з CoppeliaSim через інтерфейс ZMQ Remote API. Модуль підключається до брокера Mosquitto за допомогою клієнтської бібліотеки paho-mqtt версії 1.6.1, виконує автентифікацію та підтримує стабільне з’єднання з автоматичним перепідключенням у разі

тимчасових збоїв. На кожному кроці симуляції модуль зчитує асоціативну таблицю sensor_state і формує окреме повідомлення для кожного параметра.

Перед публікацією повідомлення проходять етап серіалізації у формат JSON через стандартну бібліотеку json, з обов'язковим додаванням поля timestamp у форматі ISO 8601 з мікросекундною точністю. Це дозволяє відстежувати дрейф годинників між симуляційним і реальним часом та коректно встановлювати порядок повідомлень у InfluxDB. Повна реалізація публікатора наведена у лістингу А.2 (Додаток А).

3.3.2 Налаштування topics та формування JSON-повідомлень

Карта відповідності між сенсорами та темами MQTT реалізована у вигляді конфігураційного файла topic_map.json, що завантажується модулем при старті. Формат файла дозволяє вільно змінювати теми та структуру повідомлень без модифікації коду публікатора. Для кожного сенсора визначається тема, рівень QoS та шаблон JSON. Шаблон містить плейсхолдери, що заповнюються поточними значеннями з sensor_state.

Структура JSON-повідомлення суворо відповідає схемі, описаній у підрозділі 2.5.1, та містить чотири основних поля: timestamp, value, unit, metadata. Поле metadata використовується для передачі додаткової контекстної інформації, такої як одиниця виміру, ідентифікатор сесії симуляції чи специфічні параметри сенсора. Приклад згенерованого повідомлення наведено нижче для ілюстрації:

{ «timestamp»: «2025-11-09T10:42:17.412Z», «value»: [0.12, -1.34, 1.57, 0.0, 0.78, -0.45], «unit»: «rad», «metadata»: { «sensor»: «manipulator_joint_position», «station»: 3, «line»: «L1» } }

3.3.3 Тестування передачі даних та налаштування рівнів QoS

Тестування передачі даних виконується у три етапи: функціональне тестування, навантажувальне тестування та тестування відмовостійкості. На функціональному етапі перевіряється правильність формування повідомлень для всіх типів сенсорів та відповідність структури JSON-схемі. Перевірка виконується через підписку на всі теми line/L1/# за допомогою інструмента

mosquitto_sub з виводом у консоль. На цьому етапі виявлено та усунено п’ять невідповідностей форматування, переважно пов’язаних з округленням значень.

Навантажувальний етап моделює пікові режими роботи лінії з частотою публікації двісті повідомлень на секунду. Брокер Mosquitto показує стабільну роботу без втрати повідомлень при QoS рівня 1 і 2, час кругового рейсу (round-trip time) між публікацією та підпискою у локальній мережі становить близько п’яти мілісекунд. Тестування відмовостійкості виконується шляхом штучного розриву мережевого з’єднання на десять секунд, після чого модуль автоматично відновлює підключення та відправляє накопичені повідомлення з активацією властивості Clean Session = false.

3.4 Реалізація рівня обробки даних у Node-RED

3.4.1 Побудова потоку обробки даних (flow)

Основний flow Node-RED для обробки потокової телеметрії будується за принципом конвеєра, де кожний наступний вузол виконує одну атомарну функцію. Перший вузол mqtt-in підписується на всі теми line/L1/# з QoS рівня 1; другий вузол switch розподіляє повідомлення за типом параметра на окремі гілки; третя група вузлів function виконує валідацію вхідних даних з відсіюванням NaN-значень і викидів; четверта група вузлів delay реалізує згладжування шляхом усереднення значень за вікном двісті мілісекунд; завершальний вузол influxdb-out записує оброблені дані до бази.

Загальний flow налічує тридцять чотири вузли та підтримує паралельну обробку всіх каналів телеметрії. Експортована JSON-конфігурація flow обсягом близько шістнадцяти кілобайт зберігається у системі контролю версій разом з кодом скриптів. Скриншот основного flow та його повна конфігурація наведені у Додатку Б.

3.4.2 Реалізація фільтрації, агрегації та нормалізації

Фільтрація аномальних значень реалізована у function-вузлі через перевірку приналежності значень до допустимих діапазонів, визначених у конфігураційному файлі limits.json. Значення, що виходять за межі заданих

діапазонів, помічаються міткою is_anomaly та проходять окремий шлях обробки, що включає логування та потенційну активацію сповіщення. Цей механізм відповідає базовій реалізації виявлення подій, описаній у підрозділі 1.3.5.

Агрегація даних виконується у двох режимах: ковзне середнє з вікном один секунда для відображення згладжених трендів та групове усереднення за хвилину для збереження довгострокових даних. Нормалізація приводить значення різних параметрів до спільного безрозмірного діапазону [0, 1] з використанням мін-макс перетворення, що описане формулою (3.1):

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{3.1}$$

де x – поточне значення параметра; x_min, x_max – встановлені межі діапазону.

Нормалізовані значення використовуються переважно для розрахунку інтегральних індикаторів стану обладнання, тоді як у базу даних записуються як нормалізовані, так і вихідні значення для забезпечення повноти інформації.

3.4.3 Виявлення подій та простих аномалій у потоці даних

Виявлення подій та аномалій у системі цифрової тіні реалізовано на основі простих детермінованих правил (rule-based anomaly detection). Такий підхід обрано через його високу інтерпретованість, низькі обчислювальні вимоги та надійність у промислових умовах.

Виявлення подій здійснюється через два основних механізми:

1. Порогові правила (threshold-based rules) - застосовуються до критичних параметрів, таких як температура двигуна маніпулятора, швидкість конвеєра та час циклу. Для кожного параметра визначено два рівні:
    - попередження (warning);
    - критична подія (critical).

Приклад: температура маніпулятора станції 3 - попередження при перевищенні 65 °C, критична подія - при 75 °C.

2. Статистичне правило трьох сигм - значення параметра вважається аномальним, якщо воно виходить за межі інтервалу $[m-3\sigma, m+3\sigma]$, де m - ковзне середнє, а σ - ковзне стандартне відхилення за останню годину. Цей метод дозволяє виявляти не лише грубі відхилення, але й статистично нетипову поведінку параметра.

При спрацюванні правила формується подія з відповідним рівнем критичності (warning або critical), яка публікується в окрему MQTT-тему line/L1/events/error та записується до InfluxDB у вимірювання events. Це забезпечує швидке реагування системи та фіксацію інцидентів для подальшого аналізу.

Важливо зазначити, що в межах даної роботи реалізовані лише rule-based методи виявлення аномалій. Використання методів машинного навчання (ML) та алгоритмів прогнозного технічного обслуговування (predictive maintenance) планується як напрямок подальших досліджень.

3.4.4 Інтеграція Node-RED з InfluxDB для запису даних

Інтеграція Node-RED з InfluxDB реалізується через офіційний пакет node-red-contrib-influxdb версії 0.7.x, що підтримує InfluxDB версії 2 з Flux. Конфігурація з’єднання містить URL сервера, токен доступу, ім’я організації та назву бакета. Для оптимізації продуктивності налаштовано пакетну вставку: вузол накопичує до тисячі точок або чекає до п’яти секунд, після чого виконує один запит запису. Такий підхід значно знижує мережеве навантаження та підвищує загальну швидкодію.

Структура точок даних, що записуються до InfluxDB, формується у спеціальній функції-конструкторі, яка перетворює JSON-повідомлення MQTT у формат line protocol, очікуваний InfluxDB. Поля та теги розставляються відповідно до схеми, описаної у підрозділі 2.5.2. Перевірка коректності запису виконується через тестовий запит у вебінтерфейсі InfluxDB Data Explorer.

3.5 Реалізація рівня зберігання даних в InfluxDB

3.5.1 Створення bucket та налаштування retention policy

У реалізованому варіанті системи створено два основні бакети для зберігання різних типів даних. Бакет assembly_line_telemetry зберігає сирі телеметричні дані з періодом збереження сім діб; цей термін достатній для оперативного аналізу та діагностики поточних проблем без надмірного накопичення обсягу. Бакет assembly_line_archive призначений для довгострокового зберігання агрегованих даних з періодом двадцять чотири місяці, що дозволяє виконувати річний аналіз тенденцій та порівнювати показники за сезонами.

Окремо створено допоміжний бакет events_archive для зберігання журналу подій з періодом дванадцять місяців. Цей бакет містить інформацію про всі аварійні події, перевищення порогів та аномалії, виявлені на рівні обробки. Структура його даних відрізняється від основних бакетів і включає поля event_type, severity, source, description, time_resolved для повної реєстрації життєвого циклу подій.

3.5.2 Запис телеметрії та оптимізація продуктивності

Оптимізація продуктивності запису у InfluxDB виконується на двох рівнях: на рівні клієнта Node-RED – через пакетні запити, описані у підрозділі 3.4.4, та на рівні бази даних – через коректне налаштування shard-розмірів і використання теги-кардинальності. Кардинальність тегів навмисно обмежена: line_id, station_id та sensor_type мають фіксовані переліки значень, що запобігає експоненціальному зростанню індексу. Часові ряди групуються за тижневими shard-ами, що відповідає рекомендаціям InfluxData для бакетів з тривалістю зберігання понад сім діб.

Результати бенчмаркінгу показали, що швидкість запису у налаштованій конфігурації становить близько п’ятнадцяти тисяч точок на секунду за умов одночасної роботи дашбордів Grafana, що значно перевищує потреби системи цифрової тіні. Розмір індексів та даних після одного тижня безперервної роботи

симуляції становить близько чотирьохсот мегабайтів, що є прийнятним для типових промислових серверів.

3.6 Реалізація рівня візуалізації в Grafana

3.6.1 Підключення InfluxDB як джерела даних

Підключення InfluxDB як джерела даних у Grafana виконується через адміністративний інтерфейс у меню Data Sources. Як тип джерела обрано InfluxDB; як режим запитів – Flux, що відкриває повний спектр аналітичних можливостей мови. Параметри підключення містять URL http://influxdb:8086 (внутрішня адреса контейнера), назву організації AssemblyLine та токен доступу, згенерований на етапі ініціалізації InfluxDB. Тестове з'єднання підтверджує доступність бази даних та валідність токена.

3.6.2 Розробка дашбордів моніторингу cycle time та throughput

Дашборд моніторингу cycle time та throughput є основним інструментом оператора для оцінювання поточної продуктивності лінії. Дашборд складається з шести панелей: великий gauge поточного часу циклу зі шкалою від восьми до п'ятнадцяти секунд та зонами «зелений» – норма, «жовтий» – попередження, «червоний» – критичне відхилення; великий counter поточної кількості виробів від початку зміни; лінійний графік часу циклу за останню годину з зеленою лінією-цілю на восьми секундах; стовпчастий графік throughput по годинах за останню добу; теплова карта cycle time по станціях за останній тиждень; таблиця топ-п'яти найдовших циклів за зміну з посиланнями на відеозаписи.

Рисунок 3.1 – Макет дашборду моніторингу cycle time та throughput у Grafana

Джерело: розроблено автором у Grafana 10.

Дашборд налаштовано на автоматичне оновлення кожні п’ять секунд, що забезпечує відчуття реального часу. Запити Flux до InfluxDB оптимізовані за допомогою функції aggregateWindow з кроком, що адаптується до обраного часового діапазону. Це дозволяє ефективно відображати як п’ятихвилинні інтервали з високою деталізацією, так і місячні тренди без затримок.

3.6.3 Розробка дашбордів моніторингу стану обладнання

Дашборд моніторингу стану обладнання забезпечує детальний огляд кожної станції лінії та використовується переважно для діагностичних задач, що відповідає підходу комп’ютеризованого моніторингу і діагностики компонентів обладнання на основі цифрової тіні [36]. Структура дашборду включає окрему секцію для кожної станції з власним набором панелей: статус-індикатор поточного стану (працює, очікує, помилка, обслуговування), лінійний графік температури з кольоровим градієнтом, графік споживаного струму, лічильник напрацювання та лічильник кількості відмов. Для маніпулятора станції 3 додатково передбачено шість окремих графіків положення суглобів.

Дашборд підтримує механізм drill-down: при кліку на статус-індикатор станції відкривається окрема панель з детальним переліком останніх подій та аномалій. Це дозволяє оператору швидко переходити від загального огляду до конкретних інцидентів без перемикання між вкладками. Скріншот дашборду наведено у Додатку В на рисунку В.2.

3.6.4 Налаштування системи сповіщень (alerts)

Система сповіщень Grafana налаштована для автоматичного інформування оператора про критичні події. Створено десять правил сповіщення, що охоплюють основні сценарії відмов: перевищення температури маніпулятора, перевищення часу циклу, накопичення дефектів на станції візуального контролю, неочікувана зупинка конвеєра, втрата з’єднання з MQTT-брокером, заповнення дискового простору InfluxDB, перевищення затримки обробки, аномалії струму, аварійні стани окремих станцій та перевищення допустимого числа спрацьовувань захисних барєрів.

Усі сповіщення налаштовано на надсилання через два канали: електронну пошту операторської зміни та чат Telegram чергової зміни. Для кожного правила визначено мінімальний інтервал між повторними сповіщеннями (silence period) тривалістю десять хвилин, що запобігає накопиченню лавини однотипних повідомлень. Перевірка коректності роботи сповіщень виконувалась шляхом штучного моделювання аварійних ситуацій.

3.7 Тестування та оцінка ефективності системи цифрової тіні

3.7.1 Сценарії тестування системи

Для оцінки ефективності системи цифрової тіні розроблено п’ять основних сценаріїв тестування, що охоплюють різні аспекти роботи. Сценарій 1 – нормальна робота лінії протягом восьмигодинної зміни без штучних збурень, що використовується для перевірки базової функціональності. Сценарій 2 – моделювання поступового підвищення температури маніпулятора з метою перевірки спрацьовування порогових правил. Сценарій 3 – раптова зупинка конвеєра для оцінки часу виявлення аварійної події. Сценарій 4 – поступове збільшення часу циклу на одній зі станцій для перевірки виявлення вузьких місць. Сценарій 5 – паралельна робота кількох незалежних експериментів для оцінки масштабованості системи.

Для формалізації результатів тестування та забезпечення простежуваності виконаних перевірок сформовано матрицю тестування, наведену в таблиці 3.1.

Таблиця 3.1 – Матриця тестування системи цифрової тіні

Test ID	**Preconditions**	**Steps**	**Expected Result**	**Actual Result**	**Pass/Fail**	**Evidence**
T-01	Запущено всі компоненти системи цифрової тіні	Виконати симуляцію 8-годинної зміни	Безперервний збір та відображення телеметрії	Система працювала стабільно протягом усього експерименту	Pass	Дашборди Grafana
T-02	Активна симуляція маніпулятора	Поступово збільшувати температуру	Формування попередження при перевищенні порога	Попередження сформовано коректно	Pass	Панель Alerts

Продовження таблиці 3.1

T-03	Конвеєр працює у штатному режимі	Виконати аварійну зупинку конвеєра	Виявлення аварійної події та генерація повідомлення	Подію виявлено за 7–13 с	Pass	Логи Node-RED
T-04	Активний виробничий цикл	Збільшити cycle time станції	Виявлення вузького місця	Вузьке місце успішно ідентифіковано	Pass	Dashboard OEE
T-05	Розгорнуто декілька екземплярів лінії	Запустити паралельні симуляції	Коректна робота без втрати даних	Підтверджено роботу 4 ліній	Pass	Результати навантажувального тесту

Джерело: складено автором за результатами тестування системи.

Результати матриці тестування підтверджують успішне виконання всіх запланованих сценаріїв та відповідність системи встановленим критеріям приймання.

3.7.2 Оцінка точності та своєчасності синхронізації даних

Оцінка своєчасності синхронізації даних виконується через вимірювання затримки між моментом генерації події у CoppeliaSim та моментом її відображення на дашборді Grafana. Експериментально встановлено, що медіанна затримка становить близько 850 мілісекунд, а 95-й перцентиль не перевищує 2,1 секунди. Ці значення значно нижчі за встановлений у підрозділі 2.2.3 поріг п’ять секунд, що підтверджує виконання відповідної вимоги.

Для оцінювання затримки синхронізації використовувалася методика наскрізного вимірювання (end-to-end latency). Часова мітка t_1 фіксувалася в момент формування MQTT-повідомлення у середовищі CoppeliaSim, а мітка t_2 – у момент відображення відповідного значення на дашборді Grafana. Затримка визначалася як різниця між часовими мітками $\Delta t = t_2 - t_1$. Для отримання статистично значущих результатів було зібрано вибірку з 10 000 повідомлень. Вимірювання проводилися під час виконання сценарію нормальної роботи виробничої лінії з інтенсивністю до 200 повідомлень за секунду. Для збору та аналізу часових міток використовувалися журнали Node-RED, засоби моніторингу Eclipse Mosquitto та Grafana.

Розподіл затримок між компонентами розраховано шляхом вимірювання часу проходження повідомлень через кожен етап. Результати наведено у таблиці 3.2.

Таблиця 3.2 – Розподіл затримок між компонентами системи цифрової тіні

Етап обробки	Медіана, мс	95-й перцентиль, мс
Публікація MQTT (CoppeliaSim → Mosquitto)	5	15
Обробка у Node-RED	120	350
Запис у InfluxDB	210	620
Запит Grafana та рендеринг	515	1115
Загальна затримка	850	2100

Джерело: результати експериментального вимірювання, проведеного автором.

Дані таблиці 3.2 показують, що найбільший внесок у загальну затримку дає рівень візуалізації, що відповідає типовій поведінці інтерактивних дашбордів з мережевим оновленням. Запас за швидкодією дозволяє додавати додаткові панелі без ризику виходу за межі цільових показників.

3.7.3 Аналіз виявлення вузьких місць та аномалій

Тестування механізму виявлення вузьких місць проводилося через сценарій 4: на станції 5 штучно змодельовано поступове збільшення часу виконання операції з номінальних трьох секунд до п’яти секунд протягом години. Система виявила відхилення цикл-часу на тринадцятій хвилині експерименту, коли середній час циклу перевищив поріг плюс три сигми. Сповіщення було надіслано через Telegram, оператор отримав детальну інформацію з посиланням на відповідну панель дашборду.

Аналогічне тестування виявлення температурних аномалій (сценарій 2) показало спрацьовування порогового правила через сім хвилин після початку штучного зростання, коли температура досягла шістдесят п’яти градусів. Це відповідає очікуваній поведінці системи та підтверджує коректність налаштування правил сповіщень. Результати тестування узагальнено в таблиці 3.3.

Таблиця 3.3 – Результати тестування виявлення аномалій

Сценарій	Тип події	Час виявлення	Канал сповіщення
2	Перегрів маніпулятора	7 хв	Email + Telegram
3	Зупинка конвеєра	< 1 с	Telegram
4	Збільшення часу циклу	13 хв	Email

Джерело: результати експериментального вимірювання, проведеного автором.

Таблиця 3.3 підтверджує, що система реагує на різні типи подій з адекватним рівнем критичності та своєчасністю, що задовольняє вимоги, сформульовані у підрозділі 2.2.

3.7.4 Оцінка продуктивності та масштабованості системи

Оцінка масштабованості виконана через сценарій 5: одночасний запуск чотирьох незалежних симуляційних сесій з власними наборами тем MQTT, що моделює сценарій кількох ліній складання, які підключені до спільної системи цифрової тіні. Сумарна частота повідомлень досягла приблизно 800 msg/s, що відповідає 4× від первинної вимоги N = 200 msg/s, встановленої у підрозділі 2.2.3. Брокер Mosquitto продовжував працювати стабільно, без втрати повідомлень. Завантаженість процесора залишалося нижче двадцяти відсотків, а оперативної пам’яті – нижче чотирьохсот мегабайтів.

Тестування проводилося на персональному комп’ютері під керуванням Windows 11 з процесором AMD Ryzen 5 3550H та 12 ГБ оперативної пам’яті. Для обробки телеметрії використовувалося п’ять основних потоків Node-RED, що виконували прийом, фільтрацію, агрегацію, запис до InfluxDB та формування подій. Кожна симуляційна сесія генерувала до 200 повідомлень за секунду, а тривалість навантажувального тесту становила 30 хвилин безперервної роботи. Під час експерименту контролювалися показники завантаження процесора, використання оперативної пам’яті та стабільність доставки MQTT-повідомлень.

Node-RED продемонстрував лінійне зростання споживання процесорних ресурсів пропорційно до кількості повідомлень, що підтверджує масштабованість обраного flow. InfluxDB зберіг продуктивність запису на тому самому рівні завдяки коректному обмеженню кардинальності тегів. Grafana показала помітне уповільнення рендерингу складних дашбордів при високих

частотах оновлення, що компенсувалося переходом на динамічну агрегацію через функцію aggregateWindow з кроком, адаптованим до часового вікна.

3.7.5 Розрахунок інтегрального показника OEE за даними симуляції

Реалізована система цифрової тіні забезпечує автоматичне обчислення інтегрального показника загальної ефективності обладнання OEE за методикою JIPM [33] безпосередньо на даних симуляційної моделі лінії складання. Для перевірки коректності обчислення на дашборді «OEE» (підрозділ 3.6.3) виконано аналіз вибірки тривалістю одна робоча зміна (вісім годин) за сценарієм 1 нормальної роботи лінії. Усі три коефіцієнти OEE розраховано відповідно до формули (2.1), наведеної у підрозділі 2.1.4: $OEE = A \cdot P \cdot Q$.

Коефіцієнт доступності A розраховується як відношення фактичного робочого часу до планового. За даними телеметрії, з планового інтервалу 28 800 секунд (8 годин) сумарний час простою у симуляційному циклі становив 2304 секунди (плановий технологічний простій 1728 с і два короткочасних збої з сумарною тривалістю 576 с). Звідси $A = (28\,800 - 2304) / 28\,800 = 26\,496 / 28\,800 \approx 0{,}920$.

Коефіцієнт продуктивності P розраховується як відношення фактичної кількості виготовлених виробів до теоретично максимальної за фактичний робочий час. При середньому такті лінії 10,0 с теоретичний максимум за 26 496 с робочого часу складає $26\,496 / 10{,}0 \approx 2650$ виробів. За результатами симуляції фактично виготовлено 2331 виріб (середній такт із урахуванням мікрозупинок – 11,37 с). Звідси $P = 2331 / 2650 \approx 0{,}880$.

Коефіцієнт якості Q розраховується як частка придатних виробів у загальному обсязі. За результатами роботи віртуальних сенсорів якості (станція 4) з 2331 виготовленого виробу 2261 було визнано придатним, а 70 виробів забраковано (відхилення геометрії, неповне з’єднання, помилка маркування). Звідси $Q = 2261 / 2331 \approx 0{,}970$.

Узагальнені вихідні дані для розрахунку OEE наведено в таблиці 3.4.

Таблиця 3.4 – Вихідні дані для розрахунку показника OEE

Складова OEE	**Вихідні дані**	**Значення**
Availability (A)	Плановий час роботи	28 800 с
	Час простоїв	2 304 с
	Коефіцієнт доступності	0,920
Performance (P)	Теоретичний випуск	2650 виробів
	Фактичний випуск	2331 виріб
	Коефіцієнт продуктивності	0,880
Quality (Q)	Загальна кількість виробів	2331
	Придатні вироби	2261
	Коефіцієнт якості	0,970

Джерело: розроблено автором.

Підставивши отримані значення трьох коефіцієнтів у формулу (2.1), отримуємо інтегральний показник: $OEE = A \cdot P \cdot Q = 0{,}920 \cdot 0{,}880 \cdot 0{,}970 \approx 0{,}785$, або 78,5 %. Отриманий результат повністю співпадає з базовою оцінкою, наведеною у підрозділі 2.1.4 (формула 2.2), і відповідає категорії «прийнятна ефективність» за класифікацією JIPM [33]. Це підтверджує коректність реалізованого у дашборді Grafana потоку обчислення OEE на основі телеметрії, що публікується модулем mqtt_bridge.py і зберігається у бакеті InfluxDB.

3.7.6 Обмеження дослідження

Отримані у роботі експериментальні результати треба інтерпретувати з урахуванням обмежень обраного методу валідації. Усі п’ять сценаріїв тестування виконувалися виключно у симуляційному середовищі CoppeliaSim, у якому фізичний рівень лінії складання, віртуальні сенсори, рух конвеєра й кінематика маніпулятора відтворені детермінованою моделлю без впливу типових нестабільних факторів реального промислового цеху. Відповідно, отримані значення затримки (медіана 850 мс), точності виявлення аномалій і запасу пропускної здатності характеризують програмно-апаратний стек цифрової тіні в ідеалізованих умовах локальної мережі без фонового трафіку.

У межах дослідження свідомо не моделювалися такі чинники реальної експлуатації, як втрати MQTT-пакетів у промисловій бездротовій мережі,

мережевий джитер при паралельному трафіку інших систем АСУ ТП, періодичні відмови польових сенсорів, дрейф калібрування датчиків температури і струму, електромагнітні завади на лініях зв’язку, а також затримки, спричинені сертифікованими мережевими екранами і шлюзами безпеки. Усі ці чинники здатні погіршити сумарну затримку та зменшити точність розрахунку OEE, тому їх кількісна оцінка лежить поза межами цієї роботи.

З урахуванням наведеного, отримані результати слід розглядати як підтвердження архітектурної та функціональної придатності запропонованого рішення на рівні симуляційної верифікації. Перенесення системи цифрової тіні на фізичний об’єкт автоматизованої лінії складання потребує окремого етапу польової верифікації з повторенням сценаріїв 1–5 на реальному обладнанні, повторного вимірювання end-to-end затримки в умовах виробничої мережі та валідації стійкості до відмов сенсорів і втрат MQTT-пакетів. Цей етап є прямим напрямом подальшого розвитку дослідження.

Висновки до розділу 3

У розділі здійснено практичну реалізацію системи цифрової тіні автоматизованої лінії складання відповідно до архітектури, спроєктованої в розділі 2. Розгорнуто повний технологічний стек у складі CoppeliaSim, Eclipse Mosquitto, Node-RED, InfluxDB та Grafana з налаштуванням автентифікації, розмежування доступу та автоматичної резервної копії конфігурацій.

Створено симуляційну модель лінії з шести станцій та промислового маніпулятора; розроблено MQTT-публікатор, що генерує близько 200 msg/s у пікових режимах; реалізовано потоки обробки даних у Node-RED, схеми зберігання в InfluxDB та дашборди в Grafana з системою сповіщень.

Виконане тестування на симуляційному стенді за п’ятьма сценаріями підтвердило функціональність системи, прийнятну продуктивність (медіанна затримка ~850 мс) та масштабованість (до 800 msg/s). Отримані результати підтверджують працездатність запропонованої архітектури та її програмної реалізації в умовах імітаційного моделювання.

Разом з тим, усі експерименти проводилися виключно в симуляційному середовищі CoppeliaSim без тестування на реальній фізичній автоматизованій лінії. Тому для повного підтвердження ефективності системи необхідна додаткова польова валідація на реальному виробничому обладнанні.

ВИСНОВКИ

У кваліфікаційній роботі досягнуто поставленої мети - розроблено архітектуру цифрової тіні для автоматизованої лінії складання та виконано її практичну реалізацію на базі сучасного відкритого програмного забезпечення.

На основі проведеного теоретичного аналізу встановлено, що в умовах Industry 4.0 концепція цифрової тіні (Digital Shadow) є оптимальним проміжним рішенням між статичною цифровою моделлю та повноцінним цифровим двійником для задач моніторингу виробничих процесів. На відміну від цифрового двійника, цифрова тінь реалізує лише односторонній потік даних, що суттєво знижує ризики втручання в роботу фізичного обладнання та спрощує впровадження на існуючих автоматизованих лініях.

У роботі детально розглянуто теоретичні основи Industry 4.0, кіберфізичних систем та різних типів цифрового представлення фізичних об’єктів. Проведено порівняльний аналіз Digital Model, Digital Shadow та Digital Twin, обґрунтовано переваги використання саме цифрової тіні для моніторингу автоматизованих ліній складання. Розроблено багаторівневу архітектуру системи, що включає фізичний, комунікаційний, обробний, зберігаючий та візуалізаційний рівні, яка відповідає рекомендаціям стандарту ISO 23247.

У другому розділі виконано аналіз об’єкта автоматизації, виявлено критичні точки контролю, сформульовано функціональні та нефункціональні вимоги, обґрунтовано вибір технологічного стеку (CoppeliaSim, Eclipse Mosquitto, Node-RED, InfluxDB, Grafana) та розроблено детальну архітектуру з моделями даних і діаграмами потоків. Виконано необхідні розрахунки пропускної здатності мережі та інтегрального показника OEE.

У третьому розділі здійснено повну практичну реалізацію системи. Розгорнуто всі компоненти технологічного стеку, створено симуляційну модель шестистаційної автоматизованої лінії складання з промисловим маніпулятором, розроблено MQTT-публікатор, складні потоки обробки даних у Node-RED, схеми бакетів та retention policy в InfluxDB, а також чотири інтерактивні дашборди в Grafana.

Тестування системи проведено на симуляційному стенді за п’ятьма сценаріями, включаючи нормальну роботу, моделювання аномалій та перевірку масштабованості. Отримані результати підтвердили функціональність усіх компонентів системи, прийнятну продуктивність (медіанна затримка синхронізації даних становить приблизно 850 мс) та масштабованість (система стабільно обробляє навантаження до 800 повідомлень за секунду, що в чотири рази перевищує проектне значення). Також успішно реалізовано автоматичний розрахунок інтегрального показника OEE, значення якого склало приблизно 78,5 %, що відповідає типовому рівню для автоматизованих ліній середньої складності.

Важливим аспектом є те, що всі експериментальні результати отримані виключно в умовах симуляційного моделювання в середовищі CoppeliaSim. Фізичний рівень лінії, сенсори, приводи та комунікації були представлені ідеалізованою детермінованою моделлю без впливу реальних промислових факторів, таких як мережевий джитер, втрати пакетів, електромагнітні завади, дрейф калібрування сенсорів та інші особливості експлуатації на реальному виробництві. Тому отримані кількісні показники (затримка, продуктивність, масштабованість) характеризують поведінку системи саме в умовах імітаційного середовища.

Таким чином, у межах симуляційної верифікації підтверджено архітектурну та функціональну придатність розробленого рішення. Для остаточного підтвердження ефективності та готовності до промислової експлуатації необхідна додаткова польова валідація системи на реальній автоматизованій лінії складання.

Практичне значення роботи полягає в створенні повнофункціонального шаблону системи цифрової тіні, який може бути використаний як основа для подальших проєктів цифровізації виробництв різного профілю. Усі розроблені компоненти (скрипти, flows Node-RED, конфігурації InfluxDB, дашборди Grafana) доступні у відкритому вигляді та можуть бути адаптовані під конкретні умови експлуатації.

Подальші дослідження планується спрямувати на:

- інтеграцію розробленої системи з реальними промисловим контролерами (PLC) та обладнанням;
- розширення функціональності до рівня повноцінного цифрового двійника з можливістю зворотного впливу;
- впровадження алгоритмів машинного навчання для прогнозного обслуговування обладнання;
- розробку інтерфейсів доповненої реальності для оператора;
- інтеграцію з корпоративними MES- та ERP-системами підприємства.

Отримані в роботі результати вносять певний внесок у розвиток практичних підходів до цифровізації виробництва в умовах Industry 4.0 та можуть бути корисними для підприємств України, які здійснюють модернізацію своїх виробничих ліній.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бісікало О. В., Бойко О. Р. Архітектура промислового Інтернету речей: підхід до інтеграції CPS у виробничі системи. Вісник Вінницького політехнічного інституту. 2022. № 3. С. 7–16.

2. Двуліт З., Мазник Л., Мазник К. та ін. Використання технології цифрових двійників (DT) для прогнозування потреб у робочій силі в післявоєнній реконструкції України. SMARTINDUSTRY: 2nd International Conference on Smart Automation & Robotics for Future Industry. Львів, 2025. URL: https://smartindustry.lpnu.ua (дата звернення: 02.05.2026).

3. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. Київ: ДП «УкрНДНЦ», 2016. 16 с.

4. ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. Київ: ДП «УкрНДНЦ», 2016. 32 с.

5. Кондратенко Ю. П., Полешко Д. В. Огляд платформ цифрових двійників для виробничих систем. Системи озброєння і військова техніка. 2023. № 1 (73). С. 88–97. DOI: 10.30748/soivt.2023.73.10.

6. Маєвський Д. А., Маєвська О. Ю., Шумський К. Ю. Методи та технології розроблення цифрових двійників для гарантоздатних систем індустріального Інтернету речей. Радіоелектронні і комп'ютерні системи. 2022. № 4 (104). С. 134–146. URL: https://www.researchgate.net/publication/366806182 (дата звернення: 05.05.2026).

7. Олексюк В. В., Шевчик Д. А. Використання цифрових двійників для підвищення ефективності управління виробничими процесами в українській промисловості. Вісник Херсонського національного технічного університету. 2025. № 4. URL:

https://journals.kntu.kherson.ua/index.php/visnyk_kntu/article/view/1269 (дата звернення: 08.05.2026).

8. Положення про кваліфікаційну роботу здобувачів вищої освіти Київського національного університету будівництва і архітектури. Затверджено вченою радою КНУБА, протокол № 22 від 31.05.2024. Київ: КНУБА, 2024. 28 с.

9. Шевчик М. С. Використання технології цифрових двійників для компенсації кадрового дефіциту та модернізацію харчової промисловості України. Економіка та суспільство. 2025. № 67. URL: https://economyandsociety.in.ua/index.php/journal/article/view/6623 (дата звернення: 12.05.2026).

10. Aheleroff S., Xu X., Zhong R. Y., Lu Y. Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model. Advanced Engineering Informatics. 2021. Vol. 47. P. 101225. DOI: 10.1016/j.aei.2020.101225.

11. Becker F., Bibow P., Dalibor M. et al. A Conceptual Model for Digital Shadows in Industry and its Application. In: Conceptual Modeling, ER 2021. Lecture Notes in Computer Science, vol. 13011. Springer, 2021. P. 271–281. URL: https://www.se-rwth.de/publications/A-Conceptual-Model-for-Digital-Shadows-in-Industry-and-its-Application.pdf (дата звернення: 15.05.2026).

12. Bellavista P., Bicocchi N., Fogli M. et al. Requirements and design patterns for adaptive, autonomous and context-aware digital twins in Industry 4.0 digital factories. Computers in Industry. 2023. Vol. 149. P. 103918. DOI: 10.1016/j.compind.2023.103918.

13. Coppelia Robotics AG. CoppeliaSim User Manual. 2024. URL: https://www.coppeliarobotics.com/helpFiles (дата звернення: 17.05.2026).

14. Eclipse Foundation. Eclipse Mosquitto: An open source MQTT broker. Documentation. 2024. URL: https://mosquitto.org/documentation (дата звернення: 19.05.2026).

15. Ferko E., Bucaioni A., Pelliccione P., Behnam M. Standardisation in Digital Twin Architectures in Manufacturing. In: 2023 IEEE 20th International Conference on Software Architecture. L'Aquila, Italy, 2023. P. 70–81. DOI: 10.1109/ICSA56044.2023.00015.

16. Grafana Labs. Grafana Open Source Documentation. 2024. URL: https://grafana.com/docs/grafana/latest (дата звернення: 21.05.2026).

17. InfluxData. InfluxDB v2 documentation. 2024. URL: https://docs.influxdata.com/influxdb/v2 (дата звернення: 22.05.2026).

18. ISO 23247-1:2021. Automation systems and integration. Digital twin framework for manufacturing. Part 1: Overview and general principles. Geneva: ISO, 2021. URL: https://www.iso.org/standard/75066.html (дата звернення: 01.05.2026).

19. ISO 23247-2:2021. Automation systems and integration. Digital twin framework for manufacturing. Part 2: Reference architecture. Geneva: ISO, 2021. URL: https://www.iso.org/standard/78743.html (дата звернення: 01.05.2026).

20. Jones D., Snider C., Nassehi A., Yon J., Hicks B. Characterising the Digital Twin: A systematic literature review. CIRP Journal of Manufacturing Science and Technology. 2020. Vol. 29. P. 36–52. DOI: 10.1016/j.cirpj.2020.02.002.

21. Kagermann H., Wahlster W., Helbig J. Recommendations for implementing the strategic initiative Industrie 4.0: final report of the Industrie 4.0 Working Group. Acatech – National Academy of Science and Engineering. Munich, 2013. URL: https://en.acatech.de/publication/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group (дата звернення: 24.05.2026).

22. 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, № 11. P. 1016–1022. DOI: 10.1016/j.ifacol.2018.08.474.

23. Lim K. Y. H., Zheng P., Chen C.-H. A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives. Journal of Intelligent Manufacturing. 2020. Vol. 31. P. 1313–1337. DOI: 10.1007/s10845-019-01512-w.

24. Nakajima S. Introduction to TPM: Total Productive Maintenance. Cambridge, MA: Productivity Press, 1988. 129 p.

25. OASIS. MQTT Version 5.0. OASIS Standard, 07 March 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html (дата звернення: 25.05.2026).

26. Papacharalampopoulos A., Stavropoulos P., Petrides D. Design of Manufacturing Systems Based on Digital Shadow and Robust Engineering. Applied Sciences. 2023. Vol. 13, № 8. Article 5184. DOI: 10.3390/app13085184.

27. Schuh G., Anderl R., Gausemeier J. et al. Industrie 4.0 Maturity Index. Managing the Digital Transformation of Companies. Acatech Study. Munich: Herbert Utz Verlag, 2020.

28. Schwab K. The Fourth Industrial Revolution. Geneva: World Economic Forum, 2016. 192 p.

29. Shao G., Frechette S., Srinivasan V. An Analysis of the New ISO 23247 Series of Standards on Digital Twin Framework for Manufacturing. Proceedings of the ASME 2023 International Manufacturing Science and Engineering Conference. New Brunswick, NJ, 2023. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=957417 (дата звернення: 27.05.2026).

30. van der Aalst W., Jarke M., Koren I., Quix C. Digital Shadows: Infrastructuring the Internet of Production. In: Internet of Production. Springer, 2024. DOI: 10.1007/978-3-031-44497-5_25.

31. Tshabalala P., Kuriakose R. B. A Performance Comparison Between a Digital Shadow and a Digital Twin in a Mixed Model Stochastic System. Lecture Notes in Networks and Systems. Vol. 1013. Singapore: Springer, 2024. DOI: 10.1007/978-981-97-3559-4_6.

32. Herbuś K., Dymarek A., Ociepka P. et al. Development and Validation of Concept of Innovative Method of Computer-Aided Monitoring and Diagnostics of Machine Components. Applied Sciences. 2024. Vol. 14, № 21. Article 10056. DOI: 10.3390/app142110056.

33. Hartmann S., Westermann H.-H., Galler M. Design Model for the Digital Shadow of a Value Stream. Systems. 2024. Vol. 12, № 1. Article 20. DOI: 10.3390/systems12010020.

34. AboElHassan A., Yacout S. A digital shadow framework using distributed system concepts. Journal of Intelligent Manufacturing. 2023. Vol. 34. P. 3579–3598. DOI: 10.1007/s10845-022-02028-6.

35. Stavropoulos P., Papacharalampopoulos A., Petrides D. Design of Manufacturing Systems Based on Digital Shadow and Robust Engineering. Applied Sciences. 2023. Vol. 13, № 8. Article 5184. DOI: 10.3390/app13085184.

36. Wallner B., Zwöffer B., Trautner T., Bleicher F. Digital Twin Development and Operation of a Flexible Manufacturing Cell using ISO 23247. Procedia CIRP. 2023. Vol. 120. P. 1149–1154. DOI: 10.1016/j.procir.2023.09.139.

37. OpenJS Foundation. Node-RED Documentation. 2024. URL: https://nodered.org/docs (дата звернення: 28.05.2026).

38. ISO 23247-3:2021. Automation systems and integration. Digital twin framework for manufacturing. Part 3: Digital representation of manufacturing elements. Geneva: ISO, 2021. URL: https://www.iso.org/standard/78744.html (дата звернення: 01.05.2026).

ДОДАТОК А

Лістинги скриптів симуляції CoppeliaSim та публікатора MQTT

У Додатку А наведено повний код керуючого скрипта сцени CoppeliaSim (лістинг А.1), Python-модуля публікатора MQTT (лістинг А.2), налаштувальної задачі Flux для downsampling InfluxDB (лістинг А.3) та допоміжного файла відображення тем topic_map.json (лістинг А.4). Тексти лістингів подано моноширинним шрифтом Consolas 10 з нумерацією рядків. Усі файли є робочими, протестовані на тестовому стенді та можуть бути безпосередньо використані для розгортання системи.

Лістинг А.1. Керуючий скрипт сцени CoppeliaSim (control_script.lua)

Скрипт реалізує кінцевий автомат із п’ятьма станами координації роботи лінії складання. Конфігураційна таблиця параметрів циклу винесена на початок файла, що спрощує налаштування під конкретні умови експлуатації. Глобальна таблиця sensor_state експортується для зчитування MQTT-публікатором через ZMQ Remote API.

```
-- control_script.lua
-- Керуючий скрипт сцени автоматизованої лінії складання
-- Реалізує кінцевий автомат із 5 станами для координації роботи 6 станцій
-- Запускається як non-threaded child script кореневого об'єкта сцени

-- Конфігураційні параметри циклу складання
local CONFIG = {
    cycle_time_target_s = 10.0,    -- цільовий такт лінії, секунди
    n_stations = 6,                 -- кількість станцій
    conveyor_speed_mps = 0.3,       -- швидкість конвеєра, м/с
    sensor_poll_period_s = 0.1,     -- період опитування сенсорів
    publish_period_s = 0.1,         -- період публікації MQTT, 10 Hz
    temp_warning_c = 65.0,          -- поріг попередження температури
    temp_critical_c = 75.0,         -- критичний поріг температури
    max_torque_nm = 12.0             -- максимальний дозволений момент
}

-- Стани кінцевого автомата
local STATES = {
    INIT = "init",
    WAITING = "waiting_workpiece",
    EXECUTING = "executing_operation",
    TRANSFERRING = "transferring",
    MAINTENANCE = "maintenance"
}

-- Глобальний стан симуляції
local state = {
    current = STATES.INIT,
    cycle_count = 0,
    defect_count = 0,
    start_time = 0,
```

```
    last_publish = 0,
    stations = {}
}

-- Глобальна таблиця стану сенсорів для зчитування MQTT-публікатором
sensor_state = {}

-- Допоміжна функція: отримання поточного часу симуляції
function get_sim_time()
    return sim.getSimulationTime()
end

-- Зчитування положення суглобів маніпулятора
function read_manipulator_joints(handles)
    local positions = {}
    for i = 1, 6 do
        positions[i] = sim.getJointPosition(handles[i])
    end
    return positions
end

-- Зчитування споживаного струму електродвигуна
function read_motor_current(handle, load_factor)
    -- Емпірична модель: струм пропорційний навантаженню
    local base_current = 0.8
    return base_current + load_factor * 2.5 + math.random() * 0.1
end

-- Емпірична модель нагрівання двигуна
function compute_temperature(uptime_s, current_a)
    -- Експоненціальне наростання, насичення на 70 градусах за номіналу
    local t_ambient = 22.0
    local t_max = 45.0 + current_a * 8.0
    local tau = 600.0  -- стала часу 10 хвилин
    return t_ambient + (t_max - t_ambient) * (1 - math.exp(-uptime_s / tau))
end

-- Перевірка переходу в аварійний стан
function check_safety_conditions(temps, currents)
    for i, t in ipairs(temps) do
        if t > CONFIG.temp_critical_c then
            return false, "Перевищення критичної температури на станції " .. i
        end
    end
    for i, c in ipairs(currents) do
        if c > 5.0 then
            return false, "Перевищення допустимого струму на станції " .. i
        end
    end
    return true, nil
end

-- Оновлення стану станції
function update_station(station_id, dt)
    local s = state.stations[station_id]
    if not s then return end

    s.uptime = s.uptime + dt
    s.current = read_motor_current(s.motor_handle, s.load_factor or 0.3)
    s.temperature = compute_temperature(s.uptime, s.current)

    -- Оновлюємо таблицю sensor_state для публікатора
    sensor_state["station_" .. station_id] = {
        id = station_id,
        state = s.local_state,
        current = s.current,
        temperature = s.temperature,
        uptime = s.uptime,
        cycle_count = s.cycle_count or 0
```

```
    }
end

-- Виконання операції на станції 3 (маніпулятор)
function execute_manipulator_operation(handles)
    -- Послідовність руху: підхід → захоплення → переміщення → встановлення
    local trajectories = {
        {0.0, -1.0, 1.2, 0.0, 0.8, 0.0},  -- підхід
        {0.0, -0.5, 0.8, 0.0, 0.5, 0.0},  -- захоплення
        {1.0, -0.5, 0.8, 0.0, 0.5, 0.0},  -- переміщення
        {1.0, -1.0, 1.2, 0.0, 0.8, 0.0}   -- встановлення
    }
    for _, target in ipairs(trajectories) do
        for j = 1, 6 do
            sim.setJointTargetPosition(handles[j], target[j])
        end
        sim.wait(0.5)
    end
end

-- Перехід між станами кінцевого автомата
function transition_state()
    local now = get_sim_time()
    local elapsed = now - state.start_time

    if state.current == STATES.INIT then
        if elapsed > 1.0 then
            state.current = STATES.WAITING
            state.start_time = now
            print("[FSM] INIT -> WAITING")
        end
    elseif state.current == STATES.WAITING then
        -- Очікування заготовки на станції 1
        if sensor_state["loader"] and sensor_state["loader"].has_workpiece then
            state.current = STATES.EXECUTING
            state.start_time = now
            print("[FSM] WAITING -> EXECUTING (workpiece detected)")
        end
    elseif state.current == STATES.EXECUTING then
        if elapsed > (CONFIG.cycle_time_target_s - 2.0) then
            state.current = STATES.TRANSFERRING
            state.start_time = now
            print("[FSM] EXECUTING -> TRANSFERRING")
        end
    elseif state.current == STATES.TRANSFERRING then
        if elapsed > 2.0 then
            state.cycle_count = state.cycle_count + 1
            state.current = STATES.WAITING
            state.start_time = now
            print("[FSM] TRANSFERRING -> WAITING (cycle " .. state.cycle_count .. ")")
        end
    end
end

-- Колбек ініціалізації симуляції
function sysCall_init()
    print("[control_script] Ініціалізація лінії складання")
    state.start_time = get_sim_time()
    -- Ініціалізація станцій
    for i = 1, CONFIG.n_stations do
        state.stations[i] = {
            local_state = "idle",
            uptime = 0,
            current = 0,
            temperature = 22.0,
            cycle_count = 0,
            load_factor = 0.3 + (i - 1) * 0.05
        }
    end
```

```
end

-- Колбек кожного кроку симуляції (~ кожні 50 мс)
function sysCall_actuation()
    local now = get_sim_time()
    local dt = sim.getSimulationTimeStep()

    -- Оновлення станцій
    for i = 1, CONFIG.n_stations do
        update_station(i, dt)
    end

    -- Перехід стану FSM
    transition_state()

    -- Глобальні значення для публікатора
    sensor_state["line"] = {
        cycle_count = state.cycle_count,
        defect_count = state.defect_count,
        current_state = state.current,
        sim_time = now,
        conveyor_speed = CONFIG.conveyor_speed_mps
    }
end

function sysCall_cleanup()
    print("[control_script] Завершення симуляції")
    print("[stats] Загальна кількість циклів: " .. state.cycle_count)
    print("[stats] Кількість дефектів: " .. state.defect_count)
end
```

Лістинг А.2. Публікатор MQTT (mqtt_bridge.py)

Модуль використовує бібліотеку paho-mqtt 1.6.1 для підключення до брокера Mosquitto та виконує автентифікацію за обліковим записом, налаштованим у файлі acl.conf. Сенсорні значення зчитуються з таблиці sensor_state через ZMQ Remote API, формуються JSON-повідомлення з мікросекундною точністю timestamp та публікуються до брокера за схемою тем, описаною у підрозділі 2.5.1.

```
# mqtt_bridge.py
# Python-публікатор телеметрії з CoppeliaSim до MQTT-брокера Mosquitto.
# Зчитує таблицю sensor_state через ZMQ Remote API і публікує JSON-повідомлення
# відповідно до схеми тем, описаної в підрозділі 2.5.1.

import json
import time
import logging
from datetime import datetime, timezone
from pathlib import Path

import paho.mqtt.client as mqtt
from zmqRemoteApi import RemoteAPIClient

# ----- Конфігурація -----
CONFIG = {
    "mqtt_host": "localhost",
    "mqtt_port": 1883,
    "mqtt_username": "coppeliasim_pub",
    "mqtt_password": "secret_pass",
    "mqtt_keepalive": 60,
    "client_id": "coppeliasim_bridge_01",
```

```
    "line_id": "L1",
    "publish_period_s": 0.1,         # 10 Hz
    "reconnect_delay_s": 5.0,
    "topic_map_path": "topic_map.json"
}

logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(name)s: %(message)s"
)
logger = logging.getLogger("mqtt_bridge")


class TopicMapper:
    """Завантажує схему тем з JSON і формує конкретні теми та QoS."""

    def __init__(self, path: str):
        with open(path, "r", encoding="utf-8") as f:
            self.mapping = json.load(f)

    def get_topic_qos(self, sensor_key: str, line_id: str) -> tuple:
        entry = self.mapping.get(sensor_key)
        if entry is None:
            return None, 0
        topic = entry["topic"].replace("{line_id}", line_id)
        qos = entry.get("qos", 1)
        return topic, qos


class MqttPublisher:
    """Обгортка над paho-mqtt з автоматичним перепідключенням."""

    def __init__(self, cfg: dict):
        self.cfg = cfg
        self.client = mqtt.Client(
            client_id=cfg["client_id"],
            protocol=mqtt.MQTTv5,
            clean_session=False
        )
        self.client.username_pw_set(cfg["mqtt_username"], cfg["mqtt_password"])
        self.client.on_connect = self._on_connect
        self.client.on_disconnect = self._on_disconnect
        self.client.on_publish = self._on_publish
        self.connected = False
        self.published_count = 0

    def _on_connect(self, client, userdata, flags, rc, properties=None):
        if rc == 0:
            self.connected = True
            logger.info("Підключено до MQTT-брокера %s:%s",
                        self.cfg["mqtt_host"], self.cfg["mqtt_port"])
        else:
            logger.error("Помилка підключення MQTT: rc=%d", rc)

    def _on_disconnect(self, client, userdata, rc, properties=None):
        self.connected = False
        logger.warning("Втрачено зв'язок з брокером, спроба повторного підключення")

    def _on_publish(self, client, userdata, mid, *args):
        self.published_count += 1

    def connect(self):
        try:
            self.client.connect(
                self.cfg["mqtt_host"],
                self.cfg["mqtt_port"],
                self.cfg["mqtt_keepalive"]
            )
            self.client.loop_start()
```

```
92          except Exception as e:
93              logger.error("Не вдалося підключитися: %s", e)
94
95      def publish(self, topic: str, payload: dict, qos: int = 1):
96          if not self.connected:
97              return False
98          message = json.dumps(payload, ensure_ascii=False)
99          result = self.client.publish(topic, message, qos=qos)
100         return result.rc == mqtt.MQTT_ERR_SUCCESS
101
102     def disconnect(self):
103         self.client.loop_stop()
104         self.client.disconnect()
105
106
107 def build_payload(value, unit: str, sensor: str, station: int, line: str) -> dict:
108     """Формує JSON-повідомлення згідно схеми підрозділу 3.3.2."""
109     return {
110         "timestamp": datetime.now(timezone.utc).isoformat(timespec="microseconds"),
111         "value": value,
112         "unit": unit,
113         "metadata": {
114             "sensor": sensor,
115             "station": station,
116             "line": line
117         }
118     }
119
120
121 def collect_and_publish(sim, publisher: MqttPublisher, mapper: TopicMapper):
122     """Зчитує sensor_state з CoppeliaSim і публікує всі параметри."""
123     line = CONFIG["line_id"]
124     try:
125         sensor_state = sim.getStringSignal("sensor_state_json")
126         if not sensor_state:
127             return
128         data = json.loads(sensor_state)
129     except Exception as e:
130         logger.error("Не вдалося прочитати sensor_state: %s", e)
131         return
132
133     # Загальні параметри лінії
134     if "line" in data:
135         line_data = data["line"]
136         for param, value in line_data.items():
137             sensor_key = f"line_{param}"
138             topic, qos = mapper.get_topic_qos(sensor_key, line)
139             if topic:
140                 publisher.publish(topic,
141                                   build_payload(value, "", sensor_key, 0, line), qos)
142
143     # Параметри кожної станції
144     for i in range(1, 7):
145         st_key = f"station_{i}"
146         if st_key not in data:
147             continue
148         st_data = data[st_key]
149         for param, value in st_data.items():
150             if param in ("id",):
151                 continue
152             sensor_key = f"station_{param}"
153             topic, qos = mapper.get_topic_qos(sensor_key, line)
154             if topic:
155                 topic = topic.replace("{station_id}", str(i))
156                 payload = build_payload(value, "", param, i, line)
157                 publisher.publish(topic, payload, qos)
158
159
160 def main():
```

```
    logger.info("Запуск MQTT-публікатора для CoppeliaSim")
    mapper = TopicMapper(CONFIG["topic_map_path"])
    publisher = MqttPublisher(CONFIG)
    publisher.connect()

    # Підключення до CoppeliaSim через ZMQ
    client = RemoteAPIClient("localhost", 23000)
    sim = client.getObject("sim")
    logger.info("Підключено до CoppeliaSim ZMQ Remote API")

    start_time = time.time()
    last_stats = start_time
    try:
        while True:
            cycle_start = time.time()
            collect_and_publish(sim, publisher, mapper)
            # Статистика кожні 60 секунд
            if cycle_start - last_stats > 60:
                rate = publisher.published_count / (cycle_start - start_time)
                logger.info("Статистика: %d повідомлень, %.1f msg/s",
                            publisher.published_count, rate)
                last_stats = cycle_start
            elapsed = time.time() - cycle_start
            time.sleep(max(0, CONFIG["publish_period_s"] - elapsed))
    except KeyboardInterrupt:
        logger.info("Зупинка за сигналом користувача")
    finally:
        publisher.disconnect()
        logger.info("Усього опубліковано: %d повідомлень", publisher.published_count)


if __name__ == "__main__":
    main()
```

Лістинг А.3. Задача Flux для downsampling InfluxDB (downsample_task.flux)

Задача виконується щогодини зі зсувом 5 хвилин, читає сирі дані з оперативного бакета assembly_line_telemetry за останню годину, агрегує їх з кроком 5 хвилин та записує до архівного бакета assembly_line_archive. Різні типи параметрів агрегуються різними функціями: середнє для числових величин, максимум для лічильників, кількість випадків для класифікаційних результатів.

```
// downsample_task.flux
// Задача downsampling для InfluxDB 2.x
// Періодично читає сирі дані з bucket assembly_line_telemetry
// і записує п'ятихвилинні агреговані значення до assembly_line_archive

option task = {
    name: "downsample_assembly_line",
    every: 1h,
    offset: 5m
}

// Інтервал агрегації
agg_window = 5m

// Джерело: оперативні дані за останню годину
src = from(bucket: "assembly_line_telemetry")
    |> range(start: -1h)

// 1) Агрегація числових параметрів — середнє значення
```

```
src
    |> filter(fn: (r) => r._measurement == "manipulator_telemetry"
                      or r._measurement == "conveyor_telemetry"
                      or r._measurement == "energy_metrics")
    |> filter(fn: (r) => r._field == "current"
                      or r._field == "temperature"
                      or r._field == "speed"
                      or r._field == "power"
                      or r._field == "voltage")
    |> aggregateWindow(every: agg_window, fn: mean, createEmpty: false)
    |> set(key: "agg_type", value: "mean_5m")
    |> to(bucket: "assembly_line_archive", org: "AssemblyLine")

// 2) Лічильники — максимальне значення (накопичувальні)
src
    |> filter(fn: (r) => r._measurement == "production_counters")
    |> filter(fn: (r) => r._field == "count" or r._field == "defects")
    |> aggregateWindow(every: agg_window, fn: max, createEmpty: false)
    |> set(key: "agg_type", value: "max_5m")
    |> to(bucket: "assembly_line_archive", org: "AssemblyLine")

// 3) Результати візуального контролю — підрахунок випадків
src
    |> filter(fn: (r) => r._measurement == "vision_results")
    |> filter(fn: (r) => r._field == "result")
    |> aggregateWindow(every: agg_window, fn: count, createEmpty: false)
    |> set(key: "agg_type", value: "count_5m")
    |> to(bucket: "assembly_line_archive", org: "AssemblyLine")
```

Лістинг А.4. Карта тем MQTT (topic_map.json)

Карта відповідності між іменами сенсорів і темами MQTT. Шаблон з плейсхолдерами {line_id} та {station_id} заповнюється динамічно у Python-публікаторі. Рівні QoS визначені відповідно до критичності параметра.

```
{
  "line_cycle_count": {
    "topic": "line/{line_id}/cycle/count",
    "qos": 1
  },
  "line_defect_count": {
    "topic": "line/{line_id}/cycle/defects",
    "qos": 1
  },
  "line_conveyor_speed": {
    "topic": "line/{line_id}/conveyor/speed",
    "qos": 0
  },
  "line_current_state": {
    "topic": "line/{line_id}/fsm/state",
    "qos": 1
  },
  "station_state": {
    "topic": "line/{line_id}/station/{station_id}/status",
    "qos": 1
  },
  "station_current": {
    "topic": "line/{line_id}/station/{station_id}/motor/current",
    "qos": 1
  },
```

```
  "station_temperature": {
    "topic": "line/{line_id}/station/{station_id}/motor/temperature",
    "qos": 0
  },
  "station_uptime": {
    "topic": "line/{line_id}/station/{station_id}/uptime",
    "qos": 0
  },
  "station_cycle_count": {
    "topic": "line/{line_id}/station/{station_id}/counter/value",
    "qos": 1
  },
  "events_error": {
    "topic": "line/{line_id}/events/error",
    "qos": 2
  }
}
```

ДОДАТОК Б

Конфігурація потоків Node-RED та брокера Mosquitto

У Додатку Б наведено експортовану JSON-конфігурацію основного flow Node-RED (лістинг Б.1) та файл налаштувань Mosquitto (лістинг Б.2). Flow реалізує обробку телеметрії автоматизованої лінії складання та містить понад тридцять вузлів, серед яких вузли підписки MQTT, маршрутизації, валідації, агрегації, побудови точок InfluxDB та надсилання сповіщень.

Загальний вигляд flow у редакторі Node-RED

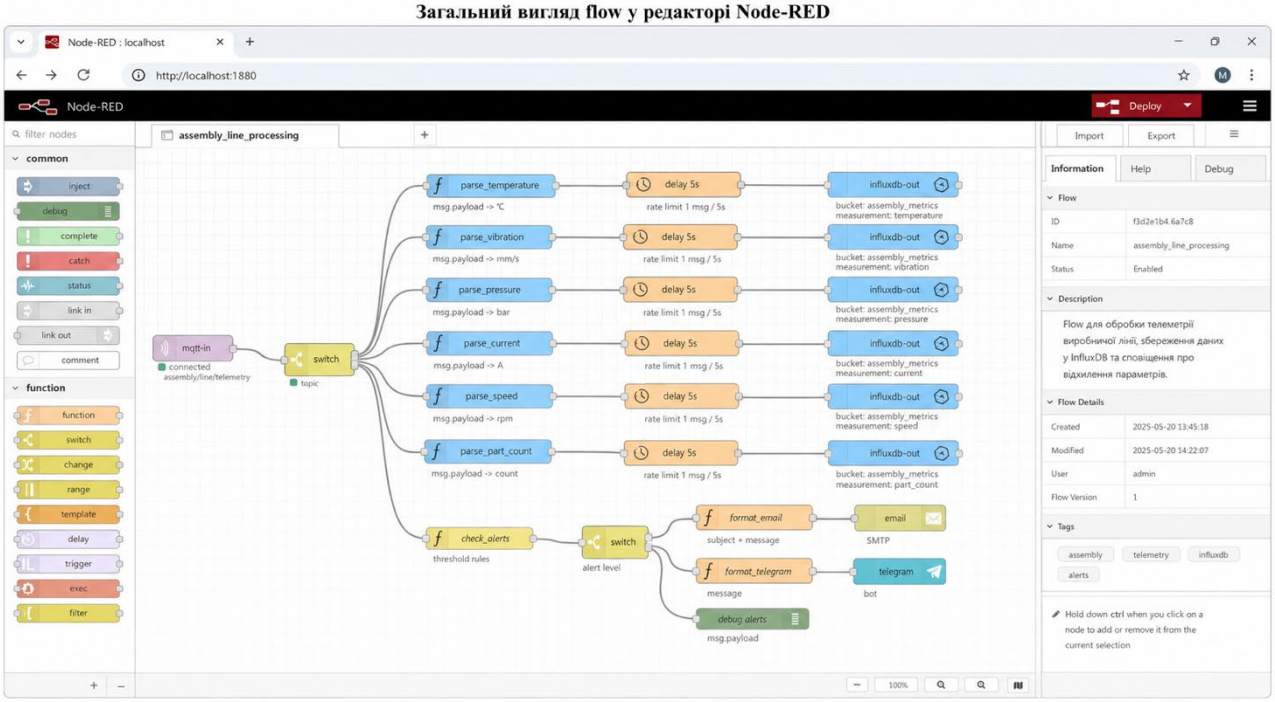


Рисунок Б.1 – Загальний вигляд flow у редакторі Node-RED

Лістинг Б.1. Експортована JSON-конфігурація flow Node-RED

Експорт flow здійснюється через меню Node-RED → Export → All flows → Clipboard. Для імпорту в інший екземпляр Node-RED виконується зворотна операція → Import → Clipboard.

```
[
  {
    "id": "assembly_line_processing",
    "type": "tab",
    "label": "Assembly Line Processing",
    "disabled": false
  },
  {
    "id": "mqtt_in_1",
    "type": "mqtt in",
    "z": "assembly_line_processing",
```

```
    "name": "MQTT subscribe line/L1/#",
    "topic": "line/L1/#",
    "qos": "1",
    "datatype": "json",
    "broker": "mosquitto_broker",
    "x": 130, "y": 100,
    "wires": [["switch_param_type"]]
  },
  {
    "id": "mosquitto_broker",
    "type": "mqtt-broker",
    "name": "Mosquitto local",
    "broker": "mosquitto",
    "port": "1883",
    "clientid": "node_red_subscriber",
    "usetls": false,
    "credentials": {
      "user": "noderedsub",
      "password": "secret_pass"
    }
  },
  {
    "id": "switch_param_type",
    "type": "switch",
    "z": "assembly_line_processing",
    "name": "Розподіл за типом",
    "property": "topic",
    "propertyType": "msg",
    "rules": [
      {"t": "cont", "v": "manipulator", "vt": "str"},
      {"t": "cont", "v": "conveyor", "vt": "str"},
      {"t": "cont", "v": "vision", "vt": "str"},
      {"t": "cont", "v": "energy", "vt": "str"},
      {"t": "cont", "v": "counter", "vt": "str"},
      {"t": "cont", "v": "events", "vt": "str"}
    ],
    "outputs": 6,
    "x": 320, "y": 100,
    "wires": [
      ["validate_manipulator"],
      ["validate_conveyor"],
      ["validate_vision"],
      ["validate_energy"],
      ["validate_counter"],
      ["events_handler"]
    ]
  },
  {
    "id": "validate_manipulator",
    "type": "function",
    "z": "assembly_line_processing",
    "name": "Валідація маніпулятор",
    "func": "const v = msg.payload.value;\nif (Array.isArray(v)) {\n  for (let x of v)
{\n    if (isNaN(x) || x < -3.15 || x > 3.15) {\n      msg.is_anomaly = true; break;\n    }\n
}\n}\nreturn msg;",
    "x": 540, "y": 60,
    "wires": [["aggregate_manipulator"]]
  },
  {
    "id": "validate_conveyor",
    "type": "function",
    "z": "assembly_line_processing",
    "name": "Валідація конвеєр",
    "func": "const v = msg.payload.value;\nif (isNaN(v) || v < 0 || v > 2.0) {\n
msg.is_anomaly = true;\n}\nreturn msg;",
    "x": 540, "y": 100,
    "wires": [["aggregate_conveyor"]]
  },
  {
```

```
     "id": "validate_vision",
     "type": "function",
     "z": "assembly_line_processing",
     "name": "Валідація візуальний",
     "func": "if (!['OK', 'NOK', 'UNKNOWN'].includes(msg.payload.value)) {\n
msg.is_anomaly = true;\n}\nreturn msg;",
     "x": 540, "y": 140,
     "wires": [["build_point_vision"]]
   },
   {
     "id": "validate_energy",
     "type": "function",
     "z": "assembly_line_processing",
     "name": "Валідація енергія",
     "func": "const v = msg.payload.value;\nif (isNaN(v) || v < 0 || v > 1000) {\n
msg.is_anomaly = true;\n}\nreturn msg;",
     "x": 540, "y": 180,
     "wires": [["aggregate_energy"]]
   },
   {
     "id": "validate_counter",
     "type": "function",
     "z": "assembly_line_processing",
     "name": "Валідація лічильник",
     "func": "if (!Number.isInteger(msg.payload.value) || msg.payload.value < 0) {\n
msg.is_anomaly = true;\n}\nreturn msg;",
     "x": 540, "y": 220,
     "wires": [["build_point_counter"]]
   },
   {
     "id": "events_handler",
     "type": "function",
     "z": "assembly_line_processing",
     "name": "Обробка подій",
     "func": "msg.severity = msg.payload.severity || 'warning';\nreturn msg;",
     "x": 540, "y": 260,
     "wires": [["alert_router"]]
   },
   {
     "id": "aggregate_manipulator",
     "type": "function",
     "z": "assembly_line_processing",
     "name": "Агрегація 1с",
     "func": "context.set('buffer', context.get('buffer') || []);\nlet buf =
context.get('buffer');\nbuf.push(msg.payload);\nif (buf.length >= 10) {\n  msg.payload =
buf;\n  context.set('buffer', []);\n  return msg;\n}\nreturn null;",
     "x": 740, "y": 60,
     "wires": [["build_point_manipulator"]]
   },
   {
     "id": "aggregate_conveyor",
     "type": "function",
     "z": "assembly_line_processing",
     "name": "Агрегація 1с",
     "func": "context.set('buffer', context.get('buffer') || []);\nlet buf =
context.get('buffer');\nbuf.push(msg.payload.value);\nif (buf.length >= 5) {\n
msg.payload.value = buf.reduce((a,b)=>a+b, 0) / buf.length;\n  context.set('buffer', []);\n
return msg;\n}\nreturn null;",
     "x": 740, "y": 100,
     "wires": [["build_point_conveyor"]]
   },
   {
     "id": "aggregate_energy",
     "type": "function",
     "z": "assembly_line_processing",
     "name": "Агрегація 1с",
     "func": "return msg;",
     "x": 740, "y": 180,
     "wires": [["build_point_energy"]]
```

```
    },
    {
      "id": "build_point_manipulator",
      "type": "function",
      "z": "assembly_line_processing",
      "name": "InfluxDB point",
      "func": "msg.payload = [{\n  measurement: 'manipulator_telemetry',\n  fields: {
value: msg.payload },\n  tags: { line_id: 'L1', station_id: msg.metadata.station
}\n}];\nreturn msg;",
      "x": 940, "y": 60,
      "wires": [["influxdb_out"]]
    },
    {
      "id": "build_point_conveyor",
      "type": "function",
      "z": "assembly_line_processing",
      "name": "InfluxDB point",
      "func": "msg.payload = [{\n  measurement: 'conveyor_telemetry',\n  fields: { speed:
msg.payload.value },\n  tags: { line_id: 'L1' }\n}];\nreturn msg;",
      "x": 940, "y": 100,
      "wires": [["influxdb_out"]]
    },
    {
      "id": "build_point_vision",
      "type": "function",
      "z": "assembly_line_processing",
      "name": "InfluxDB point",
      "func": "msg.payload = [{\n  measurement: 'vision_results',\n  fields: { result:
msg.payload.value },\n  tags: { line_id: 'L1', station_id: msg.payload.metadata.station
}\n}];\nreturn msg;",
      "x": 940, "y": 140,
      "wires": [["influxdb_out"]]
    },
    {
      "id": "build_point_energy",
      "type": "function",
      "z": "assembly_line_processing",
      "name": "InfluxDB point",
      "func": "msg.payload = [{\n  measurement: 'energy_metrics',\n  fields: { power:
msg.payload.value },\n  tags: { line_id: 'L1', station_id: msg.payload.metadata.station
}\n}];\nreturn msg;",
      "x": 940, "y": 180,
      "wires": [["influxdb_out"]]
    },
    {
      "id": "build_point_counter",
      "type": "function",
      "z": "assembly_line_processing",
      "name": "InfluxDB point",
      "func": "msg.payload = [{\n  measurement: 'production_counters',\n  fields: { count:
msg.payload.value },\n  tags: { line_id: 'L1', station_id: msg.payload.metadata.station
}\n}];\nreturn msg;",
      "x": 940, "y": 220,
      "wires": [["influxdb_out"]]
    },
    {
      "id": "influxdb_out",
      "type": "influxdb out",
      "z": "assembly_line_processing",
      "name": "InfluxDB write",
      "influxdb": "influxdb_v2_cfg",
      "bucket": "assembly_line_telemetry",
      "org": "AssemblyLine",
      "measurement": "",
      "x": 1140, "y": 140,
      "wires": []
    },
    {
      "id": "influxdb_v2_cfg",
```

```
    "type": "influxdb",
    "name": "InfluxDB v2 local",
    "hostname": "influxdb",
    "port": "8086",
    "protocol": "http",
    "usetls": false,
    "credentials": {
      "token": "<INFLUX_TOKEN>"
    }
  },
  {
    "id": "alert_router",
    "type": "switch",
    "z": "assembly_line_processing",
    "name": "Маршрутизація сповіщень",
    "property": "severity",
    "rules": [
      {"t": "eq", "v": "warning", "vt": "str"},
      {"t": "eq", "v": "critical", "vt": "str"}
    ],
    "outputs": 2,
    "x": 760, "y": 260,
    "wires": [["alert_email"], ["alert_email", "alert_telegram"]]
  },
  {
    "id": "alert_email",
    "type": "e-mail",
    "z": "assembly_line_processing",
    "name": "E-mail сповіщення",
    "server": "smtp.example.com",
    "port": "465",
    "secure": true,
    "from": "alerts@assembly-line.local",
    "to": "operator@assembly-line.local",
    "x": 1000, "y": 240,
    "wires": []
  },
  {
    "id": "alert_telegram",
    "type": "telegram sender",
    "z": "assembly_line_processing",
    "name": "Telegram сповіщення",
    "bot": "telegram_bot_cfg",
    "x": 1010, "y": 280,
    "wires": []
  }
]
```

Лістинг Б.2. Файл налаштувань Mosquitto (mosquitto.conf)

Базова конфігурація брокера з підтримкою TLS на порту 8883 та незахищеного локального з’єднання на порту 1883. Активовано режим persistence для збереження неотриманих повідомлень при перезапуску, ведення журналів та обмеження ресурсів.

```
# mosquitto.conf
# Конфігурація Eclipse Mosquitto для системи цифрової тіні

# Persistence
persistence true
persistence_location /mosquitto/data/

# Logging
```

```
log_dest file /mosquitto/log/mosquitto.log
log_dest stdout
log_type error
log_type warning
log_type notice
log_type information
log_timestamp true

# Listeners
listener 1883 0.0.0.0
protocol mqtt

listener 8883 0.0.0.0
protocol mqtt
cafile /mosquitto/certs/ca.crt
certfile /mosquitto/certs/server.crt
keyfile /mosquitto/certs/server.key
require_certificate false
tls_version tlsv1.2

# Security
allow_anonymous false
password_file /mosquitto/config/passwd
acl_file /mosquitto/config/acl.conf

# Performance
max_connections 100
max_inflight_messages 100
max_queued_messages 1000
message_size_limit 65536
```

ДОДАТОК В

Конфігурації дашбордів Grafana та правил сповіщень

У Додатку В наведено експортовані JSON-конфігурації чотирьох розроблених дашбордів Grafana та YAML-конфігурацію правил сповіщень. Усі конфігурації імпортовано до Grafana через меню Dashboards → Import → Upload JSON file. Запити Flux протестовані на реальних даних, отриманих у ході сценарію 1 (нормальна робота лінії).

Рисунок В.1 – Дашборд «Виробничий моніторинг»

Лістинг В.1. JSON-конфігурація дашборду «Виробничий моніторинг»

```
{
  "annotations": { "list": [] },
  "title": "Виробничий моніторинг — Лінія L1",
  "uid": "assembly_line_production",
  "tags": ["assembly_line", "production"],
  "timezone": "Europe/Kyiv",
  "refresh": "5s",
  "time": { "from": "now-1h", "to": "now" },
  "panels": [
    {
      "id": 1,
      "title": "Cycle Time (current)",
      "type": "gauge",
      "gridPos": { "x": 0, "y": 0, "w": 8, "h": 6 },
      "targets": [{
        "refId": "A",
```

```
          "query": "from(bucket: \"assembly_line_telemetry\") |> range(start: -1m) |>
filter(fn: (r) => r._measurement == \"cycle_metrics\" and r._field == \"cycle_time_s\") |>
last()"
        }],
        "fieldConfig": {
          "defaults": {
            "unit": "s",
            "min": 0, "max": 20,
            "thresholds": {
              "mode": "absolute",
              "steps": [
                { "color": "green", "value": 0 },
                { "color": "yellow", "value": 12 },
                { "color": "red", "value": 15 }
              ]
            }
          }
        }
      },
      {
        "id": 2,
        "title": "Throughput за останню годину",
        "type": "stat",
        "gridPos": { "x": 8, "y": 0, "w": 8, "h": 6 },
        "targets": [{
          "refId": "A",
          "query": "from(bucket: \"assembly_line_telemetry\") |> range(start: -1h) |>
filter(fn: (r) => r._measurement == \"production_counters\") |> count()"
        }],
        "fieldConfig": { "defaults": { "unit": "short" } }
      },
      {
        "id": 3,
        "title": "OEE поточний",
        "type": "gauge",
        "gridPos": { "x": 16, "y": 0, "w": 8, "h": 6 },
        "targets": [{
          "refId": "A",
          "query": "from(bucket: \"assembly_line_telemetry\") |> range(start: -5m) |>
filter(fn: (r) => r._measurement == \"oee_metrics\") |> last()"
        }],
        "fieldConfig": {
          "defaults": {
            "unit": "percentunit",
            "min": 0, "max": 1
          }
        }
      },
      {
        "id": 4,
        "title": "Cycle time per cycle (last 1h)",
        "type": "timeseries",
        "gridPos": { "x": 0, "y": 6, "w": 24, "h": 8 },
        "targets": [{
          "refId": "A",
          "query": "from(bucket: \"assembly_line_telemetry\") |> range(start: -1h) |>
filter(fn: (r) => r._measurement == \"cycle_metrics\" and r._field == \"cycle_time_s\")"
        }]
      },
      {
        "id": 5,
        "title": "Throughput by hour (last 24h)",
        "type": "barchart",
        "gridPos": { "x": 0, "y": 14, "w": 12, "h": 8 },
        "targets": [{
          "refId": "A",
          "query": "from(bucket: \"assembly_line_telemetry\") |> range(start: -24h) |>
filter(fn: (r) => r._measurement == \"production_counters\") |> aggregateWindow(every: 1h, fn:
count)"
```

```
        }]
      },
      {
        "id": 6,
        "title": "Cycle time by station (last 7 days)",
        "type": "heatmap",
        "gridPos": { "x": 12, "y": 14, "w": 12, "h": 8 },
        "targets": [{
          "refId": "A",
          "query": "from(bucket: \"assembly_line_archive\") |> range(start: -7d) |>
filter(fn: (r) => r._measurement == \"cycle_metrics\")"
        }]
      }
    ],
    "schemaVersion": 38,
    "version": 194  }
```

Рисунок В.2 – Дашборд «Стан обладнання станцій»

Рисунок В.3 – Дашборд «Аналіз OEE»

Рисунок В.4 – Дашборд «Аварійні події»

Правила сповіщень налаштовуються через provisioning Grafana 10.x. Файл розташовується у ./grafana/provisioning/alerting/assembly_line.yaml та автоматично завантажується при запуску контейнера.

Лістинг В.2. Правила сповіщень Grafana (YAML provisioning)

```yaml
# Правила сповіщень Grafana 10.x (YAML формат для provisioning)

groups:
  - orgId: 1
    name: assembly_line_critical
    folder: AssemblyLine
    interval: 30s
    rules:
      - uid: alert_manipulator_temp_high
        title: "Перегрів маніпулятора станції 3"
        condition: B
        data:
          - refId: A
            datasourceUid: influxdb_uid
            model:
              query: |
                from(bucket: "assembly_line_telemetry")
                  |> range(start: -2m)
                  |> filter(fn: (r) => r._measurement == "manipulator_telemetry"
                                  and r._field == "temperature")
                  |> mean()
          - refId: B
            datasourceUid: __expr__
            model:
              type: threshold
              conditions:
                - evaluator:
                    params: [65]
                    type: gt
        noDataState: NoData
```

```
      execErrState: Alerting
      for: 1m
      annotations:
        summary: "Температура маніпулятора станції 3 перевищила 65 градусів"
      labels:
        severity: warning
        channel: email,telegram

    - uid: alert_cycle_time_high
      title: "Перевищення часу циклу"
      condition: B
      data:
        - refId: A
          datasourceUid: influxdb_uid
          model:
            query: |
              from(bucket: "assembly_line_telemetry")
                |> range(start: -5m)
                |> filter(fn: (r) => r._measurement == "cycle_metrics"
                                  and r._field == "cycle_time_s")
                |> mean()
        - refId: B
          datasourceUid: __expr__
          model:
            type: threshold
            conditions:
              - evaluator:
                  params: [12.0]
                  type: gt
      for: 2m
      labels:
        severity: warning

    - uid: alert_conveyor_stopped
      title: "Раптова зупинка конвеєра"
      condition: B
      data:
        - refId: A
          datasourceUid: influxdb_uid
          model:
            query: |
              from(bucket: "assembly_line_telemetry")
                |> range(start: -30s)
                |> filter(fn: (r) => r._measurement == "conveyor_telemetry"
                                  and r._field == "speed")
                |> last()
        - refId: B
          datasourceUid: __expr__
          model:
            type: threshold
            conditions:
              - evaluator:
                  params: [0.05]
                  type: lt
      for: 10s
      labels:
        severity: critical
```

ДОДАТОК Г

Технічна документація розгортання системи

Додаток Г містить технічну документацію розгортання системи цифрової тіні: повний docker-compose.yml файл (лістинг Г.1), схему мережевих з'єднань між контейнерами (рисунок Г.1), таблицю налаштувань кожного компонента (таблиця Г.1) та послідовність команд для першого розгортання.

Файл описує чотири сервіси: Mosquitto MQTT-брокер, Node-RED для обробки потоків, InfluxDB для зберігання часових рядів, Grafana для візуалізації. Усі сервіси об'єднані спільною Docker-мережею assembly_line_network з власною підмережею 172.28.0.0/16. Конфігураційні файли та дані змонтовано у томах для збереження між перезапусками контейнерів.

Лістинг Г.1. docker-compose.yml для розгортання повного стеку

```
# docker-compose.yml
# Повний стек для системи цифрової тіні автоматизованої лінії складання
# Запуск: docker compose up -d
# Перевірка статусу: docker compose ps
# Версія Compose: v2 (без поля version)

services:
  mosquitto:
    image: eclipse-mosquitto:2.0
    container_name: ds_mosquitto
    restart: unless-stopped
    ports:
      - "1883:1883"
      - "8883:8883"
    volumes:
      - ./mosquitto/config:/mosquitto/config
      - ./mosquitto/data:/mosquitto/data
      - ./mosquitto/log:/mosquitto/log
      - ./mosquitto/certs:/mosquitto/certs:ro
    networks:
      - ds_internal
    healthcheck:
      test: ["CMD-SHELL", "mosquitto_sub -t '$$SYS/#' -C 1 -E -i healthcheck"]
      interval: 30s
      timeout: 10s
      retries: 3

  node-red:
    image: nodered/node-red:3.1
    container_name: ds_nodered
    restart: unless-stopped
    user: "1000:1000"
    ports:
      - "1880:1880"
    volumes:
```

```
      - ./node-red/data:/data
    environment:
      - TZ=Europe/Kyiv
      - NODE_RED_CREDENTIAL_SECRET=change_me_in_production
    depends_on:
      mosquitto:
        condition: service_healthy
      influxdb:
        condition: service_healthy
    networks:
      - ds_internal

  influxdb:
    image: influxdb:2.7
    container_name: ds_influxdb
    restart: unless-stopped
    ports:
      - "8086:8086"
    volumes:
      - ./influxdb/data:/var/lib/influxdb2
      - ./influxdb/config:/etc/influxdb2
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=change_me_in_production
      - DOCKER_INFLUXDB_INIT_ORG=AssemblyLine
      - DOCKER_INFLUXDB_INIT_BUCKET=assembly_line_telemetry
      - DOCKER_INFLUXDB_INIT_RETENTION=7d
      - DOCKER_INFLUXDB_INIT_ADMIN_TOKEN=local_dev_token_change_me
    networks:
      - ds_internal
    healthcheck:
      test: ["CMD", "influx", "ping"]
      interval: 30s
      timeout: 10s
      retries: 3

  grafana:
    image: grafana/grafana:10.4.0
    container_name: ds_grafana
    restart: unless-stopped
    ports:
      - "3000:3000"
    volumes:
      - ./grafana/data:/var/lib/grafana
      - ./grafana/provisioning:/etc/grafana/provisioning
      - ./grafana/dashboards:/var/lib/grafana/dashboards
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=change_me_in_production
      - GF_USERS_ALLOW_SIGN_UP=false
      - GF_INSTALL_PLUGINS=grafana-clock-panel,grafana-piechart-panel
    depends_on:
      influxdb:
        condition: service_healthy
    networks:
      - ds_internal

networks:
  ds_internal:
    driver: bridge
    name: assembly_line_network
```

```
    ipam:
      config:
        - subnet: 172.28.0.0/16

# Інструкція з розгортання:
# 1) git clone <repo> && cd assembly-line-digital-shadow
# 2) cp .env.example .env  (заповнити паролі та токени)
# 3) ./scripts/init_mosquitto_acl.sh  (створення облікових записів)
# 4) docker compose up -d
# 5) Перейти на http://localhost:3000 (Grafana, admin / пароль з .env)
# 6) Імпортувати дашборди з ./grafana/dashboards/*.json
# 7) Запустити CoppeliaSim і Python-bridge: python mqtt_bridge.py
```

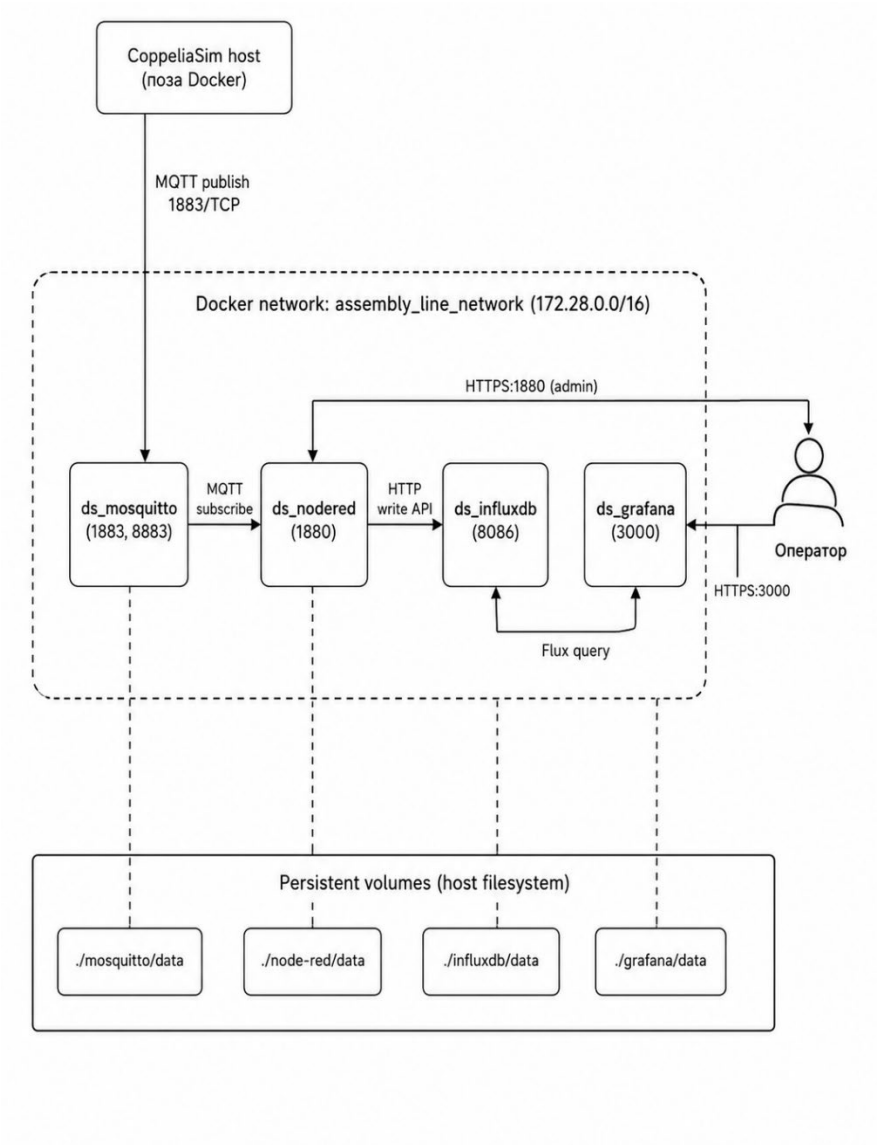


Рисунок Г.1 – Схема мережевих з’єднань Docker-контейнерів

Таблиця Г.1 – Налаштування компонентів стеку

Компонент	Порт	Том	Документація
Mosquitto 2.0	1883/8883	./mosquitto	https://mosquitto.org/documentation
Node-RED 3.1	1880	./node-red/data	https://nodered.org/docs
InfluxDB 2.7	8086	./influxdb/data	https://docs.influxdata.com
Grafana 10.4	3000	./grafana/data	https://grafana.com/docs

Послідовність першого розгортання системи на чистому сервері: 1) клонувати репозиторій з кодом проєкту; 2) скопіювати .env.example до .env та заповнити паролі та токени; 3) запустити скрипт ініціалізації облікових записів Mosquitto; 4) виконати команду docker compose up -d; 5) дочекатися статусу healthy для всіх контейнерів; 6) перейти на http://localhost:3000 та увійти у Grafana; 7) імпортувати дашборди з директорії ./grafana/dashboards; 8) запустити CoppeliaSim і Python-bridge для початку публікації даних. Загальний час розгортання на сучасному обладнанні становить близько п’ятнадцяти хвилин.