

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)
магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ПРОГРАМНИХ РЕАЛІЗАЦІЙ ЦИФРОВИХ
ДВІЙНИКІВ З ВИКОРИСТАННЯМ МОДЕЛЬНО ОРІЄНТОВАНОГО
ПРОЄКТУВАННЯ

(назва теми)

RESEARCH OF SOFTWARE IMPLEMENTATION OF DIGITAL TWINS
USING MODEL-BASED DESIGN

Виконав(ла): студент(ка) 2 курсу, групи КНТ-123м
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного програмного забезпечення
КОСТЮК В.Г..

(ПРИЗВИЩЕ та ініціали)

Керівник ТАБУНЩИК Г.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ШИТІКОВА О.В.

(ПРИЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н., проф.
Сергій СУББОТІН
« » 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОСТЮК Валентини Григорівни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження програмних реалізацій цифрових
двійників з використанням модельно орієнтованого
проектування

Research of Software Implementation of Digital Twins Using Model-based
Design

керівник проєкту (роботи) к.т.н., професор, ТАБУНЩИК Галина
Володимирівна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «07» жовтня 2024 року №398
2. Строк подання студентом проєкту (роботи) 9 грудня 2024
року

3. Вихідні дані до проєкту (роботи) рекомендована література,
технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно
розробити) 1. Аналіз проблеми та постановка завдань дослідження 2.
Матеріали та методи 3. Архітектури цифрового двійника на основі модельно-
орієнтованого проектування. 4. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень,
кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–4 Основна частина	ТАБУНЩИК Г.В., професор		
Нормоконтроль	ДЕЙНЕГА Л.Ю., старший викладач		

7. Дата видачі завдання « 4 » вересня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи та підбір літератури.	1 тиждень	Розділ 1
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Аналіз матеріалів та методів	4-5 тижні	Розділ 2
4	Вибір матеріалів та методів	6-8 тижні	Розділ 2
5	Застосування архітектури цифрового дівйника	9 тиждень	Розділ 3
6	Формування висновків та написання пояснювальної записки	10-11 тижні	Додатки
7	Нормоконтроль та рецензування.	12 тиждень	
8	Захист роботи	12 тиждень	

Студент(ка)

_____ Валентина КОСТЮК
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Галина ТАБУНЩИК
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до роботи магістра: 76 с., 33 рис., 1 дод.,
22 джерел.

**ЦИФРОВИЙ ДВІЙНИК, МОДЕЛЬНО-ОРІЄНТОВАНЕ
ПРОЄКТУВАННЯ, АРХІТЕКТУРА, ФІЗИЧНА СИСТЕМА.**

Об'єкт дослідження – програмна реалізація цифрових двійників

Предмет дослідження – модельно-орієнтоване проєктування реалізації
цифрових двійників

Мета роботи – покращення архітектури програмного забезпечення
цифрових двійників, за рахунок використання модельно-орієнтованого
проєктування.

Матеріали, методи та технічні засоби: систематичних аналіз
літературних джерел, модельно-орієнтоване проєктування, об'єктно-
орієнтоване проєктування.

Результати – дослідженні програмні засоби проєктування та
програмної реалізації цифрових двійників, розроблено підхід до комбінації
програмних заходів засобів для розроблення цифрових двійників широкого
призначення.

Галузь використання – цифрові двійники широкого призначення

ABSTRACT

Explanatory note to the master's thesis: 76 p., 33 fig., 1 appl., 22 sources,.

DIGITAL TWIN, MODEL-ORIENTED DESIGN, ARCHITECTURE,
PHYSICAL SYSTEM.

Object of research – software implementation of digital twins

Subject of research – model-oriented design of implementation of digital
twins

Purpose of work – improvement of software architecture of digital twins,
through the use of model-oriented design.

Materials, methods and technical means: systematic analysis of literary
sources, model-oriented design, object-oriented design.

Results – research of software tools for design and software implementation
of digital twins, an approach to a combination of software measures for the
development of digital twins of a wide range of purposes is developed.

Field of application – general purpose digital twins

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ	8
ВСТУП	9
1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ...	11
1.1 Цифрові двійники	11
1.1.1 Визначення цифрових двійників.....	11
1.1.2 Цифрова модель.....	11
1.1.3 Цифрова тінь	12
1.1.4 Цифровий двійник	12
1.1.5 Підходи до моделювання цифрових двійників	13
1.2 Модельне- орієнтоване проектування.....	14
1.3 Архітектура цифрових двійників.....	19
1.4 Постановка завдання до роботи.....	28
2 МАТЕРІАЛИ ТА МЕТОДИ.....	30
2.1 Аналіз платформ для цифрових двійників.....	30
2.2 Аналіз фреймворків для цифрових двійників	34
2.2.1 Фреймворк EclipseDitto	36
2.2.2 Фреймворк Eclipse Hono.....	39
2.3 Модельно орієнтоване проектування цифрових двійників з використанням Matlab.....	53
3 АРХІТЕКТУРИ ЦИФРОВОГО ДВІЙНИКА НА ОСНОВІ МОДЕЛЬНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ	55
3.1 Стек технологій.....	55
3.2 Приклад архітектури цифрового двійника.....	57
ВИСНОВКИ.....	63
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	64

ПЕРЕЛІК СКОРОЧЕНЬ

5D-DT – Five-Dimensional Digital Twin;
 ADAS Advanced driver-assistance systems;
 AI – AI Artificial intelligence;
 AIOT поєднання AI та IoT;
 API – Application Programming Interface;
 CoAP – Constrained Application Protocol;
 CPS – Cyber–Physical System;
 DD–Data Driven;
 GDTA – Generic Digital Twin Architecture;
 IoT – Internet of Things;
 IIoT – Industrial Internet of Things;
 IP – Internet Protocol;
 IMT – Інтелектуальний обробний інструмент
 MD–Model Driven;
 GDTA– Generic Digital Twin Architecture
 LCC – Life-Cycle Cost.
 OPC UA – Open Platform Communications Unified Architecture
 MBSE – Model-based systems engineering
 RAMI – Reference Architectural Model Industrie;
 TCP – Transmission Control Protocol
 REST – Representational State Transfer
 VE – Virtual Equipment
 WT – Wind Turbine
 ЦД – цифровий двійник
 ЦТ –цифрова тінь
 ЦМ – цифрова модель

ВСТУП

Комп'ютерне моделювання існує вже багато років, але воно було обмежено можливостями математичного моделювання об'єктів реального світу. Інструменти симуляції предметної області були розроблені в середині 80-х, що дозволило розробляти нові системи на основі моделювання на початку 2000-х. Завдяки прогресу в галузі інформаційних технологій моделювання перейшло на новий рівень, рівень створення цифрових двійників.

Наступний крок в інформаційних системах відбувся коли програмне забезпечення для моделювання почало поєднувати свої моделі з даними, отриманими в реальному часі. І тут виникає завдання перетворення цих даних, отриманих в великій кількості у знання, для чого вже використовуються методи аналізу даних та методи машинного навчання. Цифрові двійники є сучасним трендом, що поєднує фізичні пристрої, фізичні та математичні моделі та сучасні методи обробки даних [1].

Модельно-орієнтоване проєктування (англ. Model-based design, MBD) — це математичний та наочний метод розв'язання проблем, пов'язаних із проєктуванням комплексного управління, обробкою сигналів та систем зв'язку. Він використовується в багатьох системах управління рухом, промисловому обладнанні, аерокосмічній та автомобільній промисловості. І дослідження використання цього метода для створення цифрових двійників є актуальним завданням.

Мета роботи — удосконалення процесу моделювання цифрових двійників, за рахунок використання модельно-орієнтованого проєктування.

Завдання роботи:

- провести аналіз предметної області;
- визначити набір відкритих та пропрієтарних програмних засобів для розроблення цифрових двійників;

- провести проєктування ланцюга програмних засобів для створення цифрових двійників на основі модельно-орієнтованого проєктування;
- верифікувати розроблений ланцюг програмних засобів .

Практична значимість розробки: дозволяє робити мережі з зазначеними показниками.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Цифрові двійники

1.1.1Визначення цифрових двійників

Більшість існуючих літературних джерел зосереджені на створенні цифрового уявлення певної фізичної системи. Тому визначення терміну цифровий двійник та методів його використання дуже розпливчасте розуміння [2]. Найбільш популярне визначення було дано Глессеном, Старгелем у 2012 році: «цифровий двійник — це інтегроване мультіфізичне, мультимасштабне, ймовірнісне моделювання складного об'єкту, яке використовує найкращі доступні фізичні моделі, які оновлюються на основі даних отриманих з різноманітних датчиків, з метою відображення життєвого циклу відповідного двійника» [3]. Іншими словами ЦД — це точне й сучасне цифрове уявлення будь-якого фізичного об'єкта. Цифрове уявлення будь-якої сутності реального світу можна класифікувати на 3 основні класи - цифрові двійники, цифрові моделі та цифрові тіні [1]. Класи визначаються на основі типу потоку даних між реальною сутністю (фізичним об'єктом) і цифровою сутністю (моделлю).

1.1.2 Цифрова модель

Цифрова модель (ЦМ) — це, по суті, модель, що розроблена шляхом спостереження за властивостями фізичного об'єкта (рис.1.1). Це передбачає вручну оновлення моделі при будь-яких змінах у її стані чи дизайні. Немає автоматизованого потоку даних між фізичним об'єктом і цифровим об'єктом. Це означає, що будь-які зміни у фізичному об'єкті реального світу не будуть автоматично оновлюватися/відтворюватися у його цифровому представленні. Таким чином, ЦМ виявляється неточним (і не сучасним) представленням будь-якої динамічної системи, чий стан, форма, розмір або складність змінюються з часом [2].

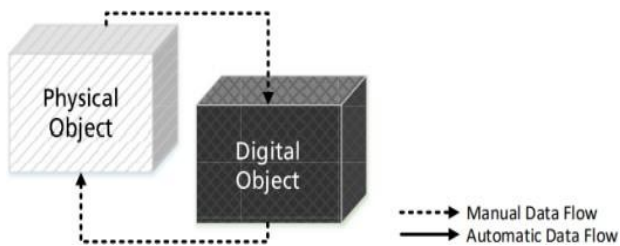


Рисунок 1.1 — Цифрова модель [2]

1.1.3 Цифрова тінь

Додаючи до визначення ЦМ, цифрова тінь (ЦТ) має автоматизований обмін даними між фізичним об'єктом і цифровим об'єктом. Такий обмін даними є одним із способів зворотнього зв'язку. Наприклад, якщо є модель електродвигуна, то положення ротора в цифровому об'єкті можна автоматично оновлювати за допомогою даних датчика положення (кодер повороту) фізичного об'єкта. Але будь-які зміни, внесені в цифровий об'єкт, не відтворюватимуться автоматично в реальному світі (рис.1.2) [2].

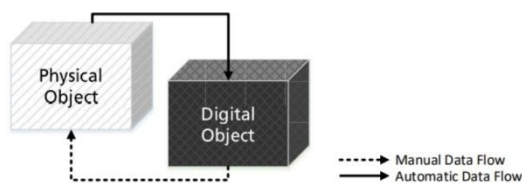


Рисунок 1.2— Цифрова тінь [2]

1.1.4 Цифровий двійник

За визначенням, цифрові двійники повинні сприяти двосторонньому автоматичному обміну даними між фізичними та цифровими об'єктами. Це означає, що ЦМ, що є повністю автономною та не потребує ручного втручання після розробки, може називатися ЦД. Після розробки ЦД може автоматично адаптуватися до будь-яких змін у стані, формі, розмірі та складності свого аналога в реальному світі [2].

На цьому етапі якість ЦД можна визначити за параметром – відповідність. Відповідність можна визначити як рівень властивостей, що копіюються або відображаються (рис 1.3). Точність у ЦД означає, наскільки швидко й точно зміни у фізичному об’єкті відтворюються в цифровому об’єкті.

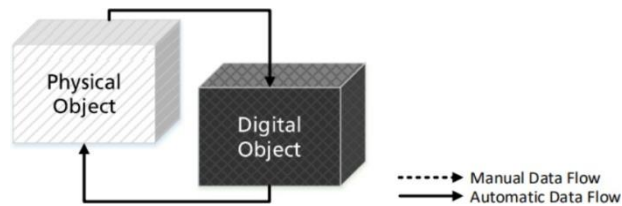


Рисунок 1.3 — Цифровий двійник [2]

1.1.5 Підходи до моделювання цифрових двійників

Процес створення ЦД для будь-якої системи, пристроя чи компонента починається зі створення його точного цифрового уявлення, наприклад за допомогою технік прототипуанн. Існує в основному 3 підходи, ідентифіковані під час створення моделі ЦД, а саме (рис.1.4) [4]:

- підхід, керований даними (КД) – спирається на історичні дані, що до об’єкта;
- модельний підхід (МД) - спирається на розуміння системи та її математичної моделі;
- Гібридний підхід до моделювання – інтеграція підходів, керованих даними та моделлю.

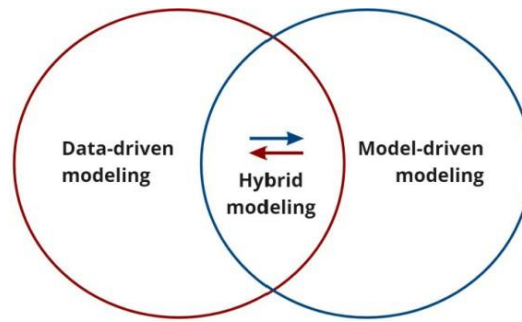


Рисунок 1.4 — Підходи до моделювання DT [4]

Обидві КД і МД підходи мають свої переваги та недоліки. Тому поєднання обох може призвести до більш ефективного рішення найбільш бажаними властивостями обох підходів [4] .

1.2 Модельне- орієнтоване проектування

Модельно-орієнтоване проєктування (англ. Model-based design, MBD) — це математичний та наочний метод розв'язання проблем, пов'язаних із проєктуванням комплексного управління, обробкою сигналів та систем зв'язку. Він використовується в багатьох системах управління рухом, промисловому обладнанні, аерокосмічній та автомобільній промисловості. Модельно-орієнтоване проєктування — це методологія, яка застосовується при розробці вбудованого програмного забезпечення.

Модельно-орієнтоване проєктування значно відрізняється від традиційної методології проєктування. Замість використання складних структур і великого програмного коду, проєктувальники можуть використовувати модельно-орієнтоване проєктування для визначення моделей об'єкта управління з розширеними функціональними характеристиками, використовуючи будівельні блоки безперервного і дискретного часу. Ці побудовані моделі, які використовуються разом з інструментами моделювання, можуть привести до швидкого створення прототипів, тестування програмного забезпечення та перевірки. Не тільки покращено

процес тестування та верифікації, але також, у деяких випадках, моделювання апаратного забезпечення в циклі може бути використане з новою парадигмою проєктування для проведення тестування динамічних впливів на систему швидше та набагато ефективніше, ніж традиційна методологія проєктування.

Модельно-орієнтоване проєктування забезпечує багатодоменне інструментальне середовище та об'єднує всі фази розробки. Загалом існує шість кроків моделювання будь-якої системи:

- Визначення системи;
- Ідентифікація компонентів системи;
- Моделювання системи за допомогою рівнянь;
- Побудова моделі;
- Запуск моделювання;
- Перевірка результатів моделювання.

Ми виконуємо перші три кроки цього процесу поза програмним середовищем, перш ніж розпочати створення нашої моделі. Головним чином ці перші три кроки є важливими, і багато людей роблять помилки на цих кроках. Нарешті, коли система моделюється за допомогою рівнянь, наступна побудова моделі є більш-менш рутинною операцією. Важливо сказати, що кожна модель не ідеальна, і кожного разу ми повинні нехтувати будь-якими моментами та спрощувати опис системи. Загальний процес вимагає досвіду. Коли ми будемо намагатися створити абсолютно ідеальну модель, результатом цього буде дуже складна модель і повільне моделювання. Визначення системи. Першим кроком у моделюванні динамічної системи є повне визначення системи. (рис 1.5) [5].

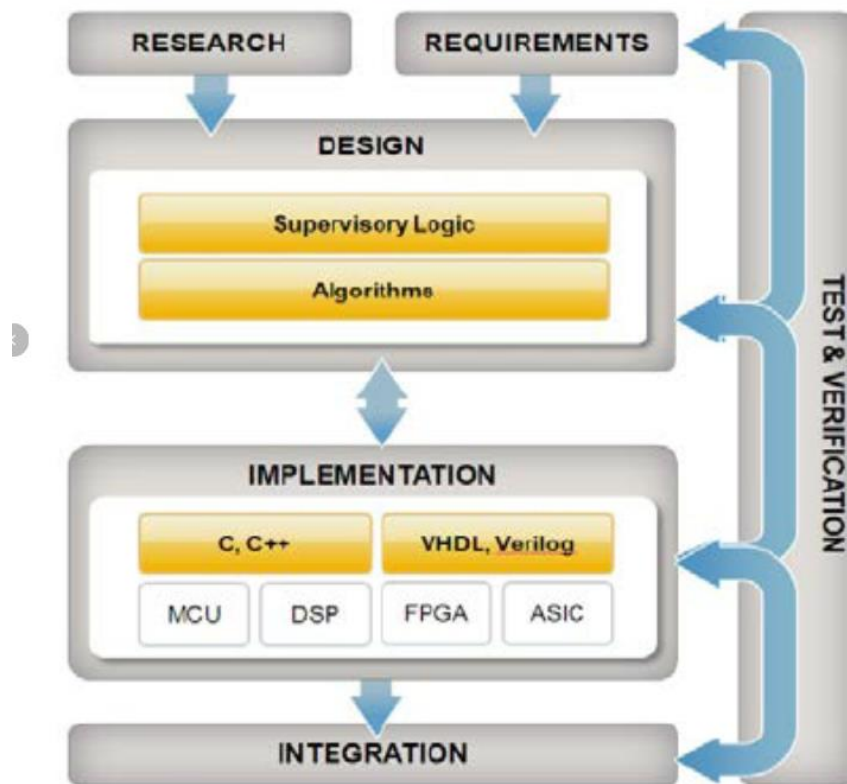


Рисунок 1.5 — Модельно-орієнтоване проектування [5]

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) — платформа та середовище розробки для візуальної мови програмування компанії National Instruments. Метою даної мови є автоматизація використання обчислювального та вимірювального лабораторного обладнання. LabVIEW зазвичай використовується для даних, управління приладами і в промисловій автоматизації на різних операційних платформах, включаючи Microsoft Windows, UNIX, Linux, Mac OSX [6].

Переваги:

- LabVIEW дозволяє здійснювати доступ до великої кількості приладів через вбудовані драйвери.
- Спрощує програмування для непрофесійних програмістів.
- До ряду приладів, зокрема, розроблених National Instruments, уже надаються готові віртуальні інструменти.
- Програми є незалежними від платформи.

- Наявна велика кількість функцій для збору даних, обчислень, генерації сигналів, аналізу тощо. Також, наявна велика кількість графічних елементів для реалізації зручного інтерфейсу користувача. Наявний додатковий програмно-текстовий компонент для проведення обчислень.

- Різні частини блок-діаграми можуть виконуватися паралельно.
- Наявні значна кількість документації.

Програми в LabVIEW називаються Віртуальними Інструментами (VI) або Віртуальними приладами. Всі дії зводяться до простої побудови структурної схеми в інтерактивній графічній системі з набором всіх необхідних бібліотечних образів, з яких збираються об'єкти. Розробка програми в LabView відбувається одночасно в двох вікнах. На передній панелі створюється графічний інтерфейс програми і паралельно ведеться зв'язка інтерфейсу з власне програмою, яка створюється за допомогою спеціальних блоків. Таким чином графічний код програми має вигляд специфічної блок-діаграми. При створенні блок-діаграми програміст з'єднує різноманітні вузли-функції, малюючи лінію-провід. Таким чином створюється потік даних. Проводами передаються змінні і вузол виконує свою роботу, як тільки усі необхідні дані будуть доступні на його входах. Таким чином виконання програми на непов'язаних один з одним вузлах може іти паралельно.

Інтеграція віртуальної системи може допомогти вам зменшити залежність від апаратного забезпечення прототипу та надати всім інженерам доступ до системи практично на будь-якому етапі циклу розробки продукту. Ви можете використовувати Simulink® для моделювання, імітації та аналізу складних віртуальних систем, що складаються з фізичного обладнання, вбудованого програмного забезпечення, алгоритмів і середовища, в якому працює система.

MATLAB&Simulink призначений для проектування на основі моделі. Від концепції до роботи. Щоб трансформувати розробку складних систем,

компанії-лідери ринку застосовують модельно-орієнтоване проектування, систематично використовуючи моделі протягом усього процесу. Використовуйте віртуальну модель для симуляції та тестування системи на ранній стадії та часто Перевірте свій дизайн за допомогою фізичних моделей, тестування обладнання в циклі та швидкого прототипування Створюйте робочий код C, C++, CUDA, PLC, Verilog і VHDL і розгортайте його безпосередньо у вашій вбудованій системі Підтримувати цифровий потік із можливістю відстеження через вимоги, архітектуру системи, дизайн компонентів, код і тести Поширюйте моделі на діючі системи для виконання прогнозного технічного обслуговування та аналізу несправностей

За допомогою Simulink ви можете:

- Опишіть архітектуру системи за допомогою інтуїтивно зрозумілих моделей архітектури;
- Моделюйте системи, що охоплюють кілька доменів, використовуючи специфічні для домену інструменти та готові блоки;
- Розробляйте зручні для обслуговування великомасштабні моделі з багаторазовими та готовими до запуску компонентами;
- Інтегруйте компоненти від різних команд і інструменти в одну симуляцію на системному рівні легко та надійно;
- Моделюйте та аналізуйте, щоб зрозуміти та підтвердити поведінку системи;
- Паралельно виконуйте масштабні завдання симуляції на багатоядерному робочому столі чи комп'ютерному кластері чи в хмарі;
- Розгортання симуляцій як окремих виконуваних файлів, веб-програм і функціональних макетів (FMU) [7].

Модельно-орієнтоване проектування забезпечує ефективний підхід для встановлення загальної структури для комунікацій протягом усього процесу проектування, підтримуючи цикл розробки (V-модель). Під час проектування систем управління на основі модельно-орієнтованого проектування розглядають чотири основні кроки

1.3 Архітектура цифрових двійників

З точки зору архітектури системи розглядаються п'ять різних підходів до ЦД [8]:

- загальна архітектура ЦД з рівнем ресурсів, рівнем інтеграції, комунікаційним рівнем, інформаційним рівнем, функціональним рівнем, бізнес-рівнем.
- архітектура ЦД з оцінкою витрат життєвого циклу. Ця архітектура складається з двох рівнів – фізичного та DT та двонаправленого зв'язку між ними.
- архітектура ЦД з шістьма рівнями з потоком даних між шарами;
- Azure ЦД — це платформа Інтернету речей, яка дозволяє користувачам створювати цифрове представлення реальних речей, місць, бізнес-процесів і людей. Рішення має містити екземпляр служби Azure Digital Twins; один або кілька клієнтських додатків, які керують екземпляром Azure Digital Twins, налаштовуючи моделі, створюючи топологію та витягуючи статистичні дані з подвійного графіка; один або кілька зовнішніх обчислювальних ресурсів для обробки подій, створених Azure Digital Twins, або підключених джерел даних, таких як пристрої; центр IoT; нижчі послуги;
- специфікація семантичної моделі Робочою групою зі структурування семантичних даних, яка забезпечує концептуальну архітектуру, яка дозволяє об'єднувати дані IT та систем керування для подальшої аналітики.

Розглянемо більш детально загальну архітектуру та шестирівневу архітектуру ЦД.

1.3.1 Загальна архітектура ЦД

Запропонована загальна архітектура цифрового двійника (ЗАЦД) визначає базову структуру та компоненти ЦД без уточнення або прив'язки до певних технологій. Для реалізації ЦД на основі ЗАЦД можна використовувати різні існуючі технології та фреймворки для реалізації його функціональності на різних рівнях (рис 1.6) [9].

Перший рівень – це фізичне представлення, цей рівень містить датчики та розумні пристрої, підключені до шлюзу IoT для зв'язку з рівнем вхідних повідомлень. Внутрішнє сховище фізичного представлення для зберігання локальних даних і передачі їх у пакетах на рівень вхідних повідомлень.

Фізичне представлення ЦД розташоване на рівні активів і відповідає фізичному об'єкту. Він містить фізичні частини елементи ЦД.

Другий рівень - рівень інтеграції. На цьому рівні можна розрізнити дані часу виконання та інженерні дані. Дані під час виконання генеруються датчиками або подіями та представляють поточний стан фізичної сутності. Ці дані дуже динамічні, тому базова інфраструктура має відповідати вимогам конкретної програми, як обробка великих даних або отримання даних в реальному часі.

Третій рівень - комунікаційний - відповідає протоколам промислового зв'язку або промислового Інтернету речей (IIoT). Конкретний протокол залежить від вимог програми, наприклад, можливостей у режимі реального часу або підтримки публікації чи підписки. Таким чином, можна очікувати комбінацію різних протоколів, які робляться прозорими для верхніх рівнів за допомогою Smart Data Service. Крім комунікаційних можливостей, інформаційне моделювання також забезпечує представлення контекстної інформації та доступ до неї.

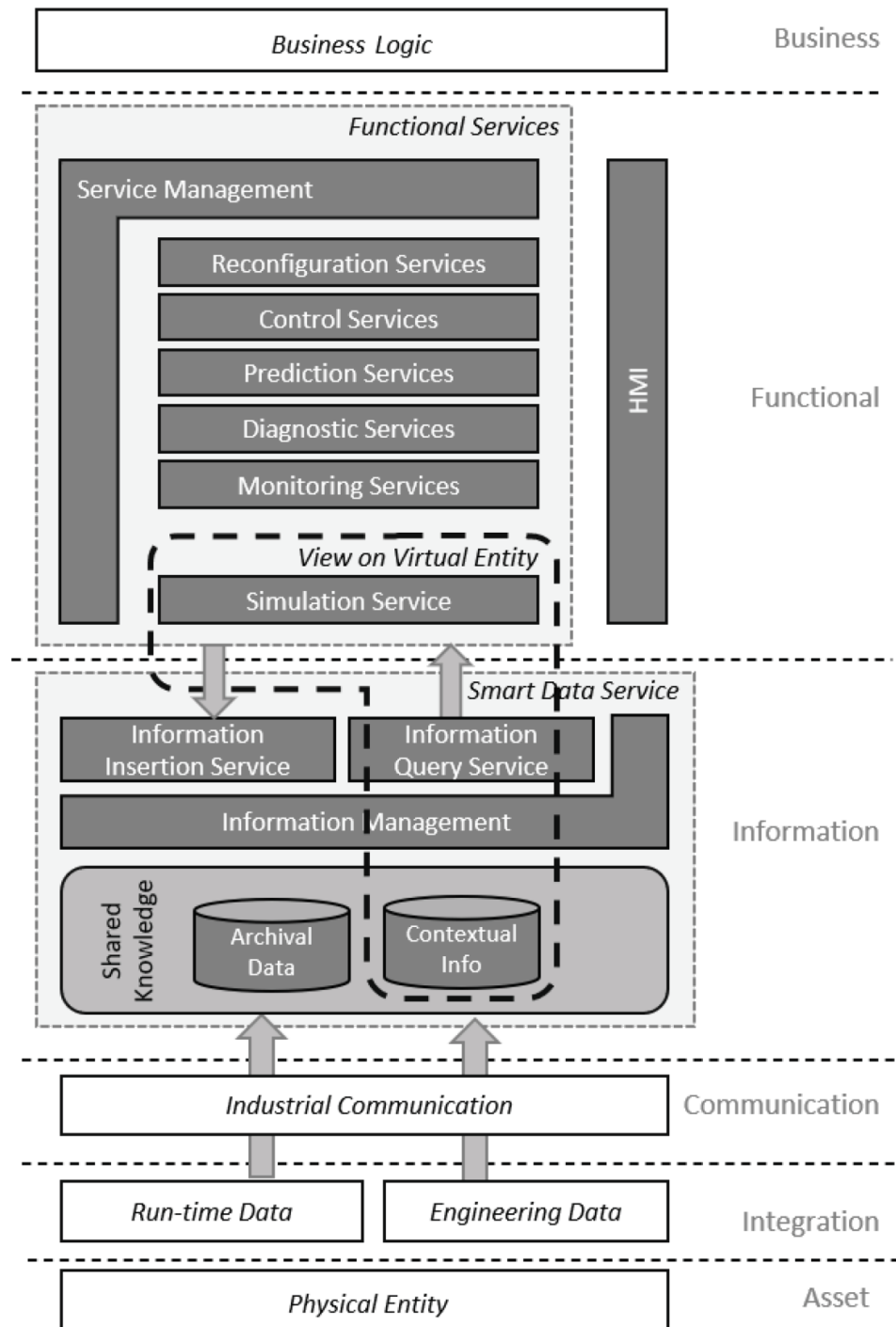


Рисунок 1.6 — Модель архітектури цифрового двійника (ЗАЦД) [9]

Четвертий рівень - інформаційний - у межах цього рівня архітектури ЦД дані збираються та збагачуються семантикою та пов'язуються з іншою контекстною інформацією. Таким чином, на цьому етапі вимір даних стає інформаційним виміром. Центральним компонентом інформаційного рівня є спільна база знань, яка зберігає контекстну інформацію про ресурси та

послуги. Дані пов'язуються з цією інформацією, щоб додати їй семантику. Застосовуючи різні технології моделювання інформації, можна досягти різного рівня семантичної виразності. В ідеалі дані зберігаються в оригінальних форматах у відповідних базах даних. База спільних знань діє як рівень семантичної інтеграції для даних під час виконання та інженерних даних, надаючи доступ до історичних даних (архівів) і зберігаючи контекстну інформацію всередині спільних знань.

П'ятий рівень - функціональний — на цьому рівні реалізується сервісний вимір. Сервісно-орієнтована архітектура використовується для забезпечення слабкого зв'язку та згуртованості певної функціональності [4]. Сервіси ЦД можна згрупувати за їх функціональними можливостями. Визначено п'ять груп сервісів: сервіси моделювання, сервіси моніторингу, сервіси діагностики, сервіси прогнозування, сервіси керування та сервіси реконфігурації. Окрім цих функціональних послуг, компонент керування послугами піклується про реєстрацію послуг, виявлення та отримання інформації про статус певних послуг. Управління ресурсами не обробляється центральним компонентом, оскільки служби повинні самостійно обробляти свої ресурси, полегшуючи спільну базу знань.

Сервіси моделювання є основними сервісами. Як правило, існує не лише одна модель фізичної сутності в ЦД, але набір моделей, які є специфічними для передбачуваної мети та також розвиваються з часом [10]. Таким чином, різні моделі для різних областей, як-от механічна структура, термічна поведінка тощо, можна розміщувати та використовувати в службі моделювання ЦД. Що стосується інших служб, ресурсами має керувати сама служба, а відповідна інформація про моделі має бути доступною через спільну базу знань.

Послуги моніторингу - це елементарні послуги для отримання даних від фізичної особи та спостереження за її поточним станом. Прикладом може бути служба виявлення несправностей, яка може бути реалізована на основі

простих статистичних моделей і вказувати на аномальні умови роботи установки [9].

Діагностичні послуги – це послуги, які підтримують, наприклад, моніторинг стану або аналіз першопричин несправностей. Вони можуть спиратися на базові служби моніторингу в поєднанні з послугами моделювання, щоб отримати більше розуміння поточного стану фізичної особи.

Служби прогнозування важливі для ЦД, щоб приймати рішення на основі інформації про майбутні події. Такі послуги можна використовувати, наприклад, для прогнозного обслуговування або прогнозування споживання енергії. Крім того, можна передбачити зовнішні змінні, такі як виробництво відновлюваної енергії або ціни на енергетичному ринку. Крім того, можна інтегрувати зовнішні служби прогнозування. Результати прогнозування можуть бути використані службами управління для реалізації оптимізованого управління або використані для створення рекомендацій для експлуатаційного персоналу [9].

Служби керування впливають на роботу заводу через рекомендації через людино-машинний інтерфейс (ЛМІ) або прямий доступ до керування процесом. Служби керування з прямим доступом можуть обійти Smart Data Service на інформаційному рівні, щоб змінити стан фізичної сутності без додаткових затримок. Зазвичай служби керування використовують служби моніторингу, діагностики та прогнозування для досягнення оптимізації роботи. Таким чином, стратегія контролю може змінюватися з часом, викликана реконфігурацією Фізичної особи або новими цілями, визначеними бізнес-логікою [9].

Послуга реконфігурації. Реконфігурація означає досить статичну зміну фундаментальних властивостей фізичної сутності самим ЦД. Це має значний вплив на контекстну інформацію всередині Спільних знань. Реконфігурацію можна ініціалізувати за допомогою подій або змінених цілей усередині бізнес-

логіки, яка знаходиться в бізнес-рівні. Служба реконфігурації опікується такими змінами всередині ЦД [9].

Люди повинні бути поінформовані про поточний стан фізичної сутності, а також про сам ДЦ, щоб взаємодіяти з ним. Таким чином, відповідний ЛМІ є дуже важливим для майже кожної служби ЦД.

Бізнес-рівень — на цьому рівні знаходиться бізнес-логіка, яка також може організовувати велику кількість ЦД. Він визначає загальні цілі (наприклад, зменшити ризик простою або витрати), яких слід досягти за допомогою ЦД. Оскільки це значною мірою залежить від бізнес-стратегії, це не має значення для розробки ЗАЦД, а стосується лише конкретних вхідних даних. API gateway— це централізована служба, яка діє як посередник між клієнтами (наприклад, мобільними програмами, веб-сайтами чи іншими програмами) та серверними службами. Він забезпечує єдину точку входу для всіх вхідних запитів і виконує такі завдання, як маршрутизація запитів, автентифікація, обмеження швидкості, переклад протоколів і кешування відповідей. По суті, він діє як воратар, контролюючи доступ до ваших API і сприяючи ефективному спілкуванню між клієнтами та службами.

1.3.2 Шестирівнева цифрову подвійну архітектуру

Шестирівневу подвійну архітектуру для ЦД з потоками даних між шарами відображено на рисунку 1.7. З рисунка видно, як дані надходять із фізичної системи або фізичного двійника (рівень 1) до хмари (рівень 5), де вони зберігаються в сховищі інформації, доступному в кіберпросторі. Інформація також може надходити з хмари до фізичного двійника. Архітектура на четвертому рівні має додаткові функції перетворення даних в знання. Шостий рівень містить програмне забезпечення для моделювання або емуляції та інші програми, які можуть використовувати інформацію з фізичного близнюка, щоб реалізувати очікування систем керування.

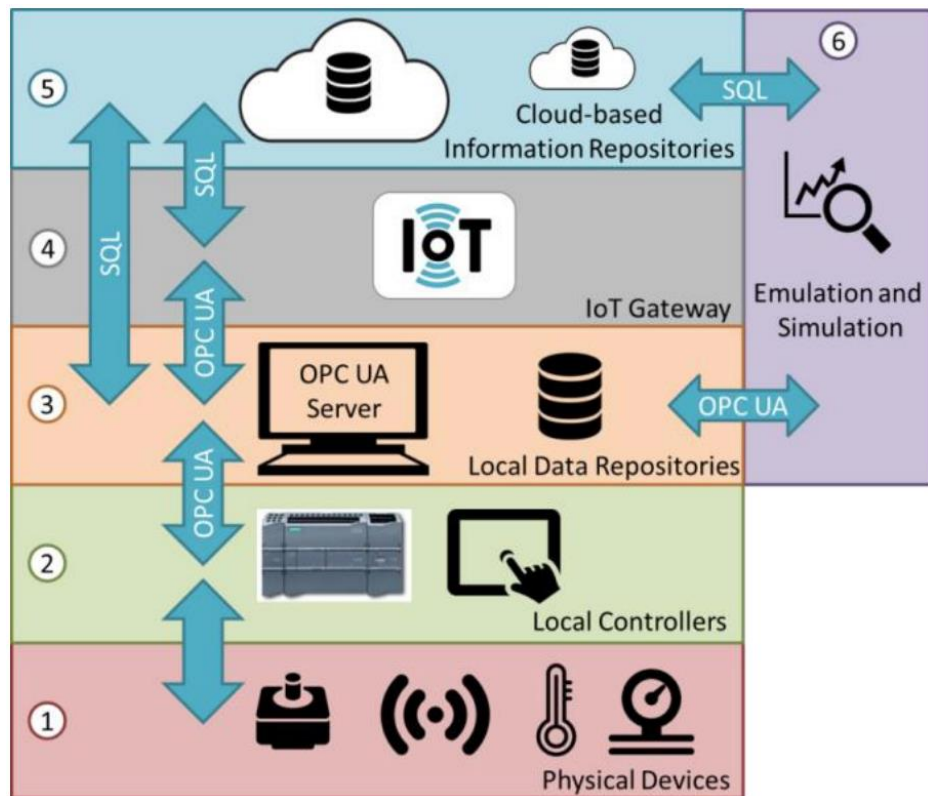


Рисунок 1.7 – Потіки даних для цифрового двійника [11]

Рівні 1 і 2 це фізичний двійник. Рівень 1 містить різні фізичні пристрої, такі як приводи та датчики, які можуть надавати або споживати сигнали, якими обмінюються з локальним контролером. Локальні контролери (рівень 2) вважаються окремим рівнем, оскільки, окрім своєї ролі фізичного двійника, вони можуть використовуватися для надання деяких специфічних функцій для ЦД. Повсюдно поширеним контролером в автоматизації виробництва є програмований логічний контролер, але будь-який контролер, який може взаємодіяти з рівнем 3, підійде для цієї архітектури.

Рівень 3 містить локальні сховища даних, розташовані поблизу фізичного близнюка, такі як сервери OPC UA і локальні бази даних. Будь-який сервер OPC UA, який не залежить від постачальника, можна налаштувати для обміну даними з фізичною системою. Сервери OPC UA здатні спілкуватися з багатьма типами пристроїв, здатних передавати та отримувати дані за допомогою драйверів OPC UA. Оскільки всі основні постачальники контролерів автоматизації надають інтерфейси для OPC UA, такий сервер

забезпечує нейтральний щодо постачальника інтерфейс між фізичним двійником і кіберпростором. Крім того, рівень OPC UA забезпечує інші важливі характеристики, такі як безпека, продуктивність у реальному часі, надійність і глобальне підключення. У деяких складних виробничих системах можуть спільно використовувати базу даних для контролю та синхронізації своєї діяльності. Такі бази даних можна вважати частиною рівня 3, незалежно від того, створені вони спеціально для ЦД чи ні. Дані на Рівні 3 відображатимуть деталі фізичного двійника, але додаткова обробка може бути додана до Рівня 2 спеціально для надання даних для ЦД. Однак, якщо фізичний двійник уже існує, рекомендується додати функціональність, необхідну для цифрового двійника, на Рівні 4, а не змінювати Рівень 2, враховуючи час простою та ризики, пов'язані з модифікацією системи, що працює. Крім того, якщо ЦД націлений на різні архітектури фізичного двійника, ймовірно, краще врахувати відмінності на рівні 4 [11].

Рівень 4 (рисунок 1.7) діє як шлюз між фізичним двійником і підключеним світом. Цей рівень забезпечує перетворення між даними рівня 3 до інформації рівня 5. У деяких ситуаціях архітектуру можна спростити, опустивши рівень 4, при цьому рівні 3 і 5 безпосередньо обмінюються даними один з одним, коли функції, додані шлюзом, не потрібні. Рівень 4 — це спеціально розроблене програмне забезпечення, яке містить клієнт OPC, який інтерфейсує з серверами OPC UA на Рівні 3, а також клієнти баз даних, що взаємодіють із серверами баз даних на Рівнях 3 і 5. Оскільки IoT Gateway взаємодіє як з локальними, так і з хмарними джерелами даних, це зручне місце для додавання графічного інтерфейсу користувача, де можна відстежувати та контролювати деякі з основних операцій цифрового близнюка. Типові ролі Рівня 4 щодо потоку інформації від фізичного близнюка до кіберпростору:

- отримувати інформацію з даних, доступних на Рівні 3, таку як багатовимірна кореляція даних [13] або «розвідка даних», коли різні форми даних аналізуються для прийняття кращих майбутніх рішень;

- вибрати дані для передачі до сховищ даних, щоб уникнути надмірних вимог до бази даних [11];
- передавати в це сховище лише дані або інформацію, що відповідають кожному конкретному сховищу даних на рівні 5;
- запобігання вузьким місцям пропускної здатності шляхом обмеження обсягу даних, що обробляються через мережевий шлюз;
- перетворювати дані з різноманітних подвійних архітектур на інформацію в більш загальному форматі.

Рівень 4 також може відігравати важливу роль у потоці інформації з кіберпростору до фізичного двійника:

- забезпечувати безпеку фізичного близнюка, наприклад, гарантуючи, що фізичний близнюк перебуває у відповідному стані перед тим, як зміни, надані з Рівня 6 через Рівень 5, будуть передані на Рівень 3;
- вирішення конфліктів у даних, що надходять із різних сховищ даних на рівні 6, або коли зміни на рівні 3 і на рівні 6 несумісні.

Для складних виробничих комірок, які містять велику кількість станцій, які є складними самі по собі, архітектура, показана на рисунку 1.15, передбачає, що станції не гарантують наявності власних цифрових двійників, і що бази даних і ОРС-сервери на рівні 3 надають дані, необхідні для шлюзом IoT на рівні клітини. Цього має бути достатньо для компанії, яка використовує виробничий осередок. Однак компанія, яка розробила або обслуговує виробничий осередок, може віддати перевагу ієрархічній структурі, де кожна станція має власний шлюз IoT для підвищення модульності цифрових близнюків. Тоді архітектуру на рис. 3 можна розширити так, щоб шлюз IoT на рівні клітини взаємодівав із шлюзами IoT на рівні станції, а не зі звичайними джерелами даних, показаними на рівні 3 [11].

Рівень 5 містить хмарні сервери баз даних, які діють як інформаційні сховища для інформації фізичного та цифрового двійників. Інформація зазвичай записує історію фізичного двійника та поточний/останній доступний стан фізичного з підтвердженими затримками. Передбачено кілька сховищ,

оскільки різні зацікавлені сторони, ймовірно, матимуть різні інформаційні потреби та права доступу до інформації. Наприклад, розробник фізичного двійника може вимагати доступу до критичних параметрів продуктивності, але може хотіти контролювати доступ до цієї інформації, тоді як завод-виробник може вимагати, щоб інформація про забезпечення якості зберігалася, але хотів би зберегти цю інформацію конфіденційною. Розміщення цих сховищ у хмарі покращує доступність, доступність і підключення цифрового двійника. Спеціалізовані знання, необхідні для управління таким сервером бази даних, беручи до уваги масштабованість, надійність і безпеку, зазвичай недоступні на виробничих підприємствах. Повідомляється, що деякі постачальники засобів автоматизації розробляють такі хмарні сховища, але подібні послуги вже доступні з багатьох інших джерел [11].

У той час як рівні з 1 по 5 забезпечують, у певному сенсі, необхідну інфраструктуру, інтелект цифрового двійника додається на рівні 6. Оскільки цей рівень сильно залежить від фактичної програми, мало що можна вказати в загальній архітектурі. Цей рівень може виконувати будь-яку роль цифрового двійника. Ці ролі, як правило, залежатимуть від наявності доступу до емуляцій (які моделюють поточну поведінку) та симуляцій (які моделюють майбутню чи потенційну поведінку) фізичного двійника. Деякі з ролей вимагатимуть розробки спеціального програмного забезпечення, але деякі можуть бути виконані за допомогою комерційно доступного програмного забезпечення моделювання. Саме програмне забезпечення може бути хмарним або працювати зі звичайного комп'ютера

1.4 Постановка завдання до роботи

Аналіз сучасного стану розроблення цифрових двійників довів, що в їх основі лежить п'яти рівнева архітектура, що широко використовується при реалізації IoT систем. Але для забезпечення передання даних між рівнями

необхідно мати моделі як фізичних моделей, так і цифрових моделей для симуляції та урахування потоків даних в цифровому двійнику.

Ефективним інструментом інтегрування різноманітних систем є модельно орієнтоване моделювання, що широко використовується для проєктування складних об'єктів та систем. Тому наступним кроком є аналіз існуючих засобів створення цифрових двійників та інтеграція їх з засобами модельно-орієнтованого проєктування.

2 МАТЕРІАЛИ ТА МЕТОДИ

2.1 Аналіз платформ для цифрових двійників

Приклади платформ цифрових двійників:

- Altair Engineering: One Total Twin platform Altair One Total Twin platform – платформа, що надає інструментарій для розробки цифрових двійників. та перепозиціонування існуючих продуктів та послуг компанії Altair Engineering, до числа яких входять високопродуктивні обчислювальні кластери, рішення в галузі штучного інтелекту та машинного навчання, а також один із найбільших наборів програмних систем для інженерного аналізу та моделювання;
- Ansys: Ansys Twin Builder Ansys Twin Builder – мультитехнологічна платформа, яка дозволяє інженерам створювати цифрові двійники на основі моделювання (simulation-based digital twins). За твердженням фахівців компанії Ansys, рішення Twin Builder є технологією, що призначена для моделювання, інтеграції та аналізу кіберфізичних систем, до складу яких входить силова електроніка, математично описані фізичні структури та керуюче програмне забезпечення;
- COMSOL: COMSOL Multiphysics COMSOL Multiphysics – платформа для створення розрахункових моделей та додатків для моделювання, розроблена компанією COMSOL. Перша версія платформи була представлена в 1998 році. чисельні моделі, необхідні для подальшої побудови Цифрових двійників. Можливості платформи включають також проведення валідації таких моделей з використанням даних вимірювань, а також методик параметричної оцінки та припасування (parameter estimation), оптимізації та контролю, методики зниження порядку моделей (model order reduction) та побудови зосереджених моделей, які можуть бути використані. для створення та валідації полегшених моделей;

– Dassault Systèmes: 3DEXPERIENCE 3DEXPERIENCE (3DX) – платформа для бізнесу та інновацій, яка виступає як єдине джерело достовірної інформації про всі процеси та операції компанії, за допомогою якої компанія в режимі реального часу отримує цілісне уявлення про власну бізнес-екосистему і реалізується платформа 3DEXPERIENCE розроблена компанією Dassault Systèmes, була запущена в 2012 році. business model до transform a business) ;

– Dassault Systèmes і Bureau Veritas: Veristar AIM3D Veristar AIM3D – програмне забезпечення, яке підтримує повний життєвий цикл і включає тривимірний цифровий двійник об'єкта з інтелектуальною обробкою даних через платформу для спільної роботи та Dassault Systèmes, була запущена в 2017 році. Єдина платформа Veristar AIM3D була розроблена на основі платформи 3DEXPERIENCE спеціально для власників комерційних судів, нафтогазових платформ і т. д. Рішення Veristar AIM3D є єдиним майданчиком управління об'єктами, що підходять бурових вишок, так і їх сукупності;

– General Electric: Predix Platform Predix Platform – промислова програмна платформа, що забезпечує інтеграцію даних, які отримані за допомогою Інтернету речей, а також дозволяє користувачам мати інтерфейс взаємодії з джерелами даних та здійснювати обробку даних, аналітику даних та підтримку промислових додатків. Digital, дочірньою компанією американської корпорації General Electric, у 2013 році для внутрішнього використання в General Electric, пізніше була запущена для використання замовниками та партнерами компанії;

– Hexagon AB / MSC Software: SimManager SimManager – SPDM-система, розроблена компанією MSC Software (з 2017 року є підрозділом компанії Hexagon AB), що є комплексом програмних інструментів, призначених для інженерного аналізу та управління об'єктами комп'ютерного моделювання. представлена 2004 року;

– Microsoft: Azure Digital Twins Azure Digital Twins – платформне рішення, яке дозволяє реалізовувати функції інтернету речей, будувати моделі

фізичних середовищ, створювати просторові граfi відносин та взаємодій між людьми, місцями та пристроями. Платформа Azure Digital Twins була розроблена та запущена компанією Microsoft у тестовому форматі у 2018 році, у 2020 році – стала загальнодоступною. Twins функціонує за моделлю «платформа як послуга» (Platform as a Service, PaaS) Azure Digital Twins інтегрується зі сховищами даних, інструментами аналітики та штучного інтелекту Microsoft Azure, допомагаючи користувачам відстежувати поведінку досліджуваного об'єкта в минулому, а потім передбачати його майбутню поведінку.

Azure Digital Twins зазвичай використовується в поєднанні з іншими службами Azure як частина більшого рішення IoT(рис 2.1) [12].

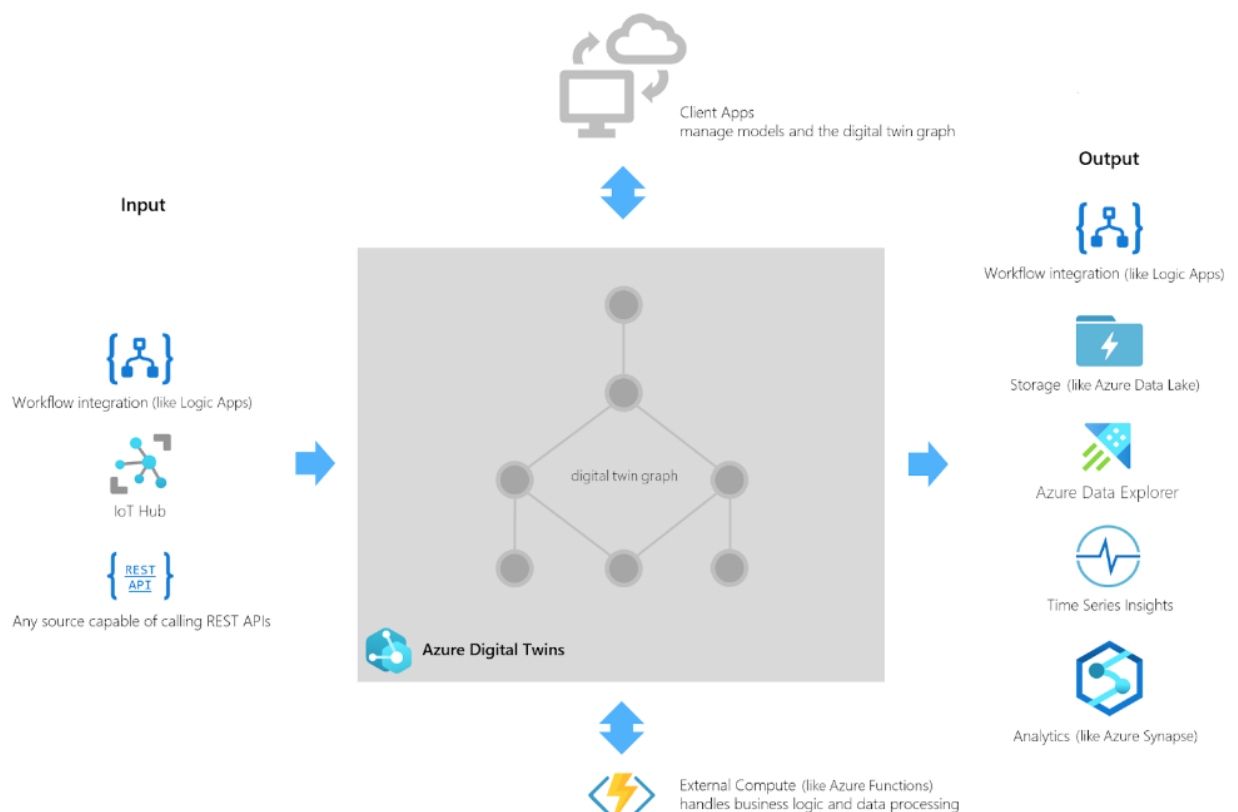


Рисунок 2.1 – Платформа Azure Digital Twins IoT [12].

Azure Digital Twins зазвичай використовується в поєднанні з іншими службами Azure як частина більшого рішення IoT.

Можлива архітектура повного рішення з використанням Azure Digital Twins може містити такі компоненти:

- екземпляр служби Azure Digital Twins. Ця служба зберігає моделі-близнюки та ваш графік-близнюк із його станом і організовує обробку подій;
- одна або кілька клієнтських програм, які керують екземпляром Azure Digital Twins, налаштовуючи моделі, створюючи топологію та витягуючи аналітичні дані з подвійного графіка;
- один або кілька зовнішніх обчислювальних ресурсів для обробки подій, створених Azure Digital Twins, або підключених джерел даних, таких як пристрої. Одним із поширених способів надання обчислювальних ресурсів є функції Azure;
- концентратор Інтернету речей для забезпечення керування пристроями та можливостей потоку даних Інтернету речей;
- низхідні служби для надання таких речей, як інтеграція робочого процесу (як-от Logic Apps), холодне зберігання (як-от Azure Data Lake) або аналітика (як-от Azure Data Explorer або Time Series Insights) [12].

Сітка подій Azure — це повністю керована служба розповсюдження повідомлень Pub Sub, яка повністю масштабується, яка пропонує гнучкі шаблони споживання повідомлень за допомогою протоколів MQTT і HTTP. За допомогою сітки подій Azure можна створювати конвеєри даних із даними пристрою, інтегрувати програми та створювати безсерверні архітектури на основі подій. Сітка подій дозволяє клієнтам публікувати та підписуватись на повідомлення за протоколами MQTT версії 3.1.1 та версії 5.0 для підтримки рішень Інтернету речей (IoT). За допомогою HTTP-протоколу сітка подій дозволяє створювати рішення на основі подій, в яких служба видавця оголошує про зміни стану системи (події) для програм передплатників. Сітка подій може бути налаштована для надсилання подій передплатникам (примусової доставки) або передплатникам можна підключитися до сітки подій для читання подій (доставку за запитом). Служба "Сітка подій" підтримує специфікацію CloudEvents 1.0 для забезпечення взаємодії між

системами. Високорівнева схема сітки подій, що показує видавців та передплатників за допомогою протоколів MQTT та HTTP. Сітка подій Azure — це загальнодоступна служба, яка розгорнута в зонах доступності в усіх регіонах, що їх підтримують (рис 2.2) [12].

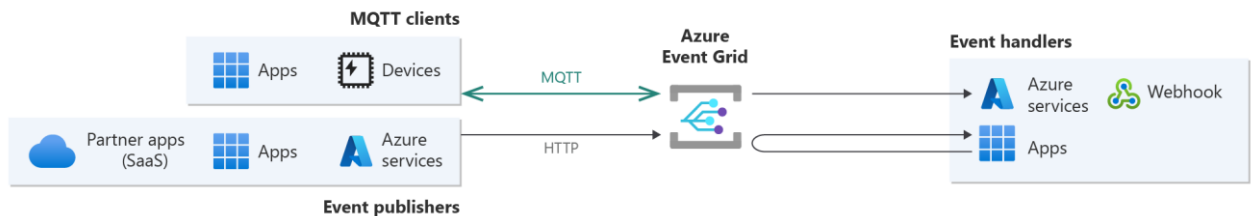


Рисунок 2.2 – Сітка подій Azure [12].

Обмін повідомленнями MQTT. Пристрої та програми Інтернету можуть взаємодіяти один з одним за допомогою протоколу MQTT. конвеєри даних, що починаються з прийому даних з пристроїв Інтернету речей.

Розподіл даних за допомогою режимів відправлення та вилучення У будь-який момент у конвеєрі даних HTTP-додатки можуть використовувати повідомлення за допомогою API відправки або вилучення.

- служби Azure;
- Додатки користувача;
- Системи зовнішнього партнера (SaaS).

Механізм примусової доставки Сітки подій надсилає дані до місць призначення, які включають власні веб-перехоплювачі додатків та служби Azure [12].

2.2 Аналіз фреймворків для цифрових двійників

Серед відкритих рішень для цифрових двійників можна назвати рішення від Digital Twin Consortium: <https://github.com/digitaltwinconsortium>, підтримує використання DTDL (Digital Twin Definition Language) та OPC UA

(Industrial Digital Twins), а також надає глосарій з термінології цифрових двійників: <https://github.com/digitaltwinconsortium/dtc-glossary>. До них відносяться:

- Eclipse Ditto - фреймворк для створення моделей цифрових двійників на основі датчиків інтернету речей (використовується DTDL), дозволяє підключатися до існуючих датчиків, порівнювати стан з очікуваним, додавати аспект прогнозування очікуваного стану. Підтримує конектори на HTTP/SSE, веб-сокети, AMQP, MQTT, Kafka та HTTP Push;
- Eclipse Hoнонадає інтерфейси віддаленого обслуговування для підключення великої кількості пристроїв IoT до серверної частини та взаємодії з ними єдиним способом незалежно від протоколу зв'язку пристрою;
- MCX Framework дозволяє розробляти цифрові двійники для кіберфізичних систем;
- Twinbase використовується для створення поверху Web мережі для об'єднання цифрових двійників з автоматичним розгортанням з github-репозиторіїв;
- Ayла дозволяє об'єднати датчики через HTTPS/MQTT/CoAP із підтримкою інтеграцій з Android, Linux, пристроями Bluetooth Low Energy та хмарними провайдерами;
- iTwin.js використовується для розробки веб-застосунків для інфраструктури цифрових двійників.

Загалом Ditto фокусується на керуванні цифровими двійниками та станом пристроїв, тоді як Hoно надає інфраструктуру для обміну повідомленнями та взаємодії з пристроями. Обидва проекти можуть використовуватись разом для створення комплексних IoT-рішень.

Інфраструктура в контексті IoT (Інтернет речей) та проектів, таких як Eclipse Hoно та Ditto, відноситься до набору технологій, систем та компонентів, які забезпечують функціонування та взаємодію IoT-пристроїв, додатків та сервісів.

Eclipse Hono сам не надає функціональності для роботи з цифровими двійниками (digital twins), але він може бути використаний у поєднанні з іншими інструментами, такими як Eclipse Ditto, для створення комплексних IoT-рішень, які включають цифрові двійники.

2.2.1 Фреймворк Eclipse Ditto

Eclipse Ditto безпосередньо не інтегрує пристрої. Його фокус полягає в наданні API для веб-додатків, мобільних додатків або інших серверних служб.

ЦД відображає фізичні пристрої в хмарі. З одного боку, це дозволяє отримати доступ до певних аспектів пристрою, дозволяючи вам, наприклад, оцінити дані пристрою. З іншого боку, ЦД пропонують можливість надавати додаткові послуги навколо пристрою, наприклад покращувати віртуальне представлення пристрою за допомогою даних про погоду або додавати інформацію про його запасні частини.

Концепція ЦД часто містить також імітацію пристроїв. Ви можете створити цифрового двійника пристрою ще до його виготовлення чи розгортання. Потім ви можете змоделювати пристрій, який використовується, перш ніж його розгорнути в реальному світі, і використовувати API, перш ніж пристрій почне надсилати дані [14].

Схема роботи EclipseDitto зображена на рисунку 2.3

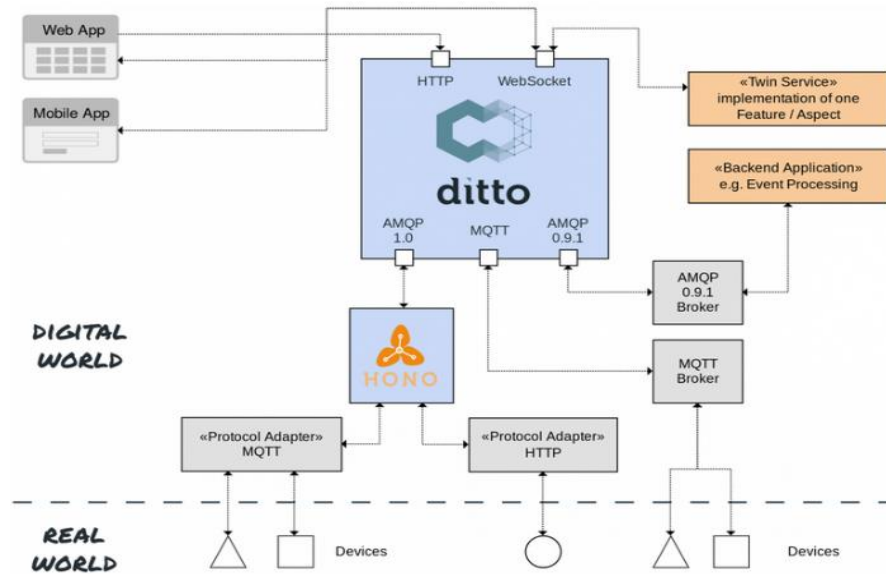


Рисунок 2.3 – Схема роботи EclipseDitto [14]

Загальна мета Eclipse Ditto — створити основу для цифрових двійників. Основна увага приділяється оркестровці між реальними пристроями та їх віртуальними представленнями. Ви також можете покращити ЦД за допомогою API для доступних запасних частин. Архітектура мікросервісів відображена на рис 2.4. Вона демонструє взаємодію мікросервісів, зовнішні залежності та відкриті та споживані інтерфейси

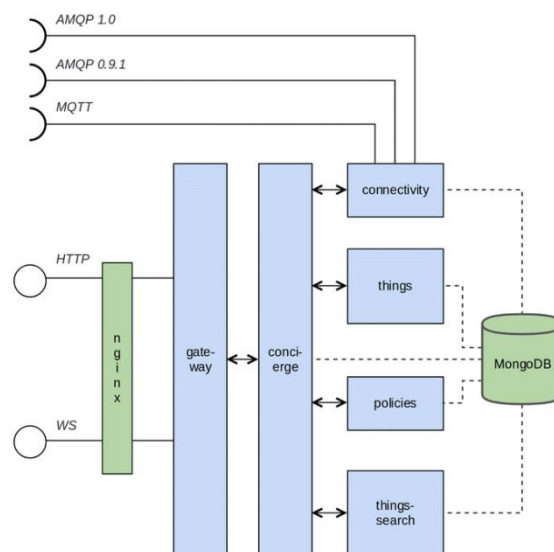


Рисунок 2.4 – Архітектура мікросервісів Eclipse Ditto [14]

Модель Ditto Eclipse Ditto надає структуру, яка дає змогу працювати зі станом цифрових близнюків і керувати ним. Вона будує міст між реальними пристроями IoT та їхніми цифровими двійниками. Розміщуючи це в контексті: Eclipse Ditto не має на меті напряду інтегрувати пристрої. Для цієї мети краще підходять інші варіанти, наприклад Eclipse Hono. Натомість Eclipse Ditto зосереджується на наданні API для веб-додатків, мобільних додатків або інших серверних служб, які використовують API WebSocket, кінцеву точку AMQP 1.0, брокер AMQP 0.9.1 або API HTTP. Нещодавно також було додано підключення через посередника MQTT 3.1.1. Eclipse Ditto має модульну структуру. Вам потрібні лише служба речей і служба шлюзу, а також MongoDB, щоб мати мінімальний час виконання Eclipse Ditto. За потреби до Eclipse Ditto можна додати службу підключення (яка може підключатися до AMQP 1.0, AMQP 0.9.1 або MQTT), службу детальної політики або функцію пошуку. Дві точки зору: двійня проти живого. За своєю суттю цифрове представлення фізичного пристрою існує для того, щоб надати інформацію про стан цього пристрою. Eclipse Ditto наближається до цього за допомогою двох каналів зв'язку: подвійного та прямого каналу.

Подвійна перспектива Eclipse Ditto схожа на базу даних, але вона виходить за рамки: це постійна структура даних, до якої пристрої підключаються та зберігають свій останній зареєстрований стан. Програми можуть зчитувати цей останній повідомлений стан і отримувати сповіщення про зміни. Це можна зробити через відповідну кінцеву точку протоколу. Пристрої також використовують ці кінцеві точки для оновлення свого стану в базі даних або отримання сповіщень, якщо програма хоче щось змінити на пристрої. Ditto забезпечує контроль доступу під час кожного виклику API. На завершення це означає, що Ditto обробляє команди та видає події в подвійному перегляді. Потім є перегляд у реальному часі. У цьому випадку Ditto функціонує як маршрутизатор. Ditto виконує ретрансляцію/маршрутизацію та забезпечує контроль доступу. Програма може захотіти отримати доступ до температури датчика. Але він хоче отримати

живу цінність, а не постійну. Тоді Ditto направляє це повідомлення на пристрій. У цьому випадку Ditto не обробляє команди та не випускає події. Це робиться програмами та пристроями. Ditto просто передає інформацію.

Подвійна перспектива Eclipse Ditto описує постійну структуру даних, яка пов'язана з пристроями та зберігає їх останній зареєстрований стан (рис.2.5) [14].

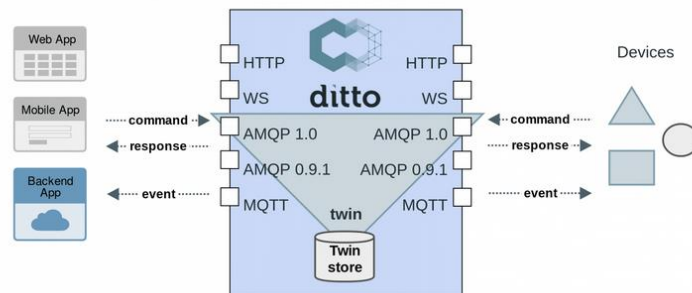


Рисунок 2.5 – Структура даних Eclipse Ditto [14]

2.2.2 Фреймворк Eclipse Hono

Дані передаються від пристроїв через Hono до бізнес-додатків і назад. Eclipse Hono™ знаходиться в хмарі IoT. Метою є спрощене підключення пристроїв для бізнес-додатків (або рішень). Він абстрагує численні особливості різних протоколів підключення на стороні пристрою з точки зору серверної програми. Hono визначає невеликий набір API для єдиного зв'язку з дуже різними пристроями, незалежно від того, чи використовують вони, наприклад, протоколи, орієнтовані на з'єднання, такі як MQTT і AMQP 1.0, або протоколи без збереження стану, такі як HTTP. У результаті Hono звільняє розробників внутрішніх додатків IoT від необхідності детально знати протоколи пристроїв і дозволяє їм додавати або змінювати протоколи підключення без необхідності змінювати свої програми. Для кожного підтримуваного протоколу підключення Hono містить мікросервіс під назвою Protocol Adapter, який

відображає протокол підключення пристрою до API Hono. Ці API пропонуються через центральну систему обміну повідомленнями. Кожен із API обміну повідомленнями має так званий північний API для бізнес-додатків. Південні API використовуються адаптерами протоколів для пересилання повідомлень від пристроїв до програм і навпаки. Повідомлення, що надходять від пристроїв до бізнес-додатків, називаються низхідними повідомленнями, оскільки на практиці це основний напрямок зв'язку. Інший напрямок зв'язку називається висхідним. Це повідомлення, надіслані з бізнес-програми на пристрій через Command & Control API. API, визначені Hono для обміну повідомленнями, це Telemetry, Event і Command & Control. API телеметрії призначений для надсилання довільних даних, напр. від датчиків, вниз по течії. Події також є висхідними повідомленнями, призначеними для менш частих, але більш важливих повідомлень. Залежно від наданої мережі обміну повідомленнями події можуть зберігатися та доставлятися пізніше, якщо програма стикається з обмеженим часом простою. Command & Control API дозволяє надсилати повідомлення, наприклад команди, із північних додатків угору та за бажанням повертати відповідь із пристрою. Посібник «Початок роботи» допомагає ознайомитися з Hono на практичному прикладі. Якщо вас цікавить певний протокол, у відповідному Посібнику користувача вичерпно пояснюється використання. Щоб глибше вивчити концепції, почніть читати про архітектуру та концепцію ідентифікації пристрою Hono. Концептуальна сторінка Multi-tenancy пояснює, як Hono дозволяє ізолювати кілька наборів пристроїв і даних.

У розділі «Підключення пристроїв» показано різні способи підключення пристрою (безпосередньо або через шлюз), а в «Налаштуванні пристроїв» пояснюються деякі варіанти реєстрації пристроїв у Hono. Схеми передачі даних зображені на рисунках 2.6, 2.7, 2.8, 2.9, 2.10, 2.11.

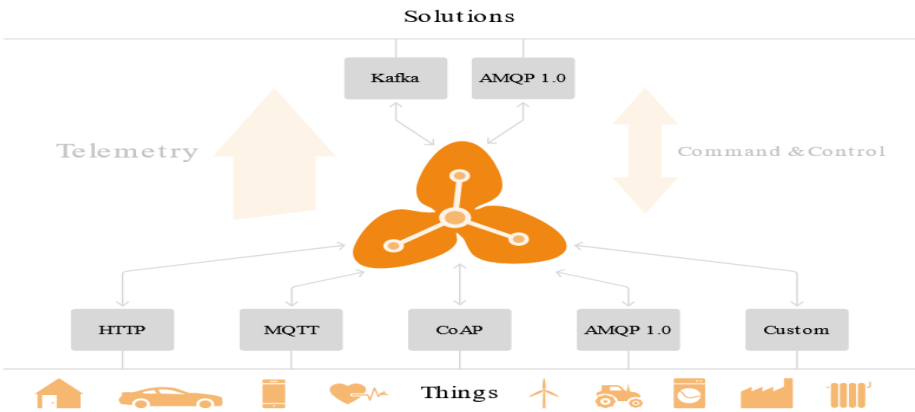


Рисунок 2.6 - Рішення компонентів Ноно [15]

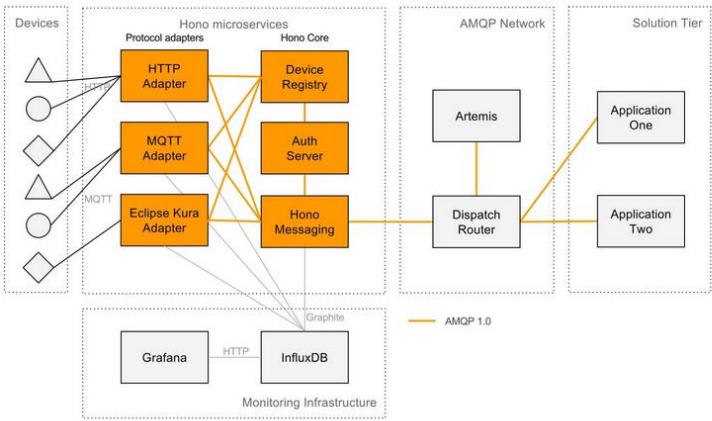


Рисунок 2.7 - Категорії компонентів Ноно [16]

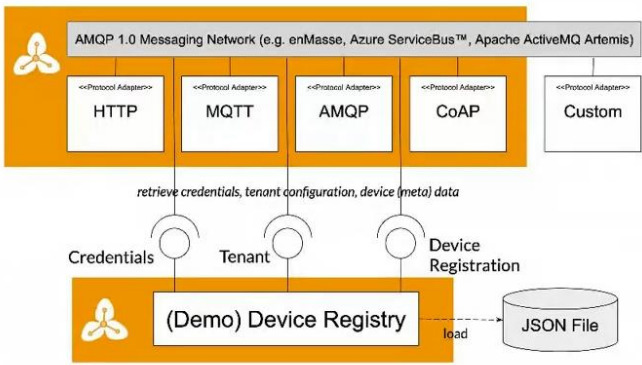


Рисунок 2.8 – South bound Inerface [17]

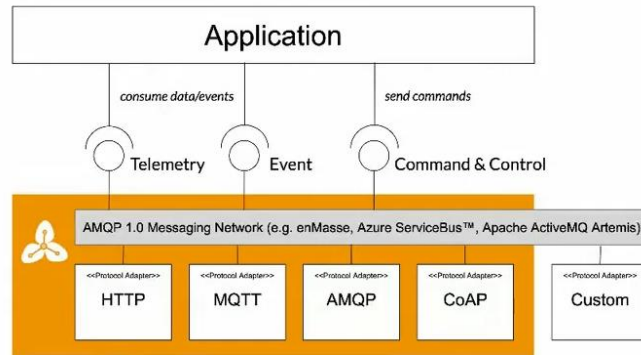


Рисунок 2.9 – North bound Interface [17]

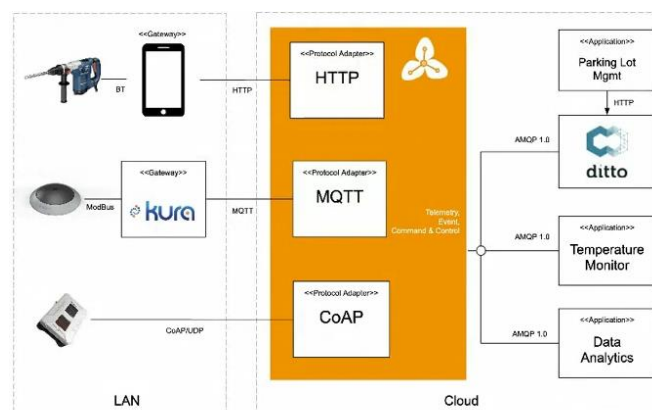


Рисунок 2.10 –Платформа підключень з відкритим кодом [17]

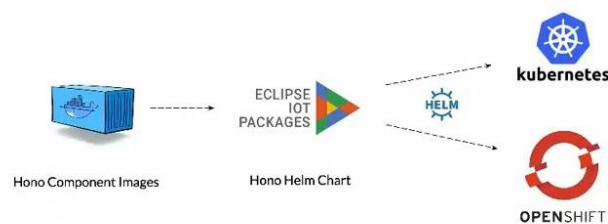


Рисунок 2.11 –Підтримувані платформи оркестрування контейнерів [17]

Адаптери протоколів підключають пристрої до Hono. Кожен охоплює певний протокол і відповідає за автентифікацію пристроїв шляхом запиту до реєстру пристроїв. Hono надає такі адаптери протоколу:

HTTP MQTT Шлюзи на основі Eclipse Kura Користувацькі адаптери протоколу можна додавати за допомогою API Hono Core. Ядро Hono Ядро

Hono складається з: Реєстр пристроїв, впровадження реєстрація та активація пристроїв, що належать орендарю надання облікових даних для пристроїв Сервер авторизації, реалізація створення маркерів безпеки для підтримки автентифікації клієнта Hono Messaging, впровадження API обміну повідомленнями для телеметрії та повідомлень про події перевірка відповідності API пересилання повідомлень до мережі AMQP 1.0 Мережа AMQP 1.0 відокремлює основні мікросервіси Hono від програмних рішень. Він забезпечує масштабовану інфраструктуру обміну повідомленнями, яка підтримує різні шаблони зв'язку з різними рівнями якості обслуговування. Кілька властивостей протоколу обміну повідомленнями AMQP 1.0 роблять його ідеальним кандидатом для цієї мети. По-перше, це один із небагатьох протоколів обміну повідомленнями, який справді є симетричним. Для цього не потрібен брокер як посередник у процесі обміну повідомленнями. Він підтримує шаблони зв'язку «зберігання та пересилання» (за посередництва) та прямі повідомлення. Симетричний характер AMQP 1.0 дозволяє Hono використовувати різну семантику зв'язку та різні компоненти обміну повідомленнями для обробки різних потоків повідомлень. Вхідним шлюзом до системи обміну повідомленнями AMQP Hono є Apache QPid Dispatch Router. Він забезпечує прямий зв'язок між кінцевими точками виробництва та споживання. На відміну від брокера, маршрутизатор ніколи не бере на себе право власності на повідомлення; натомість він просто передає пакети AMQP між різними кінцевими точками. Це означає, що з точки зору обміну повідомленнями маршрутизатор в основному не має стану. Таким чином, він набагато більш вертикально масштабований, ніж брокер повідомлень. Крім того, набагато легше масштабувати його горизонтально в хмарному середовищі – як ми побачимо за мить. Іншою важливою особливістю AMQP 1.0 є його механізм керування потоком, який реалізований безпосередньо в протоколі. Це означає, що споживачі в кожній точці декларуватимуть свою потужність або кількість повідомлень, які їм мають бути доставлені. Ця

інформація передається маршрутизатором, і виробники повідомлень можуть бути обмежені до швидкості споживання.

AMQP (англ. Advanced Message Queuing Protocol) — відкритий стандарт протоколу прикладного рівня для проміжного програмного забезпечення, орієнтованого на обробку повідомлень. Семантика обміну повідомленнями налаштовується під потреби конкретного проекту. «Брокер повідомлень», який здійснює маршрутизацію, зазвичай гарантує доставку, розподіл потоків даних, підписку на потрібні типи повідомлень.

Протокол AMQP (рис 2.12) влаштований досить цікаво: на нижньому рівні визначається формат кодування даних у бінарний вид для передачі по TCP-з'єднанню, вище лежить формат передачі RPC-запитів між сервером та клієнтом. Сама семантика роботи із повідомленнями, створення черг тощо. описується в XML-специфікації, яка по суті задає RPC-інтерфейс сервера та клієнта (приклади таких XML-файлів для версій 0-8 та 0-10). Цей XML є останньою та кінцевою специфікацією протоколу. Більше того, версії протоколу 0-8 та 0-10 відрізняються настільки сильно, що підтримувати їх одночасно навряд чи можливо в одній програмі. Що ще цікавіше, іноді такі спрес-файли для різних брокерів AMQP, формально підтримують одну і ту ж версію протоколу, відрізняються настільки, що не взаємозамінні. Але це скоріше невеликі технічні проблеми.

«Повідомлення» (message) відправляються в «точку обміну». Точка обміну розподіляє повідомлення в одну або кілька «черг». При цьому в точці обміну повідомлення не зберігаються. Точки обміну бувають трьох типів:

- розгалуження (англ. fanout) — повідомлення передається в усі причеплені до неї черги ;
- прямі (direct) — повідомлення передається в чергу з ім'ям, що збігається з ключем маршрутизації (routing key), який вказується при відправці повідомлення ;

– теми (topic) — щось середнє між розгалуженням і прямим типом, повідомлення передається в черзі, для якої збігається маска на ключ маршрутизації.

Повідомлення зберігаються в чергах доти, поки не буде забрано клієнтом. Клієнт завжди забирає повідомлення з однієї або декількох черг.

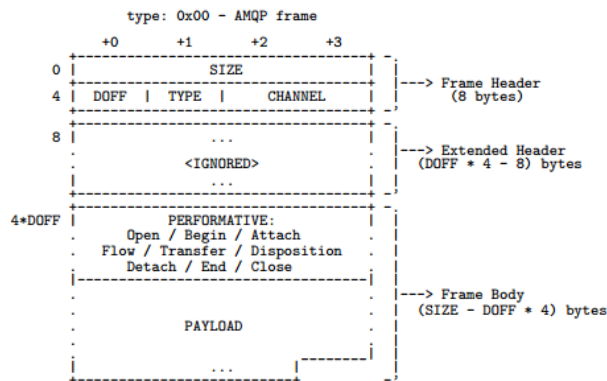


Рисунок 2.12 – Фрейм AMQP [15]

Система зв'язку, побудована на MQTT, складається з сервера-видавця, сервера-брокера та одного або кількох клієнтів (рис 2.13, 2.14).Видавець не вимагає жодних налаштувань за кількістю або розташуванням передплатників, які отримують повідомлення.Крім того, передплатникам не потрібне налаштування на конкретного видавця. У системі може бути кілька брокерів, які розповсюджують повідомлення.

MQTT надає спосіб створення ієрархії каналів зв'язку - свого роду гілка з листям.Щоразу, коли видавець має нові дані для поширення серед клієнтів, повідомлення супроводжується приміткою контролю доставки.Клієнти вищого рівня можуть отримувати кожне повідомлення, тоді як клієнти нижчого рівня можуть отримувати повідомлення, що стосуються лише одного чи двох базових каналів, «відгалуженим» у нижній частині ієрархії. Це полегшує обмін інформацією розміром від двох байт до 256 мегабайт.

Приклад того, як можна налаштувати клієнт для підключення через брокер MQTT:

```
var options = {
  keepalive: 60,
  username: 'FIRST_HALF_OF_API_KEY',
  password: 'SECOND_HALF_OF_API_KEY',
  port: 8883
};
var client = mqtt.connect('mqtts:mqtt.ably.io', options);
```

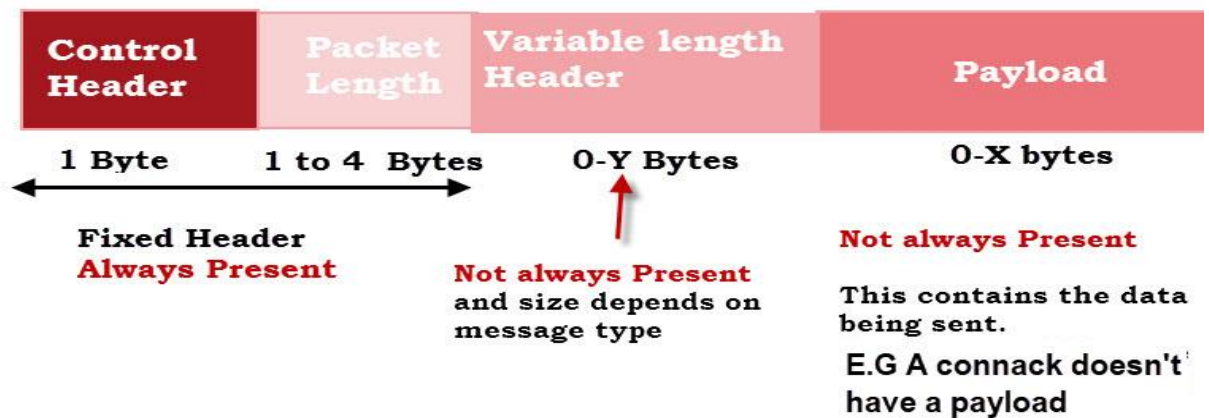


Рисунок 2.13 – фрейм MQTT [16]



Рисунок 2.14 –MQTT клієнт брокер проткол [16]

У протоколі CoAP визначено всього чотири види повідомлень: Confirmable, Non-confirmable, Acknowledgement, Reset. Усі повідомлення кодуються в бінарному вигляді. Формат повідомлення зображено на рис. 2.14. Повідомлення починається із заголовка, з фіксованим розміром в 4 байта, за

ним слід маркер, опції та корисна завантаження, яка займає всю останню частину дейтаграми.

На рис .2.15 область "А" показує приклад взаємодії користувача пристроєм (HTTP-клієнта) із CoAP-датчиком через мережу Інтернет. HTTP-клієнт генерує HTTP-запити та відправляє їх на сервер, який за відсутності необхідної інформації на отриманий запит звертається до CoAP-проксі. Тут CoAP-проксі – це пристрій, що з'єднує мережу Інтернет на основі протоколу HTTP та мережу з обмеженими ресурсами, що підтримує протокол CoAP. Проксі перетворює повідомлення однієї мережі (протокол HTTP) на повідомлення, зрозумілі іншій (протокол CoAP) (рис 2.16).

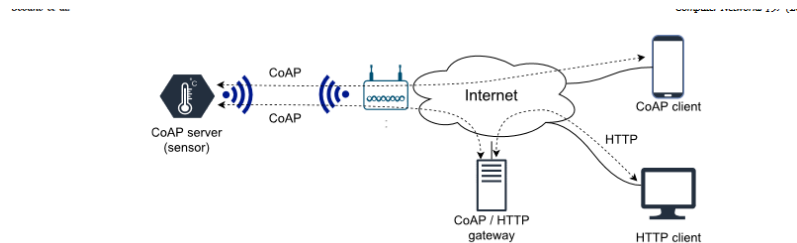


Рисунок 2.15 – Взаємодія протоколів CoAP та HTTP [17]

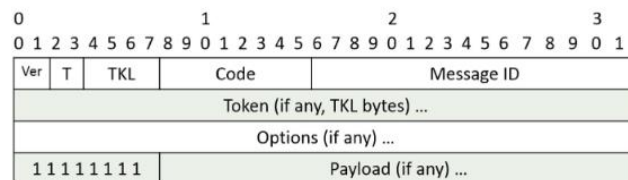


Рисунок 2.16 – Фрейм протокола CoAP [17]

Поля у повідомленні визначаються в такий спосіб. Версія (Version) (Ver) – двобитне ціле число, що вказує на номер версії CoAP. На даний момент існує лише одна версія протоколу. Тип (Type) (T) – двобитне ціле число. Вказує на тип повідомлення Confirmable (0), Non-confirmable (1), Acknowledgement (2) або Reset (3). Довжина маркера (Token Length) (TKL) – чотирибитне ціле число. Вказує довжину поля маркера (Token) змінної

довжини (0-8 байт). Довжини від 9 до 15 зарезервовані. Код (Code) – восьмибітне ціле число, поділяється на трибітний та п'ятибітний клас

Поля у повідомленні визначаються в такий спосіб. Версія (Version) (Ver) – двобитне ціле число, що вказує на номер версії CoAP. На даний момент існує лише одна версія протоколу. Тип (Type) (T) – двобитне ціле число. Вказує на тип повідомлення Confirmable (0), Non-confirmable (1), Acknowledgement (2) або Reset (3). Довжина маркера (Token Length) (TKL) – чотирибітне ціле число. Вказує довжину поля маркера (Token) змінної довжини (0-8 байт). Довжини від 9 до 15 зарезервовані. Код (Code) – восьмибітне ціле число, поділяється на трибітний та п'ятибітний клас (рис. 3). Поле "код" записується як "c.dd", де "c" – це цифра від 0 до 7 для трибітного підполя і "dd" – дві цифри в діапазоні від 00 до 31 для п'ятибітного підполя. Код 0.00 вказує на пусте повідомлення. Код 0.01-0.31 вказує на запит. Код 1.00–1.31 зарезервовано. Код 2.00–2.31 вказує на успішну відповідь. Код 3.00–3.31 зарезервовано. Код 4.00-4.31 вказує на помилку клієнта. Код 5.00-5.31 вказує на помилку сервера. Код 6.00–7.31 зарезервовано. Значення "dd" – це фіксовані табличні величини, які вказують у разі, якщо повідомлення – запит на метод запиту (наприклад, 0.01 – це запит GET, 0.02 – це запит POST); у випадку, якщо повідомлення – відповідь, вказують на код відповіді (наприклад, 2.05 – це відповідь, що містить запитувану інформацію (Content), 4.04 – це помилка, що виникає, коли запитуваний пристрій відсутній або тимчасово недоступний (Not Found)). Ідентифікатор повідомлення (Message ID) – шістнадцятибітне ціле число, що є унікальним ідентифікатором повідомлення. Message ID ідентифікує повідомлення, щоб визначити, на який запит надійшла інформація чи помилка. CoAP-пристрій, надсилаючи повідомлення Confirmable або Non-confirmable, випадково генерує значення ідентифікатора для цього повідомлення. У відповідях Acknowledgement- або Reset-ідентифікатори будуть ті ж, що і в повідомленні, на яке вони відповідають. За заголовком слідує значення маркера (Token), яке може становити від 0 до 8 байт, як зазначено в полі TKL. Значення маркера

випадково генерується пристроєм CoAP і використовується в межах однієї сесії для встановлення відповідності запиту з відповіддю, тобто відбувається угрупування повідомлень по ланцюжку "запит-відповідь". Нульове значення маркера використовується, коли інші маркери не використовуються у пристрої, на який надсилається запит, або якщо запити робляться послідовно і в малій кількості. Далі йдуть опції (Options), у яких описуються різні параметри повідомлень. Наприклад, є Max-Age Option, яка встановлює максимальний час зберігання інформації в тимчасовій пам'яті, цей параметр за промовчанням дорівнює 60 сек. Інша опція, Content-Format Option, задається у вигляді числа, яке встановлює формат подання корисного навантаження: значення 0 вказує на те, що корисне навантаження буде текстовим, а значення 41 вказує на те, що корисне навантаження буде у форматі xml. Для адресації URI в протоколі CoAP передбачені опції Uri-Host (визначає адресу інтернет-хоста запитуваного ресурсу), Uri-Port (визначає номер порту транспортного рівня запитуваного ресурсу), Uri-Path (визначає частину шляху до ресурсу) і Uri-Query (визначає параметр, який вимагає ресурс). Наприклад: Uri-Host = "example.net" Uri-Port = 5683 Uri-Path = ".well-known" Uri-Path = "core" Uri-Query = "login?" coap://example.net:5683/.well-known/core/login? У протоколу CoAP визначено лише чотири види повідомлень: Confirmable, Non-confirmable, Acknowledgement, Reset. Усі повідомлення кодуються у бінарному вигляді. Список інших опцій протоколу. Поле, що завершує формат повідомлення, - це корисне навантаження (Payload), в якій міститься інформація, що запитується. У COAP використовується чотири типи повідомлень: Confirmable(CON) – повідомлення, що містить запит або відповідь, що вимагає підтвердження та вважається надійним. Кожне повідомлення Confirmable викликає одне повідомлення у відповідь підтвердження або скидання; Non-confirmable(NON) – повідомлення, що містить запит або відповідь, що не вимагає підтвердження та надійної передачі, оскільки передається регулярно (наприклад, показання від датчика); Acknowledgement(ACK) – повідомлення, що підтверджує, що надійшло

повідомлення Confirmable. При цьому саме повідомлення Acknowledgement не означає успіху чи невдачі будь-якого із запитів, що міститься в повідомленні Confirmable; Reset(R) – повідомлення скидання, що вказує на те, що конкретне повідомлення (Confirmable або Non-confirmable) було отримано, але деяка частина тексту відсутня і неможливо правильно обробити його. Така ситуація зазвичай виникає, коли приймаючий вузол перевантажений. Виклик цього повідомлення (наприклад, шляхом надсилання порожнього повідомлення Confirmable) також корисний як перевірка доступності вузла ("CoAP ping").

Архітектура CoAP зображена на рисунку 2.17.

Специфікація протоколу розрахована досить скромні мережеві можливості, і саме у його рамках не реалізуються багаторівневі транспорти передачі повідомлень. Натомість використовується протокол UDP поверх IP.

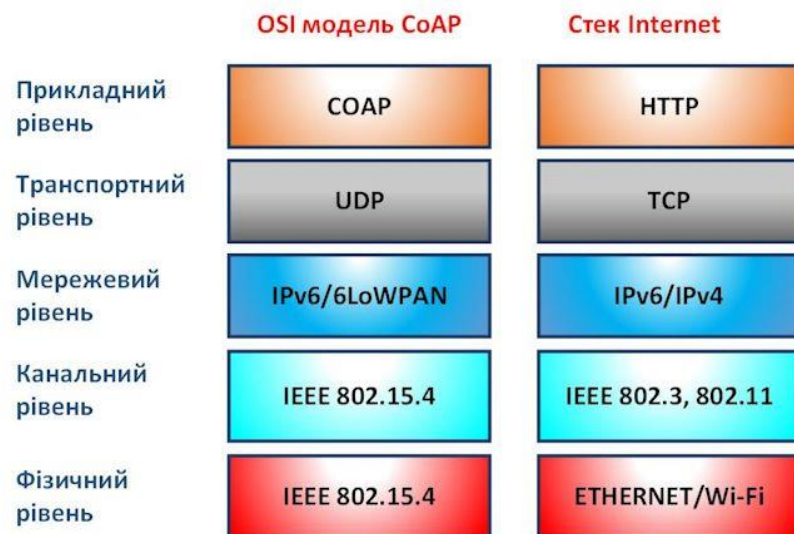


Рисунок 2.17 – Архітектура CoAP [18]

Протокол CoAP використовує UDP (UserDatagramProtocol) як транспортний протокол за умовчанням, що дозволяє зменшити розмір службових даних і збільшити ефективність роботи. В окремих випадках можуть також використовуватися TCP або SCTP. Оскільки CoAP – це протокол прикладного рівня, то у випадках, коли необхідно забезпечити

безпечне з'єднання, використовується шифрування повідомлень за протоколом DTLS (Datagram Transport Layer Security). DTLS працює поверх протоколу UDP та служить для захисту даних на транспортному рівні (працює між рівнем додатків та транспортним рівнем) [18].

Рівень CoAP складається з двох різних рівнів: повідомлень і запитів/відповідей (рис 2.18).



Рисунок 2.18 – Рівні CoAP [18]

Рівень повідомлень працює з UDP і асинхронними повідомленнями. Рівень запит/відповідь керує взаємодією запит/відповідь на основі повідомлень запиту/відповіді.

CoAP підтримує чотири різні типи повідомлень:

CON — таке повідомлення (запит/відповідь) вимагає повідомлення у відповідь про підтвердження;

ACK — повідомлення у відповідь на повідомлення типу CON;

NON — повідомлення (запит/відповідь), що не вимагає відповідного підтвердження;

RST — скидання повідомлення (Reset). Таке повідомлення надсилається у відповідь, якщо отримана інформація частково відсутня або її неможливо прийняти. У тому числі подібні повідомлення можна використовувати для пінгування вузла, відправляючи на вузол порожнє повідомлення типу CON.

Протокол CoAP дозволяє об'єднання пристроїв у групи для розсилки групового повідомлення.

Обмін повідомленнями між сервером та клієнтом можливий в асинхронному режимі, що суттєво прискорює роботу системи. Інформація про наявні у доступі ресурси доступна клієнту у вигляді списку, включаючи метадані. Це дозволяє клієнту зрозуміти, які ресурси є в наявності та їх формат. [18]

Такі програми, як gzip, уже давно використовуються для стиснення даних, надісланих у повідомленнях HTTP, особливо для зменшення розміру файлів CSS і JavaScript. Однак компонент заголовка повідомлення завжди надсилається як звичайний текст. Незважаючи на те, що кожен заголовок є досить малим, навантаження цих нестиснених даних стає все важчим і важчим для з'єднання, оскільки надходить більше запитів, особливо штрафуючи складні веб-програми з великим навантаженням на API, які вимагають багато різних ресурсів і, отже, багато різних запитів на ресурси. Крім того, використання файлів cookie іноді може зробити заголовки набагато більшими, що збільшує потребу в певному стисненні. Щоб усунути це вузьке місце, HTTP/2 використовує стиснення HPACK для зменшення розміру заголовків, тему, яка обговорюється далі в наступному розділі. HTTP/2 Однією з тем, яка знову і знову виникає в HTTP/2, є його здатність використовувати бінарний кадровий рівень для більшого контролю над дрібнішими деталями. Те ж саме стосується стиснення заголовка (рис. 2.19). HTTP/2 може відокремлювати заголовки від своїх даних, створюючи кадр заголовка та кадр даних. Спеціальна для HTTP/2 програма стиснення HPACK може стиснути цей кадр заголовка. Цей алгоритм може кодувати метадані заголовка за допомогою кодування Хаффмана, таким чином значно зменшуючи його розмір. Крім того, HPACK може відстежувати раніше передані поля метаданих і додатково стискати їх відповідно до динамічно зміненого індексу, який спільно використовують клієнт і сервер.

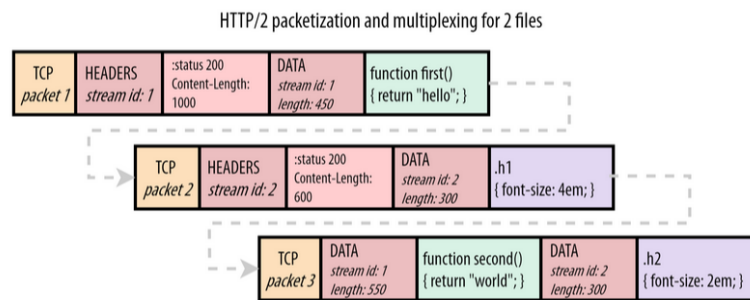


Рисунок 2.19 – Фрейм .HTTP/2 [19]

2.3 Модельно орієнтоване проектування цифрових двійників з використанням Matlab

Стандартом для розробки програмного забезпечення в автомобільній сфері є V-модель. У Matlab доступні різні інструменти для підтримки V-моделі програмного забезпечення. На рисунку 2.20 нижче наведено декілька з них.

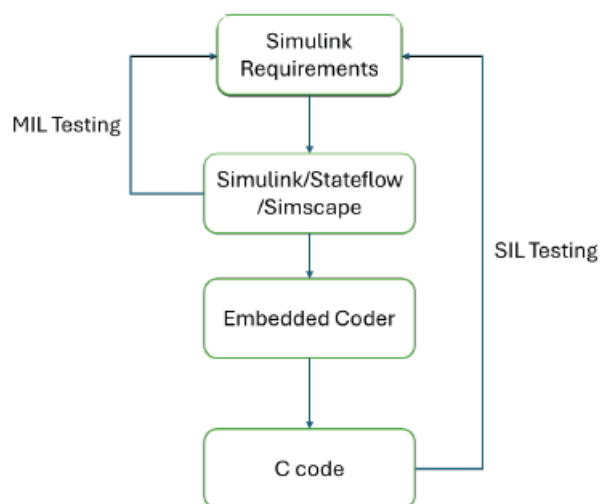


Рисунок 2.20 – MathWorks ланцюг розробника для розроблення та тестування

Модуль «Simulink Requirements» можна використовувати для виявлення та відстеження вимог. Для цілей моделювання Matlab має такі інструменти, як «Simulink», «Simscape» і «Stateflow». Розроблені моделі потім можуть бути

перетворено у вбудований C за допомогою інструменту кодування MATLAB «Embedded Coder». Цей процес, як показано (рис.2.5), є циклічним. Доопрацювання можна зробити шляхом перевірки моделей (Model-in-Loop), а потім згенерованого коду C (Software-in-Loop) із вимогами. Такий підхід сприяє розвитку і тестування відповідно до підходу V-моделі.

3 АРХІТЕКТУРИ ЦИФРОВОГО ДІВЙНИКА НА ОСНОВІ МОДЕЛЬНО-ОРІЄНТОВАНОГО ПРОЄКТУВАННЯ

3.1 Стек технологій

Розглянемо основні компоненти розгортання цифрового двійника на основі вільного програмного забезпечення.

Крок1. Розгортається хмарний кластер на основі Kubernetes, для розгортання архітектури з використанням Matlab (рис. 3.1)

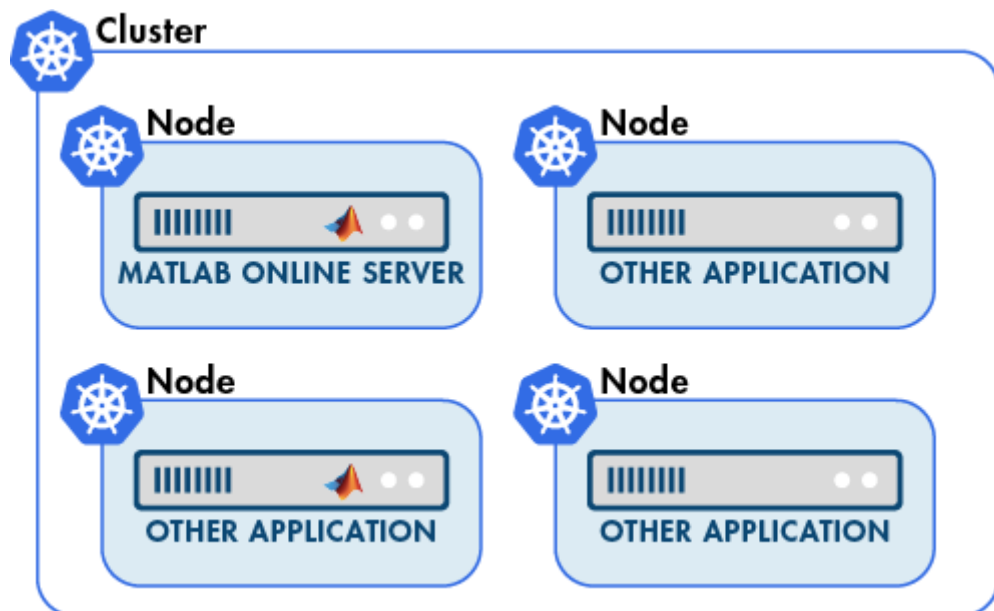


Рисунок 3.1 – Розгортання кластеру [20]

Для налаштування пакетів рекомендується використовувати Helm.

Клієнт Helm — це клієнт командного рядка для кінцевих користувачів. Клієнт несе відповідальність за: розробки локальної діаграми управління репозиторіями, управління випусками, взаємодію з бібліотекою Helm, надсилання діаграм для встановлення запиту на оновлення або видалення існуючих версій.

Крок 2. Налаштовується MATLAB Online Server

Потрібно скопіювати необхідні діаграми Helm із реєстру контейнерів MathWorks у кореневу папку MATLAB Online Server у папці під назвою charts.

Завантажується необхідне зображення Docker із контейнера MathWorks.

Встановлюється Matlab Online Server.

Виконуються кроки налаштування, необхідні для налаштування сервера. Налаштовується автентифікація та авторизація користувача, вказавши дані постачальника ідентифікаційної інформації. Налаштуйте постійне сховище до яких каталогів, дисків і файлів користувачі мають доступ. Налаштуйте інсталяції MATLAB або інтегруйте інші продукти MathWorks на сервер.

Крок 3. Налаштовується NGINX Ingress Controller (рис.3.2). Вхідний контролер NGINX® обробляє вхідні запити до кластера Kubernetes® і діє для балансування навантаження на MATLAB® Online Server™.

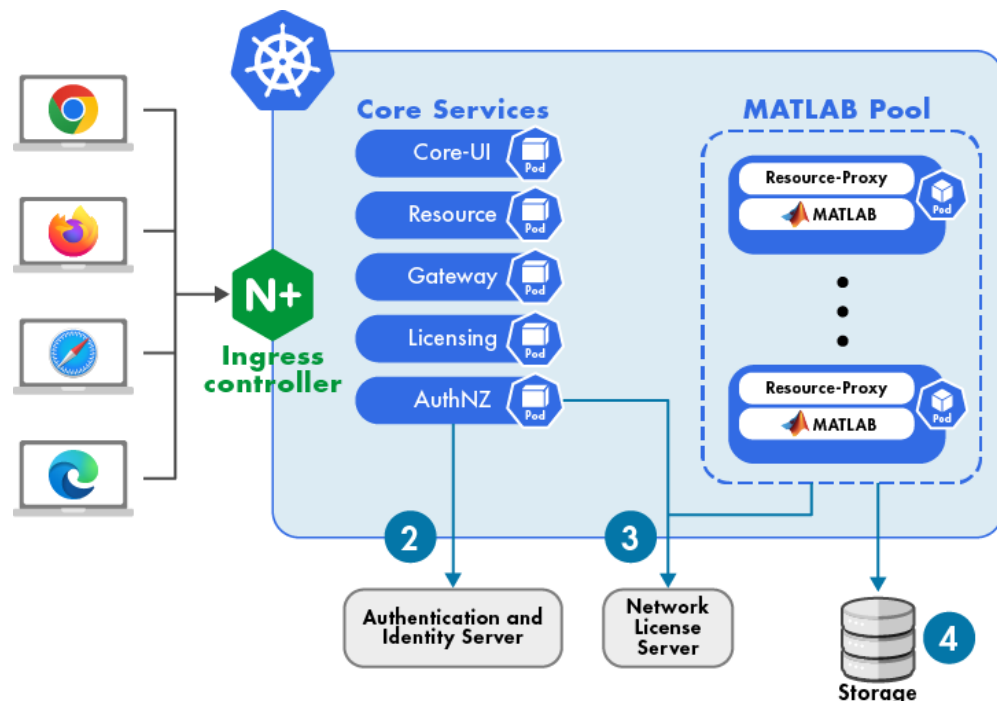


Рисунок 3.2 – Використання NGINX Ingress Controller [21]

Крок 4. Налаштовується хмарна база даних, куди збираються дані з пристроїв. При використанні вільного програмного забезпечення.

Крок 5. Використовується остання версія офіційного образу Eclipse Ditto.

Крок 6. Налаштування з'єднання з IoT-пристроями (рис.2.4)

Крок 7. Налаштовується програмне забезпечення для моніторингу і тестування.

При використанні препроітарного програмного забезпечення, змінюються крок 5 вимагає встановлення образу відповідного програмного забезпечення. Наприклад, для автомобільної промисловості DSpace Veos.

3.2 Приклад архітектури цифрового двійника

Розглянемо приклад модельно-орієнтованого проектування цифрового двійника електромобіля.

Програма Virtual Vehicle Composer — це потужний інструмент для створення та оцінки цифрових моделей транспортних засобів для ретельної оцінки продуктивності на рівні системи. Він дозволяє виконувати широкий діапазон оцінок, від розміру компонентів і економії палива до продуктивності батареї та динаміки автомобіля, за рахунок набору функцій. Це адаптивне налаштування відповідає різноманітним вимогам тестування, наприклад як моделювання циклу водіння, оцінка маневру для керування транспортним засобом та інтеграційні тести як програмного, так і апаратного забезпечення в циклі. Користувачі можуть швидко будувати архітектури, вивчати варіанти архітектур, і отримати уявлення про їхні конструкції, ввівши дані параметрів автомобіля та використовуючи блоки і підсистеми довідкових додатків усередині Powertrain Blockset, Vehicle Dynamics Blockset та Додатки до Simscape. [22] Програма надає повну платформу для налаштування кількох архітектур транспортних засобів, включаючи звичайні, електричні та гібридно-електричні конфігурації для користувачів, які мають трансмісію Набір блоків. Він дозволяє симулювати за певних параметрів, таких як цикли

керування FTP, увімкнення обширний аналіз і визначення розмірів компонентів, які мають вирішальне значення для покращення продуктивності автомобіля. Програмне забезпечення також містить можливість відображення цих складних віртуальних автомобілів у середовищі симуляції Unreal Engine, що додає елемент занурення до процесу оцінювання.

Для налаштування віртуального транспортного засобу, потрібно виористовувти останню версію MATLAB (вище R2022a).

Після запуску програми VVC запускається нове налаштування конфігурації, як показано на рисунку 3.3

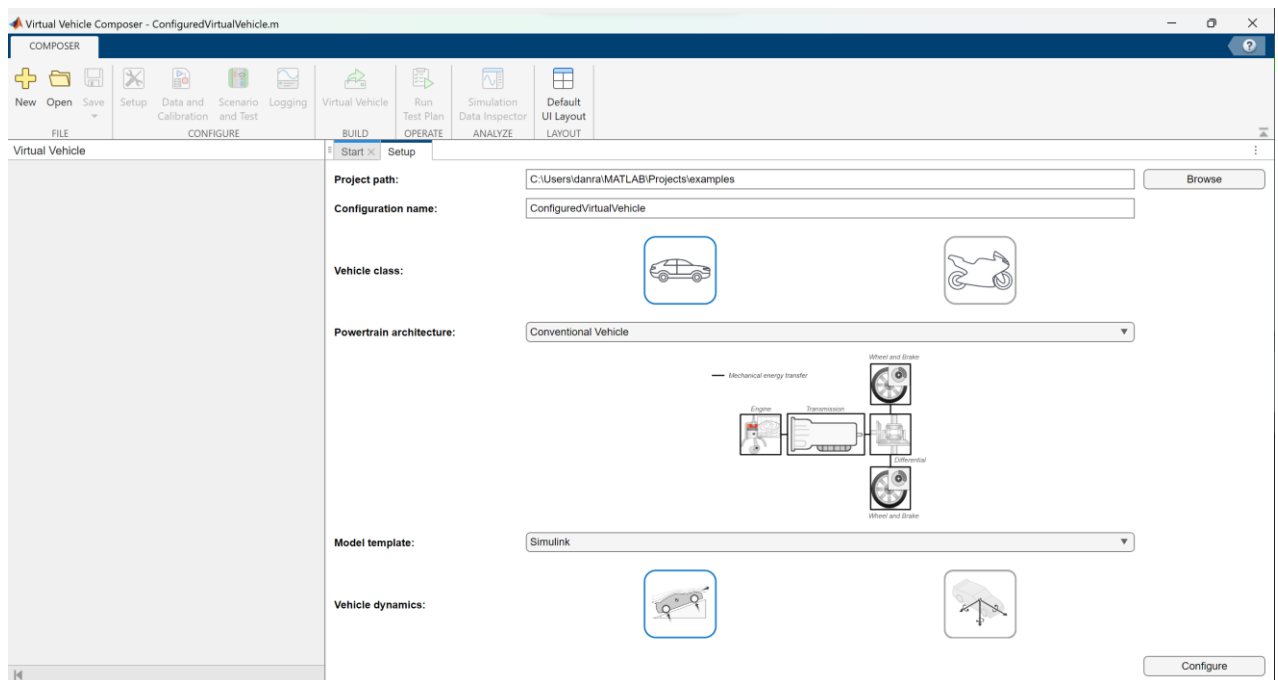


Рисунок 3.3 – Визначення конфігурації

Наступний крок калібрування даних (рис. 3.4)

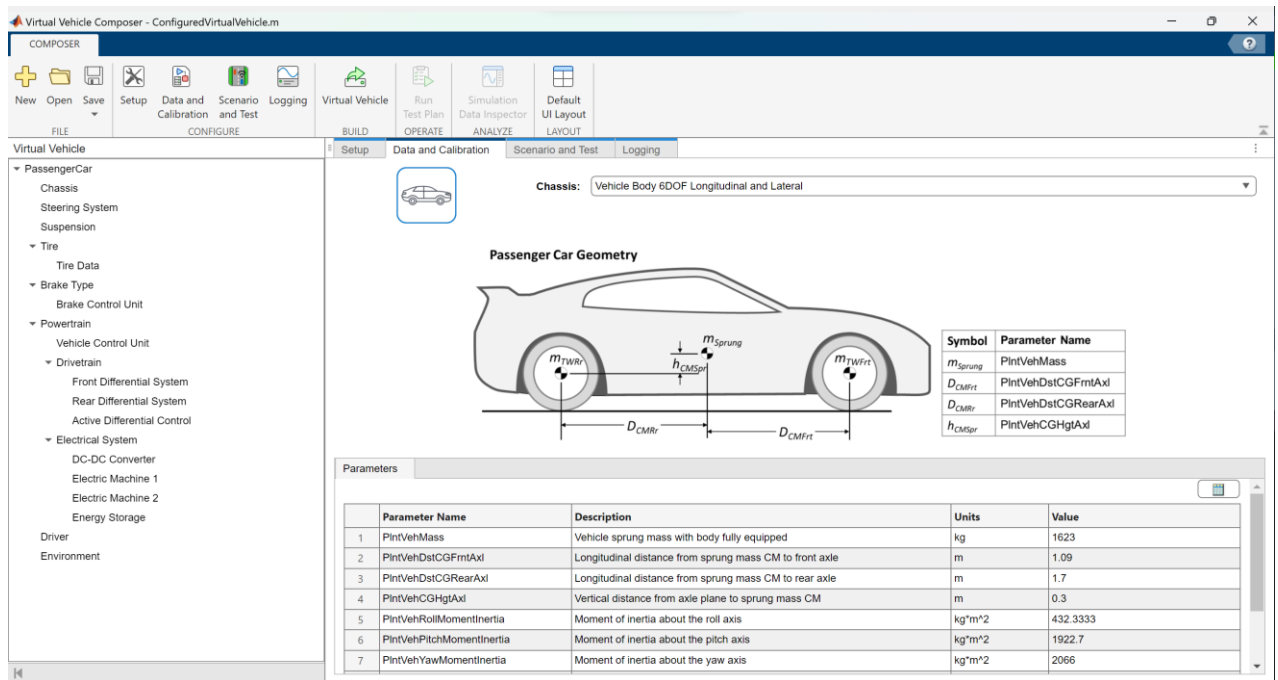


Рисунок 3.4 – Калібрування даних

Тут потрібно вказати інформацію для різних компонентів автомобіля, таких як шасі, рульове керування, підвіска, шини, гальма, силовий агрегат (звичайний, EV-1ЕМ, 2ЕМ тощо), а також фактори навколишнього середовища для кожного елемента.

Виконання покрокових процедур на вкладці «Творець» дає користувачам змогу всебічно налаштовувати, керувати та оцінювати віртуальні транспортні засоби, сприяючи глибокому розумінню їх показники ефективності та характеристики (рис. 3.5).

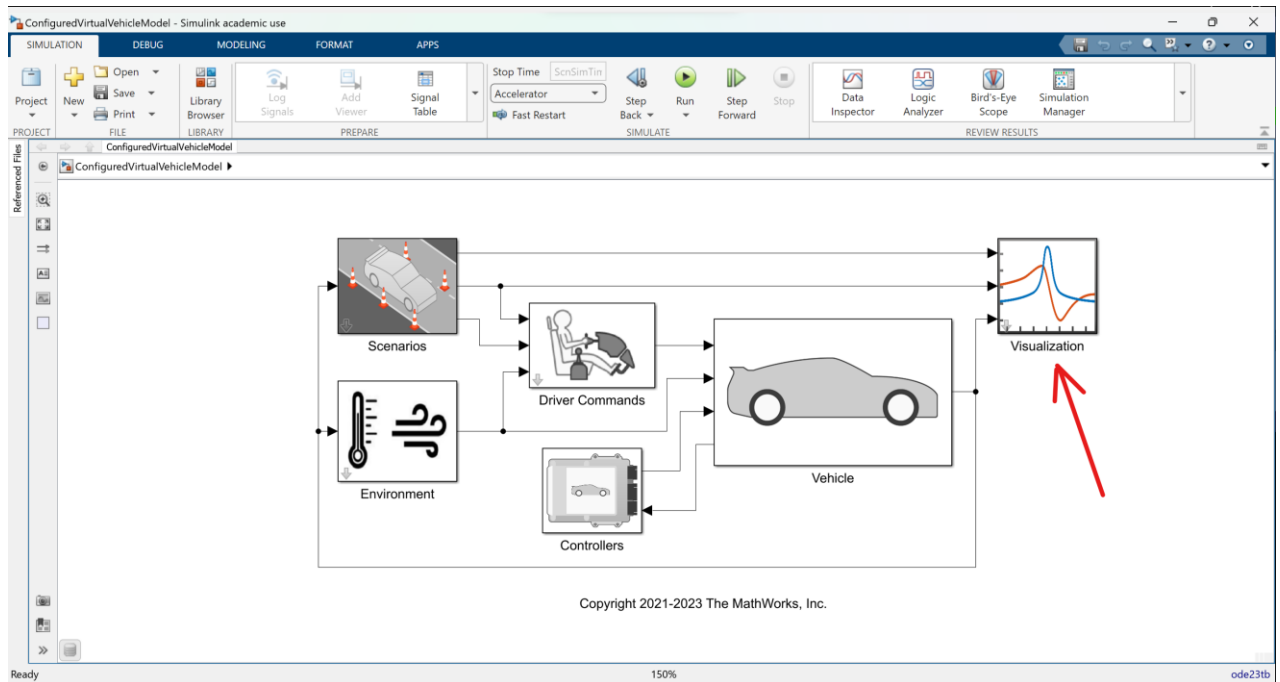


Рисунок 3.5 – Віртуальний автомобіль

Метод розробки програмного забезпечення, який ми тут використовували, — «розробка через моделювання». Така розробка обладнання за допомогою моделювання полегшує інженерам і дослідникам аналіз, покращує якість і дозволяє створювати алгоритми ще до створення коду. Симуляція була досягається за допомогою ланцюжка інструментів MATLAB. Оскільки автомобільний транспортний засіб є комбінацією великого діапазону підсистем і компонентів, проектувати все з нуля є громіздкою справою. Крім того, архітектура автомобіля складається з підсистем, спроектованих і розроблених інженерами з різних галузей, таких як механіка, електрика, електроніка та програмне забезпечення. Саму тут, нам на допомогу приходить додаток Virtual Vehicle Composer (VVC).

Virtual Vehicle Composer (VVC) — це конструктор автомобільних транспортних засобів. Додаток VVC надав нам гарну відправну точкою є створення базової конструкції транспортного засобу на основі наших вимог до дизайну Simulink. Ми змогли налаштувати середовище моделювання відповідно до проектних вимог хвилин і подальше зіставлення його з

фактичними фізичними моделями. Основна мета моделювання системи, це можливість експериментувати з нею, змінюючи її стани або компоненти.

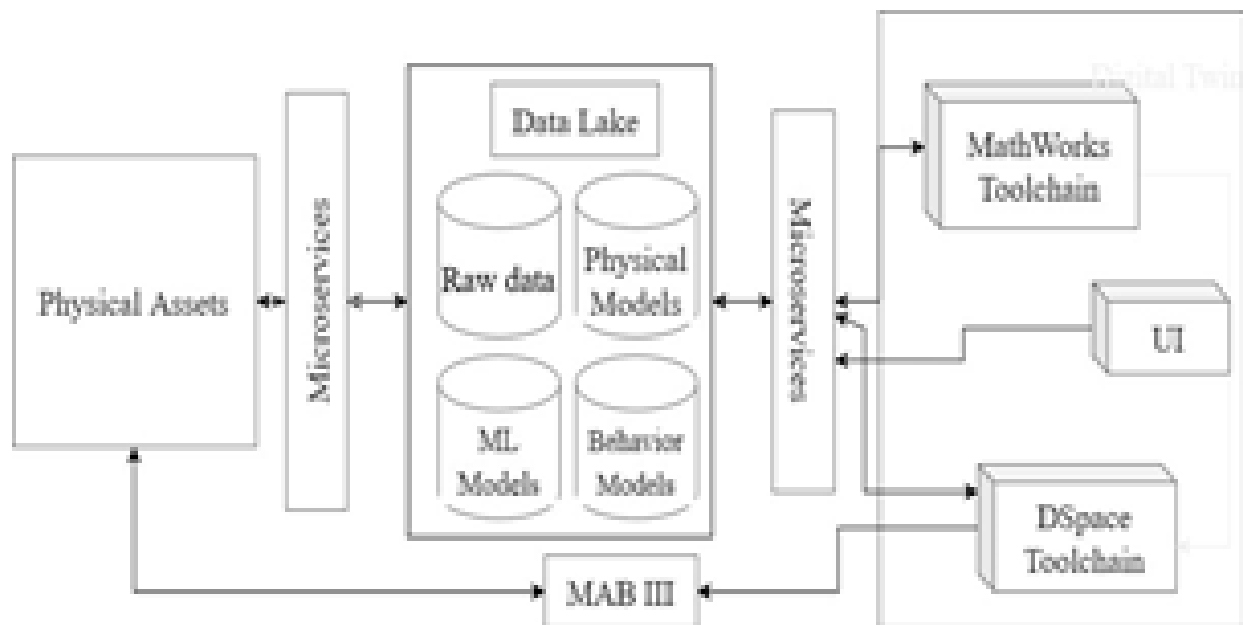
Після розробки поведінкової моделі проводяться як статичні, так і динамічні тести на рівні моделі та вихідного коду. Статичні тести виконуватимуться на рівнях компонентів та інтеграції, щоб виявити помилки в моделі та пов'язаному вихідному коді.

Після верифікації та валідації моделі, щоб переконатися в її правильності та відповідності вказівкам щодо моделювання, наступним кроком є генерація коду, яка виконується за допомогою вбудованого кодера, щоб уникнути порушення будь-яких стандартних вказівок щодо кодування. Є можливість генерувати індивідуальний код для конкретної цілі, включаючи підтримку стандартів Autosar Classic і Autosar Adaptive, з однієї моделі Simulink. Після створення робочого коду його слід проаналізувати, щоб виявити дефекти, переконатися у відсутності помилок виконання та перевірити відповідність стандартам кодування MISRA C.

Залежно від обсягу тестування, динамічне тестування виконується шляхом запуску реалізованої моделі та програмних блоків на симуляції на базі ПК або реального цільового ECU.

Для цифрового двійника тут можна використовувати чи доступні контролери та вільне програмне забезпечення на прикладі Eclipse Dito. Основним протоколом взаємодії було обрано CAN.

В проєкті використовувалось пропрієтарне програмне забезпечення DSpace (рис.3.6).



ML - Machine Learning
MAB III - MicroAutoBox III by DSpace
UI - User Interface

Рисунок 3.6 – Архітектура запропонованого цифрового двійника

ВИСНОВКИ

Робота присвячена дослідженню програмних архітектур програмних засобів проєктування та програмної реалізації цифрових двійників. Досліджено поняття цифрового двійника, цифрової тіні та цифрової моделі. Проаналізовані засоби створення 5-ти та 6-ти рівневих архітектур цифрових двійників. Розглянуто модельно-орієнтований підхід к проєктуванню цифрових двійників, та запропоновано підхід для розгортання цифрового двійника з інтеграцію моделей отриманих з використанням програмного забезпечення Matlab та Simulink.

Для проєктування взаємодії з фізичними та віртуальними моделями складних систем можливо використання як вільного так і пропрієтарного програмного забезпечення. В роботі проаналізовані Eclipse Ditto та Eclipse Nano, що дозволяють розробляти цифрові двійники широкого призначення.

Запропоновано фреймворк для розгортання цифрових двійників на хмарі на осанові докерезованих контейнерів. Приведено приклад конструювання вимог до архітектури цифрового двійника для електромобіля.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wenzel, T. & Burnham, Keith & Blundell, Mike & Williams, R.. (2007). Kalman filter as a virtual sensor: Applied to automotive stability systems. Transactions of the Institute of Measurement and Control. 29. 95-115. 10.1177/0142331207072990.
2. W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," IFAC-PapersOnLine, vol. 51, no. 11, pp. 1016-1022, 2018. doi: 10.1016/j.ifacol.2018.08.474.
3. Glaessgen, Edward & Stargel, David. (2012). The digital twin paradigm for future NASA and U.S. air force vehicles. 10.2514/6.2012-1818.
4. Yassine El-Ouardi, Oussama Hasidi, Jakob Khamis, El Hassan Abdelwahed, Aimad Qazdar, Stephan Sauter, Jens Timmermann, and Ahmed Bendaouia. 2023. Digital Twin Technology for Enhanced Health Monitoring, Maintenance, and Energy Efficiency of Electric Vehicles: E-mule Project. In Proceedings of the 6th International Conference on Networking, Intelligent Systems & Security (NISS '23). Association for Computing Machinery, New York, NY, USA, Article 64, 1–7. <https://doi.org/10.1145/3607720.3607788>
5. ResearchGate LogoE [Electronic resource]: – Access mode: https://www.researchgate.net/figure/Model-Based-Design-provides-a-multi-domain-tool-environment-and-integrates-all-phases-of_fig6_259443931
6. Вікіпедія [Electronic resource]: – <https://uk.wikipedia.org/wiki/LabVIEW>
7. MathWorks [Electronic resource]: – <https://uk.mathworks.com/solutions/system-design-simulation.html>
8. G. Tabunshchyk, P. Arras and C. Wolff, "Digital Twins in Engineering Education, Preparing Students for Industrial Digital Transformation," 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced

Computing Systems: Technology and Applications (IDAACS), Dortmund, Germany, 2023, pp. 271-274, doi: 10.1109/IDAACS58523.2023.10348648

9. Generic Digital Twin Architecture for Industrial Energy Systems [Electronic resource]: – Access mode: <https://www.mdpi.com/2076-3417/10/24/8903/htm>

10. L. Kumar et al., "Electric Vehicle (EV) Preventive Diagnostic System: Solution for Thermal Management of Battery packs using AIOT," 2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2023, pp. 0041-0046, doi: 10.1109/CCWC57344.2023.10099185.

11. A Six-Layer Digital Twin Architecture for a Manufacturing Cell [Electronic resource]: – Access mode: https://www.researchgate.net/publication/329580256_A_SixLayer_Digital_Twin_Architecture_for_a_Manufacturing_Cell_Proceedings_of_SOHOMA_2018

12. Azure Digital Twins [Electronic resource]: – Access mode: <https://azure.microsoft.com/en-gb/products/digital-twins/#overview>

13. Laura Cattaneo, Marco Macchi, A Digital Twin Proof of Concept to Support Machine Prognostics with Low Availability of Run-To-Failure Data, IFAC-PapersOnLine, Volume 52, Issue 10, 2019, Pages 37-42, ISSN 2405-8963, <https://doi.org/10.1016/j.ifacol.2019.10.016>.

14. Bosch Digital Blog [Electronic resource]: – <https://blog.bosch-digital.com/how-digital-twins-boost-development-in-the-iot/>

15. Wordpress [Electronic resource]: – <https://paolopatierno.wordpress.com/2015/07/23/amqp-on-the-wire-messages-content-framing/>

16. Alfacomp [Electronic resource]: – <https://alfacomp.net/2022/03/24/mqtt-conheca-o-protocolo-padrao-do-iot/>

17. University Carlos III of Madrid, Av. de la Universidad, 30, Leganés (Madrid), Spain https://iotmadlab.es/wp-content/uploads/2023/10/Performance_CN_2021.pdf

18. Інформаційний ресурс про електроніку та програмування [Electronic resource]: – <https://itmaster.biz.ua/directory/standarts/coap.html>
19. RUSLAN NOVIKOV [Electronic resource]: – <https://novikov.ua/head-of-line-blocking-%D0%B2-quic-%D0%B8-http-3/>
20. MathWorks [Electronic resource]: – <https://de.mathworks.com/help/matlab-online-server/admin/host-matlab-online.html>
21. MathWorks [Electronic resource]: – <https://de.mathworks.com/help/matlab-online-server/admin/host-matlab-online.html>
22. Design of an Open Modular Experimental Vehicle (OMAxVehicle)
Link: <https://www.fh-dortmund.de/projekte/omaxvehicle.php> [Last accessed - 06 September 2024]

ДОДАТОК А
Слайди презентації

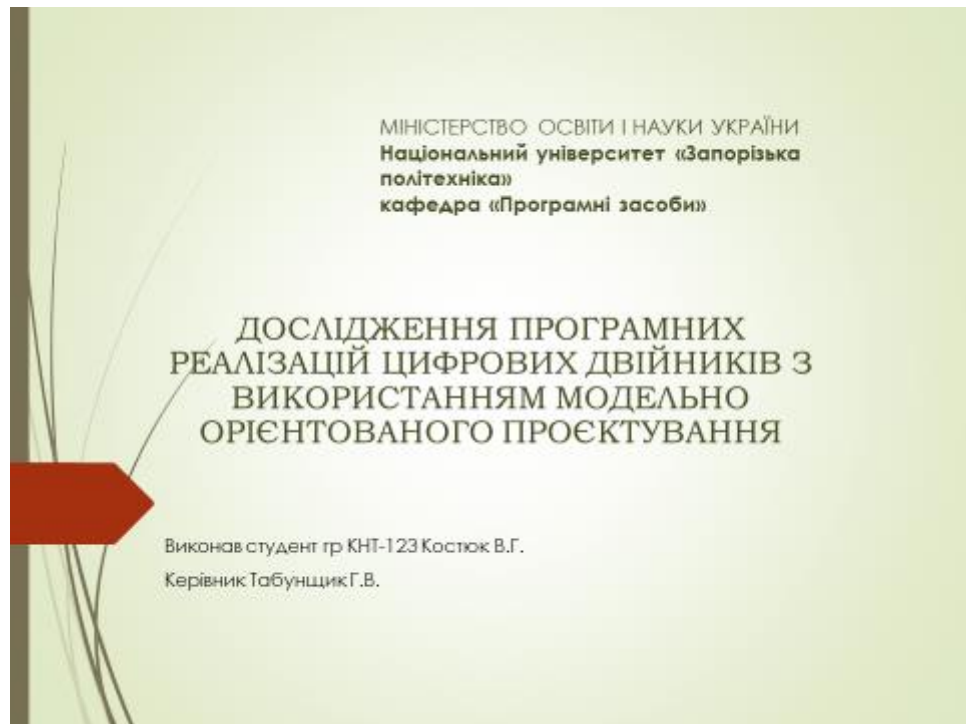


Рисунок А.1 – Слайд 1

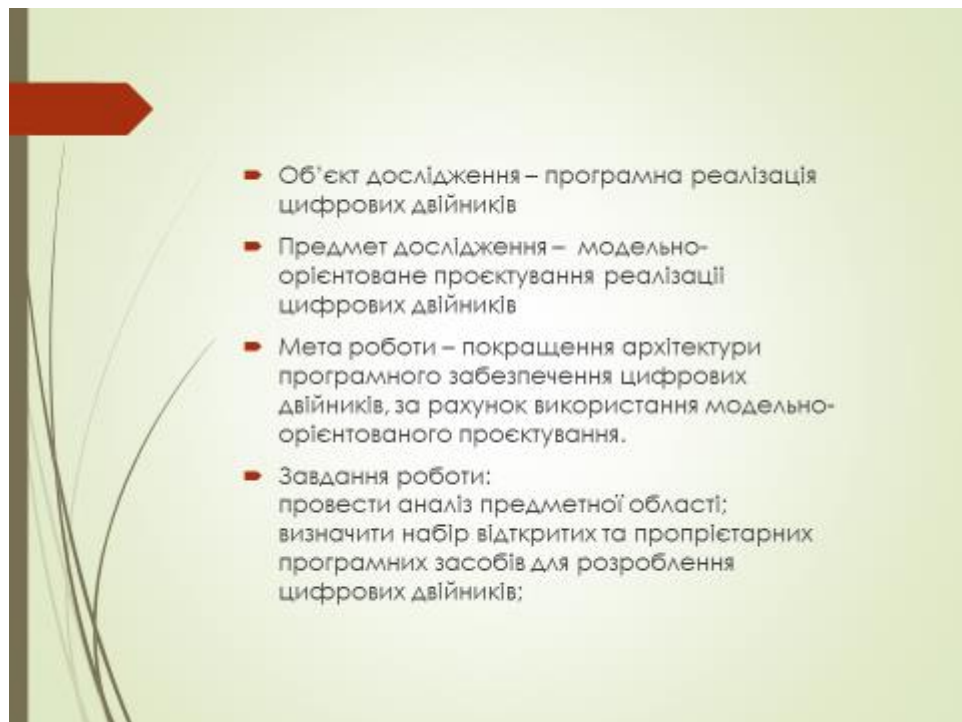


Рисунок А.2 – Слайд 2



Рисунок А.3 – Слайд 3



Рисунок А.4 – Слайд 4

Потоки даних для цифрового двійника

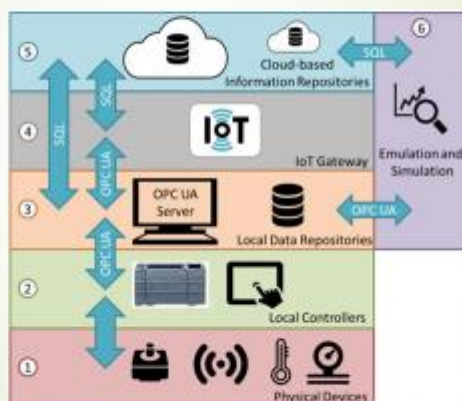


Рисунок А.5 – Слайд 5

Схема работы EclipseDitto

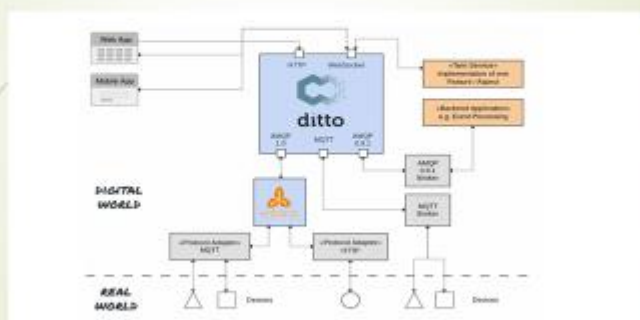


Рисунок А.6 – Слайд 6

Рішення компонентів Hono

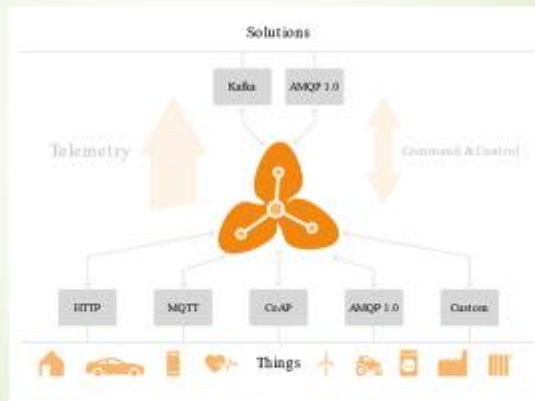


Рисунок А.7 – Слайд 7

Платформа підключень з відкритим кодом

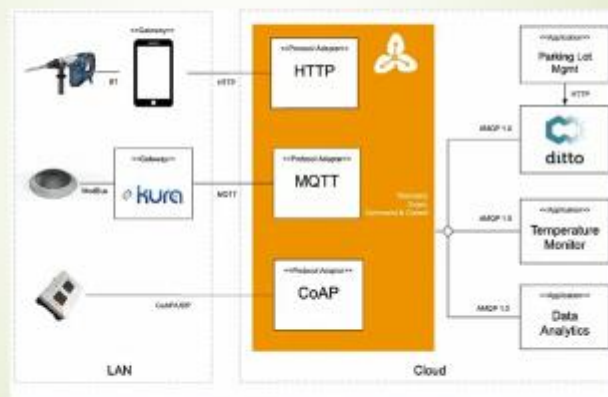


Рисунок А.8 – Слайд 8



Рисунок А.9 – Слайд 9

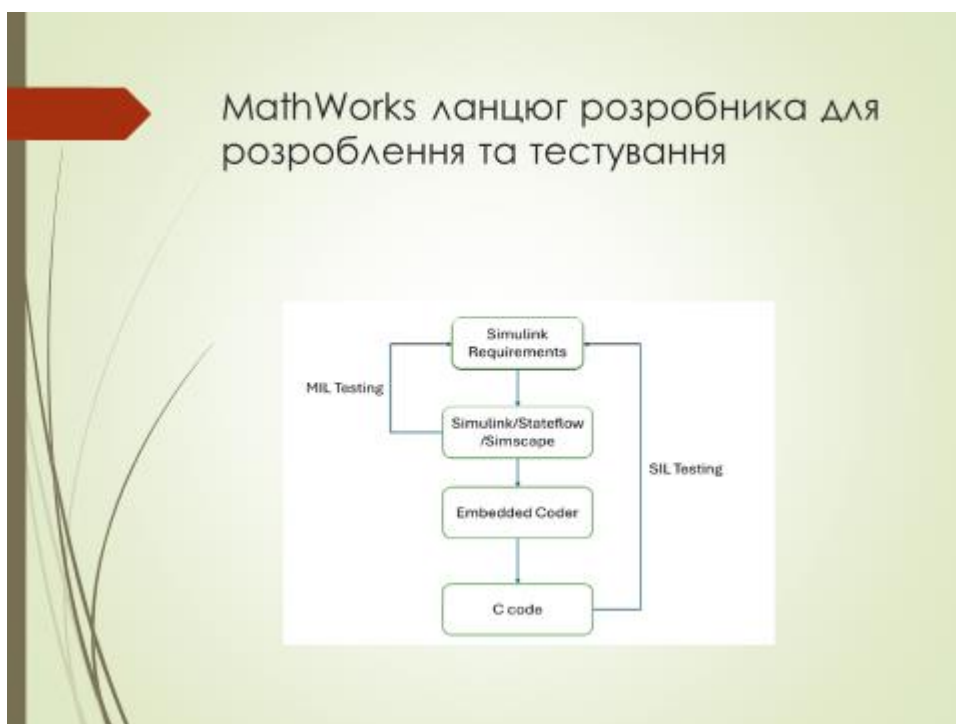


Рисунок А.10 – Слайд 10

Розгортання хмарного кластера на основі Kubernetes

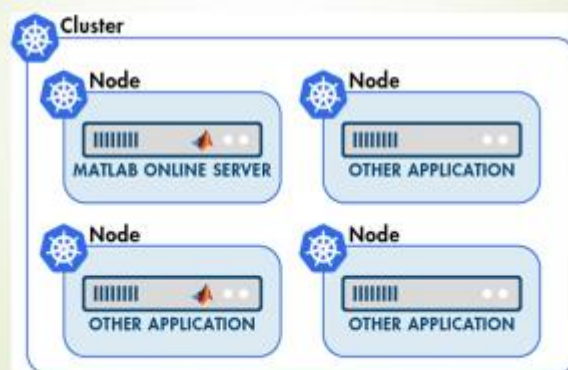


Рисунок А.11 – Слайд 11

Використання NGINX Ingress Controller

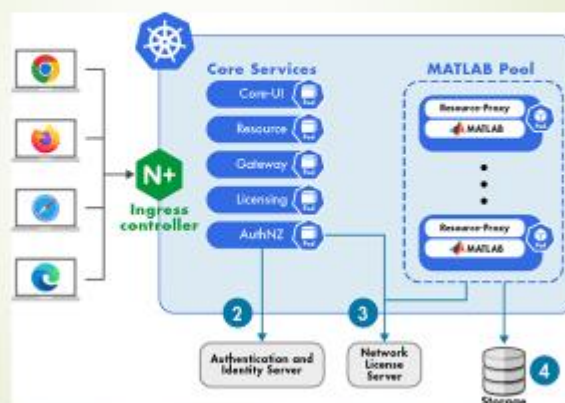


Рисунок А.12 – Слайд 12

Калібрування даних

Рисунок А.14 – Слайд 14

Віртуальний автомобіль



Рисунок А.15 – Слайд 15

Архітектура запропонованого цифрового двійника

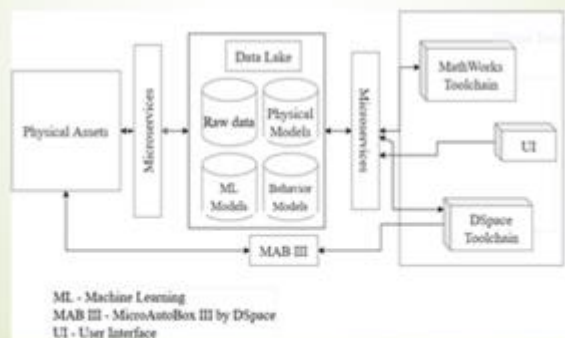


Рисунок А.16 – Слайд 16

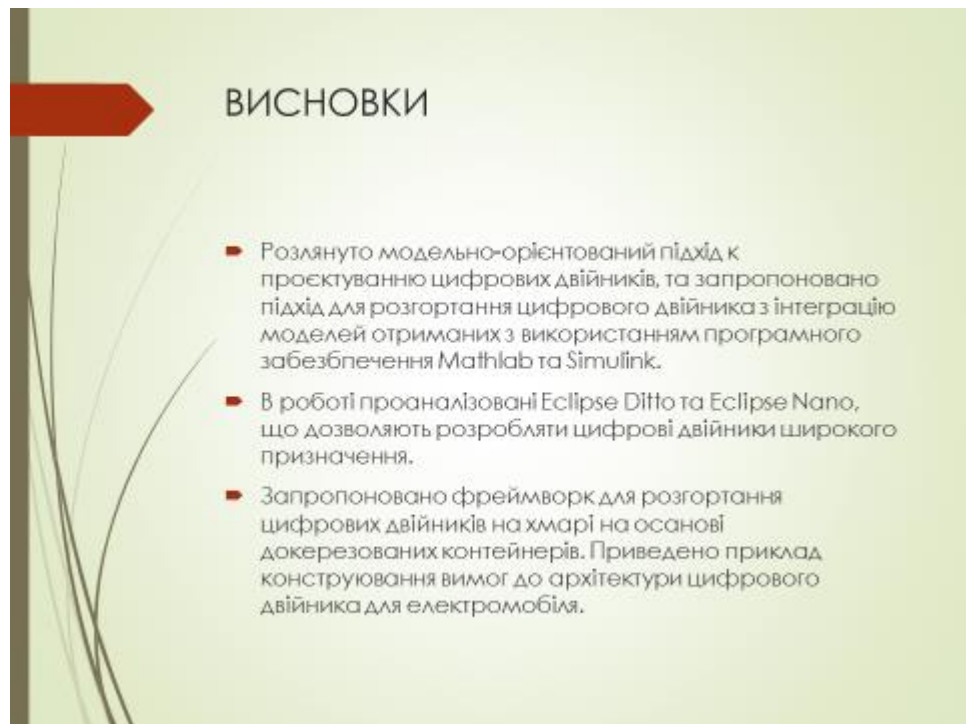


Рисунок А.17 – Слайд 17