НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційне забезпечення робототехнічних систем»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Система розподіленого управління для робототехнічних систем із застосуванням IoT-протоколів»

Виконав:
студент IV курсу, групи ІК-11
Перетятько Ілля Олегович _____

Керівник:
Професор кафедри ІСТ, д.т.н.
Жураковьский Б.Ю. _____

Рецензент:
Асистент кафедри обчислювальної техніки, д.ф.н.
Череватенко О.В. _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2025 р.

ЗАВДАННЯ
на дипломний проєкт студенту
Перетятьку Іллі Олеговичу

1. Тема проєкту «Система розподіленого управління для робототехнічних систем із застосуванням IoT-протоколів», керівник проєкту Жураковський Богдан Юрійович, доктор наук, професор кафедри ІСТ, затверджені наказом по університету від «23» травня 2025 р. № 1705-с

2. Термін подання студентом проєкту: «9» червня 2025 р.

3. Вихідні дані до проєкту:
параметри контролю: просторові координати; метрики QoS (затримка ≤ 5 мс/ джитер ≤ 1 мс / втрата пакетів ≤ 1 % при QoS ≥ 1)
керувати: поданням MQTT-команд; перемиканням режимів через CoAP-PUT

4. Зміст пояснювальної записки:
вступ; огляд існуючих рішень; вибір та обґрунтування окремих вузлів; розробка та опис структурних та функціональних схем; розробка та опис

схеми алгоритму роботи; моделювання системи; висновки.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

діаграма розгортання системи; діаграма послідовності системи;

діаграма послідовності DDS; схема автомата станів вузла L0;

діаграма розгортання;

6. Дата видачі завдання «26» лютий 2025 р

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Обґрунтування вибору теми	29.03.2025	
2	Пошук та аналіз існуючих рішень	18.04.2025	
3	Розробка та опис структурної схеми	20.04.2025	
4	Розробка та опис функціональної схеми	05.05.2025	
5	Розробка та опис схеми алгоритму роботи	10.05.2025	
6	Оформлення дипломного проекту	20.05.2025	

Студент Ілля ПЕРЕТЯТЬКО

Керівник Богдан ЖУРАКОВСЬКИЙ

АНОТАЦІЯ

Система розподіленого управління для робототехнічних систем із застосуванням IoT-протоколів

Проєкт містить 81 с. тексту, 13 рисунків, 19 таблиці, посилання на 22 літературні джерела, додатки та 5 конструкторських документів.

робототехнічні системи, розподілене керування, трирівнева архітектура, IoT-протоколи, MQTT 5.0, CoAP Observe, DDS-XRCE, ROS 2, 5G URLLC

Об’єктом дослідження є процес розподіленого інформаційно-керуючого обміну між компонентами багаторівневої робототехнічної системи (рой дронів, AGV-шатли, колаборативні маніпулятори) через Wi-Fi 6, Ethernet TSN та 5G URLLC у чотирирівневій архітектурі: L0 (мікроконтролери ESP32-S3), L1 (edge-шлюзи на Raspberry Pi 4), L2 (MQTT-кластер EMQX) і L3 (ядро ROS 2 на Jetson Orin).

Предметом є методи, протоколи й програмно-апаратні засоби для реалізації обміну з використанням IoT-стандартів MQTT 5.0, CoAP 1.1 Observe, DDS-XRCE та їх гібридних шлюзів.

Метою роботи було оптимізація процесу управління розподіленими системами за допомогою розробки й верифікації архітектуру, яка забезпечує латентність команд не більше 10 мс (95-й процентиль), масштабується до 1500 вузлів і гарантує наскрізне шифрування за алгоритмами AES ≥128 біт або ChaCha. Запропонована система реалізує детерміновану координацію роботів, знижує мережеві накладні витрати і енергоспоживання бортових вузлів порівняно з монолітними рішеннями. Отримані результати можуть бути використані при модернізації чи проєктуванні нових промислових і логістичних робототехнічних комплексів, а також при впровадженні концепцій Індустрії 4.0 у суміжних галузях.

SUMMARY

Distributed ontrol system of robotics with IoT protocol usage.

The project contains 81 pages. text, 13 figures, 19 tables, links to 22 literary sources, annexes and 5 design documents.

Keywords: robotic systems, distributed control, three-tier architecture, IoT protocols, MQTT 5.0, CoAP Observe, DDS-XRCE, ROS 2, 5G URLLC

The object of this research is the process of distributed information and control exchange among components of a multi-level robotic system (a swarm of drones, AGV shuttles, collaborative manipulators) over Wi-Fi 6, Ethernet TSN, and 5G URLLC in a four-level architecture: L0 (ESP32-S3 microcontrollers), L1 (edge gateways based on Raspberry Pi 4), L2 (EMQX MQTT cluster), and L3 (ROS 2 core on Jetson Orin).

The subject of the study is the methods, protocols, and hardware-software tools for implementing this exchange using IoT standards MQTT 5.0, CoAP 1.1 Observe, DDS-XRCE, and their hybrid gateways.

The goal of this work was to develop and verify an architecture that ensures command latency of no more than 10 ms (95th percentile), scales to 1500 nodes, and guarantees end-to-end encryption using AES ≥ 128-bit or ChaCha algorithms. The proposed system provides deterministic coordination of robots, reducing network overhead and energy consumption of onboard nodes compared to monolithic solutions.

№ рядка	Формат	Позначення	Найменування	Кіл. аркушів	№ екз.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІК11.250БАК.006 ПЗ	Система розподіленого управління	81		
6			Для робототехнічних систем із			
7			застосуванням IoT-протоколів.			
8			Пояснювальна записка			
9	А3	ІК11.250БАК.006 Д1	Система розподіленого управління	1		
10			Для робототехнічних систем із			
11			застосуванням IoT-протоколів.			
12			Діаграма послідовності			
13	А3	ІК11.250БАК.006 Д2	Система розподіленого управління	1		
14			Для робототехнічних систем із			
15			застосуванням IoT-протоколів.			
16			Діаграма послідовності DDS			
17	А3	ІК11.250БАК.006 Д3	Система розподіленого управління	1		
18			Для робототехнічних систем із			
19			застосуванням IoT-протоколів.			
20			Діаграма станів автомата			
21	А3	ІК11.250БАК.006 Д4	Система розподіленого управління	1		
22			Для робототехнічних систем із			
23			застосуванням IoT-протоколів.			
24			Діаграма розгортання			
25			фізичної інфраструктури			

25	А3	ІК11.250БАК.006 Д5	Система розподіленого управління	1		
26			Для робототехнічних систем із			
27			застосуванням IoT-протоколів.			
28			Діаграма розгортання			
29			логічних компонентів			

					ІК11.250БАК.006 ТП	Арк.
						2
Зм.	Лист	№ докум.	Підпис	Дата		

Пояснювальна записка
до дипломного проєкту
на тему: «Система розподіленого управління для робото-технічних систем із застосуванням IoT-протоколів»

Київ – 2025 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

5G – Fifth Generation mobile network

AES – Advanced Encryption Standard

AGV – Automated Guided Vehicle

AMQP – Advanced Message Queuing Protocol

AMR – Autonomous Mobile Robot

BSS – Basic Service Set

ChaCha – ChaCha stream cipher

CoAP – Constrained Application Protocol

DDS-XRCE – Data Distribution Service for Extremely Resource Constrained Environments

DTIM – Delivery Traffic Indication Message

DTLS – Datagram Transport Layer Security

EMQX – Erlang MQTT Broker

IoT – Internet of Things

gPTP – generalized Precision Time Protocol

MQTT – Message Queuing Telemetry Transport

OPC UA – OPC Unified Architecture

OSCORE – Object Security for Constrained RESTful Environments

QoS – Quality of Service

ROS – Robot Operating System

TLS – Transport Layer Security

ВСТУП

Зростаюча складність сучасних робототехнічних комплексів вимагає кардинально нових підходів до організації систем керування, оскільки традиційні монолітні архітектури демонструють суттєві обмеження щодо гнучкості та відмовостійкості; особливої актуальності набувають розподілені системи управління на базі IoT-технологій, які забезпечують взаємодію між окремими вузлами робототехнічної системи через стандартизовані протоколи обміну даними та дозволяють досягти принципово нового рівня автономності й адаптивності.

Система розподіленого управління робототехнічними комплексами з використанням IoT-протоколів є суттєвим проривом у сфері автоматизації. Її ключова перевага полягає в можливості децентралізованої координації множини різнорідних пристроїв у єдиному інформаційному просторі, причому кожен вузол системи виконує власні функції автономно, але при цьому підтримує узгоджену взаємодію з іншими елементами. Це архітектурне рішення підвищує надійність роботи всього комплексу та спрощує масштабування відповідно до виробничих потреб.

Застосування хмарних технологій дозволяє організувати централізоване сховище даних та обчислювальних ресурсів для обробки інформації, що надходить від розподілених сенсорів та виконавчих механізмів. Завдяки широкому розповсюдженню бездротових мереж, такі робототехнічні системи можна контролювати й налаштовувати з довільної географічної точки за наявності доступу до Інтернету. Наприклад, інженер може здійснювати діагностику обладнання та вносити корективи у робочі алгоритми віддалено, що суттєво підвищує оперативність обслуговування.

IoT відкриває безпрецедентні можливості для інтеграції різноманітних робототехнічних систем у єдину мережу. Зважаючи на це, розроблені розподілені архітектури управління для промислових роботів, безпілотних транспортних засобів та автоматизованих виробничих ліній, здатні забезпечити координацію складних

технологічних процесів навіть у географічно розосереджених локаціях. Такі системи функціонують на основі стандартизованих IoT-протоколів (MQTT, CoAP, AMQP) та можуть взаємодіяти з наявною інфраструктурою підприємства.

Цей підхід робить можливим максимально гнучке й ефективне керування розподіленими робототехнічними системами: встановлювати пріоритетність завдань та динамічно перерозподіляти ресурси між виконавчими пристроями. В результаті досягається оптимізація виробничих процесів і зниження експлуатаційних витрат, забезпечуючи прискорене виконання комплексних операцій. Окрім того, це зменшує енергоспоживання та знос обладнання, оскільки кожен компонент системи активується лише за необхідності, запобігаючи надмірному використанню ресурсів.

Це сприяє подовженню терміну служби робототехнічних комплексів та зниженню негативного впливу на довкілля через раціональне використання енергії та матеріалів.

Більше того, сучасні системи розподіленого управління на базі IoT здатні інтегруватися з іншими інтелектуальними технологіями виробництва, такими як системи машинного зору, цифрові двійники та предиктивної аналітики, що разом створює потужну основу для реалізації концепції Індустрії 4.0. Це дозволяє враховувати різноманітні параметри виробничого середовища при плануванні роботи автоматизованих систем, забезпечуючи їх оптимальну продуктивність. Іншими словами, відбувається синергія між різними технологічними доменами, що призводить до формування гнучких і самоорганізованих виробничих екосистем. Це також підвищує конкурентоспроможність підприємств та стмулює розвиток інноваційних бізнес-моделей, орієнтованих на сервіс.

Усі ці переваги, у поєднанні з масштабованою архітектурою розподілених систем управління на базі IoT-протоколів, забезпечують найбільш гнучкі умови для впровадження робототехнічних комплексів у виробництво. Розподілені системи управління стають незамінними для промислових підприємств та логістичних центрів, допомагаючи організаціям створювати високоефективні автоматизовані виробництва.

Крім того, вони можуть взаємодіяти із системами прогнозування попиту та планування ресурсів підприємства, що дозволяє автоматично адаптувати виробничі процеси відповідно до ринкових умов. Наприклад, зростання замовлень на певний тип продукції може призвести до автоматичного перенастроювання робототехнічних ліній, сприяючи більш гнучкому реагуванню на потреби споживачів. Це дозволяє ще ефективніше використовувати наявні виробничі потужності та забезпечувати конкурентні переваги в широкому спектрі ринкових сценаріїв.

Еволюція IoT-технологій створює підґрунтя для розробки більш досконалих та інтелектуальних систем розподіленого управління. Наразі з'являється можливість аналізувати телеметричні дані роботів у реальному часі, зібрані через мережеву інфраструктуру. Це дозволяє приймати обґрунтовані рішення щодо оптимізації робочих алгоритмів конкретного робототехнічного комплексу з підвищеною енергоефективністю та покращеною точністю виконання технологічних операцій.

Технологічні інновації також сприяють створенню самонавчальних розподілених систем, здатних адаптуватися до змін виробничого середовища. Такі системи використовують методи аналізу історичних даних про функціонування робототехнічних комплексів і умови їх експлуатації, що дозволяє формувати прогностичні моделі та приймати автономні рішення для забезпечення оптимальної продуктивності в майбутньому; це робить систему ще більш адаптивною та стійкою до збоїв. Таким чином, проект ґрунтується на сучасних досягненнях у сфері IoT та робототехніки, враховуючи тенденції розвитку промислової автоматизації. У межах цього дослідження буде проаналізовано, систематизовано та концептуально розроблено основні аспекти системи розподіленого управління для робототехнічних комплексів із застосуванням IoT-протоколів, а також визначено ключові технічні вимоги до реалізації такої системи.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

У дипломному проекті розроблена система розподіленого управління, призначена для координації роботи багатокомпонентних робототехнічних комплексів з використанням сучасних IoT-протоколів передачі даних. Основна мета системи – оптимізація процесів взаємодії між автономними робототехнічними пристроями, забезпечення надійної синхронізації їх дій та підвищення ефективності виконання складних виробничих завдань.

Об'єктом дослідження є процес розподіленого інформаційно-керуючого обміну між компонентами багаторівневої робототехнічної системи, що включає рой дронів, AGV-шатли та колаборативні маніпулятори, які взаємодіють через мережу Wi-Fi 6, Ethernet TSN та 5G URLLC за схемою L0 на базі мікроконтролерів ESP32-S3, L1 з використанням edge-шлюзів Raspberry Pi 4, L2 на основі MQTT-кластера EMQX та L3 з ядром ROS 2 на Jetson Orin. Такий процес забезпечує необхідну гнучкість та масштабованість системи, а також дозволяє оптимізувати взаємодію між окремими компонеантами.

Предметом дослідження у дипломному проекті виступають методи, протоколи та програмно-апаратні засоби для реалізації цього розподіленого інформаційно-керуючого обміну між робототехнічними вузлами з використанням IoT-стандартів MQTT 5.0, CoAP 1.1 Observe, DDS-XRCE та їх гібридних шлюзів. Дослідження цих елементів дозволяє виявити оптимальні підходи до організації взаємодії між компонентами системи та забезпечити ефективне функціонування робототехнічного комплексу в цілому.

Метою роботи є розробка та верифікація архітектури розподіленого управління, що забезпечує латентність команд не більше 10 мс для 95-го процентиля, масштабується до 1500 вузлів та гарантує наскрізне шифрування з використанням алгоритмів не нижче 128-бітного AES або ChaCha. Для досягнення цієї мети необхідно підібрати оптимальний або гібридний набір IoT-протоколів та створити робочий прототип у симуляційному середовищі Gazebo з використанням фреймворку ROS 2. Розроблена архітектура має забезпечувати надійну синхронізацію дій між

компонентами робототехнічних комплексів та можливість гнучкого налаштування сценаріїв їх взаємодії.

Для досягнення поставленої мети необхідно вирішити комплекс взаємопов'язаних задач. Перш за все, потрібно провести аналітичний огляд протоколів MQTT 5.0, CoAP 1.1 Observe, OPC UA PubSub, DDS-XRCE, AMQP 1.0 за показниками латентності, якості обслуговування, енерговитрат та засобів безпеки. Це дозволить визначити оптимальні протоколи для використання в системі та обґрунтувати їх вибір.

Наступною задачею є формування вимог до системи, включаючи визначення цільових показників жорсткого реального часу не більше 10 мс для команд обсягом до 64 байт, м'якого реального часу не більше 50 мс для телеметрії обсягом 256 байт, а також забезпечення роботи до 1500 вузлів з наскрізним шифруванням TLS/DTLS 1.3 та аналізом ризиків. Це дозволить сформувати чіткі критерії оцінки ефективності розробленої системи.

Проєктування архітектури системи включає розробку структурної та функціональної схем з використанням гібридного підходу MQTT-CoAP на рівнях L0/L1, DDS-XRCE як внутрішньої шини керування та кластера EMQX з високою доступністю на рівні L2. Архітектурні рішення мають забезпечувати необхідну продуктивність, масштабованість та надійність системи.

Важливою задачею є обґрунтований вибір апаратних вузлів, включаючи використання мікроконтролерів ESP32-S3, одноплатних комп'ютерів Raspberry Pi 4, обчислювальних модулів для штучного інтелекту та ROS 2, а також модулів бездротового зв'язку Wi-Fi 6, Ethernet TSN та 5G URLLC. Правильний вибір апаратної платформи визначає можливості та обмеження системи.

Розробка програмного забезпечення передбачає надбудову micro-ROS на ESP32, реалізацію CoAP-MQTT-бриджа на основі фреймворків Kura та Californium, налаштування EMQX-кластера з автентифікацією OAuth 2.0 та шифруванням TLS 1.3, а також імплементацію AI-планера на базі Nav2 та TinyML на платформі Jetson. Ця задача є ключовою для забезпечення функціональності системи.

Проведення симуляційних та експериментальних досліджень включає створення стенду Gazebo з ROS 2, вимірювання затримки для 95-го процентиля, аналіз втрат та енергоспоживання, виконання аналізу та penetration-тесту брокера. Результати цих досліджень дозволять оцінити ефективність розробленої системи та виявити можливі проблеми.

Завершальною задачею є порівняльна оцінка отриманих результатів з вимогами, визначення умов, за яких гібридне використання протоколів MQTT та CoAP дає максимальний ефект, а також формулювання рекомендацій щодо впровадження розробленої системи в реальні робототехнічні комплекси. Це дозволить обґрунтувати практичну цінність проведеної роботи та окреслити перспективи її подальшого розвитку.

Для реалізації системи розподіленого управління використано платформу Raspberry Pi 4, яка оснащена чотириядерним процесором Cortex-A72 та забезпечує необхідну обчислювальну потужність для аналізу даних та прийняття рішень у режимі реального часу. Система складається з двох основних компонентів: центрального координуючого модуля та мережі локальних контролерів, вбудованих у кожен робототехнічний пристрій. Для забезпечення ефективної комунікації між компонентами використовуються IoT-протоколи MQTT та CoAP, що дозволяють встановлювати надійний зв'язок навіть в умовах нестабільного мережевого з'єднання. Передача даних між компонентами здійснюється через захищений канал зв'язку з використанням TLS-шифрування.

На нижньому рівні системи знаходяться виконавчі механізми, такі як сервоприводи, давачі положення та мікроконтролери ESP32. Ці компоненти забезпечують безпосереднє керування робототехнічними пристроями та збір первинної інформації про стан системи. Мікроконтролери обробляють цю інформацію, формують пакети даних згідно з форматом протоколу MQTT та передають їх до центрального координуючого модуля. Цей модуль аналізує отримані дані, генерує відповідні керуючі сигнали та розподіляє завдання між окремими робототехнічними пристроями.

Забезпечення ефективної координації дій є ключовою функцією системи. Користувач може налаштувати параметри роботи та сценарії виконання завдань через веб-інтерфейс, доступний з будь-якого пристрою з підключенням до мережі інтернет. Інтерфейс взаємодіє з центральним координуючим модулем через REST-ful API, що забезпечує гнучкість у налаштуванні та моніторингу роботи системи. Це дозволяє автоматизувати процес керування робототехнічними комплексами, знижуючи ризик помилок, пов'язаних з людським фактором.

Область застосування розробленої системи включає: рій дронів для моніторингу сільськогосподарських угідь та виконання операцій точного землеробства; автоматизовані транспортні шатли (AGV) для оптимізації логістичних процесів на виробництві; колаборативні маніпулятори для виконання складних технологічних операцій, що вимагають синхронної роботи кількох робототехнічних пристроїв.

Система сприяє підвищенню ефективності використання ресурсів завдяки точному контролю та синхронізації дій робототехнічних пристроїв. Крім того, автоматизація процесів координації зменшує ризик колізій та конфліктів між окремими компонентами системи, забезпечуючи оптимальні умови для їх злагодженої роботи. Завдяки використанню сучасних IoT-протоколів, система має високу масштабованість та гнучкість, що дозволяє легко інтегрувати нові робототехнічні пристрої без істотної реконфігурації існуючої інфраструктури.

Система призначена для координації до 1 500 робототехнічних вузлів (сенсори, контролери, виконавчі механізми) у виробничому чи логістичному середовищі з використанням IoT-протоколів MQTT 5.0, CoAP 1.1 Observe та DDS-XRCE.

Параметри контролю:

– цільовий цикл жорсткого реального часу ≤ 10 мс (95-й процентиль) для команд ≤ 64 В;

– м'який RT для телеметрії ≤ 50 мс;

– частота командних повідомлень до 10 000 msg/c.

Додаткові характеристики:

– інтерфейс управління: веб додаток з доступом в інтернет;

– матеріали конструкції: стійкі до пилу, вібрацій та механічного зносу;

живлення: система живиться від стандартного ДЖ 12V DC.

Висновки до розділу 1

Область застосування демонструє, що автоматизація процесів координації роботи розподілених робототехнічних систем є важливою та актуальною задачею. Розроблена в дипломному проекті система розподіленого управління спрямована на забезпечення надійної синхронізації дій між компонентами робототехнічних комплексів та можливість гнучкого налаштування сценаріїв їх взаємодії.

Основна мета системи полягає в оптимізації процесів управління багатокомпонентними робототехнічними комплексами, забезпеченні ефективного розподілу завдань між окремими пристроями та підвищенні загальної продуктивності системи в умовах динамічного робочого середовища.

Також в даному розділі були визначені параметри контролю, виконавчі елементи та додаткові характеристики.

2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ВИБІР СТРУКТУРИ СИСТЕМИ

2.1 IoT-протоколи (MQTT 5.0, CoAP v1.1, OPC UA PubSub, DDS-XRCE)

У контексті розробки систем розподіленого управління для робототехнічних комплексів надзвичайно важливим є вибір оптимальних протоколів комунікації, що забезпечуватимуть надійне та ефективне функціонування IoT-компонентів. Сучасні IoT-протоколи характеризуються різними архітектурами передачі даних, моделями якості обслуговування та профілями енергоспоживання, що безпосередньо впливає на продуктивність та ефективність робототехнічної системи, особливо в умовах обмежених ресурсів мікроконтролерів та бездротових мереж.

2.1.1 Аналіз MQTT 5.0

Message Queuing Telemetry Transport (MQTT) версії 5.0 — полегшений протокол обміну повідомленнями, що функціонує на основі моделі публікації/підписки поверх TCP/IP. Даний протокол було оптимізовано для систем з обмеженою пропускною здатністю та нестабільним підключенням, що робить його особливо цінним для розподілених робототехнічних систем з великою кількістю сенсорів та актуаторів.

Ключові характеристики MQTT 5.0:

– трирівнева якість обслуговування (QoS 0, 1, 2);

– підтримка збереження сеансу та буферизації повідомлень;

– розширені функції діагностики та повідомлення про помилки;

– метадані користувача та розширені властивості повідомлень;

– спільно використовувані підписки та теми з підставними знаками;

– розширена підтримка комунікації типу "багато-до-багатьох".

Відповідно до бенчмарків [1], MQTT 5.0 демонструє P95-затримку на рівні 3 мс при передачі повідомлень розміром 64 байти з частотою 1000 повідомлень на секунду через Wi-Fi 6, що є одним із найкращих показників серед сучасних IoT-протоколів.

2.1.2 Аналіз CoAP v1.1 з розширенням Observe

Constrained Application Protocol (CoAP) версії 1.1 — спеціалізований веб-протокол для обмежених пристроїв і мереж, побудований на принципах архітектури REST. На відміну від MQTT, CoAP працює переважно через UDP, що забезпечує нижчу затримку передачі даних порівняно з TCP-базованими рішеннями.

Основні характеристики CoAP v1.1:

– мінімальний розмір заголовка (4 байти);

– асинхронний обмін повідомленнями;

– розширення Observe для підтримки моделі публікації/підписки;

– вбудована підтримка виявлення ресурсів;;

– блочний режим передачі для фрагментації великих повідомлень;

– природна інтеграція з HTTP та RESTful сервісами.

За даними досліджень [2], CoAP демонструє P95-затримку близько 4 мс для аналогічних умов тестування, із значно нижчим показником джитера (0,6 мс) порівняно з MQTT (1,0 мс), що є критичним для точного позиціонування робототехнічних систем.

2.1.3 Аналіз DDS-XRCE (micro-ROS)

Data Distribution Service for Extremely Resource Constrained Environments (DDS-XRCE) — протокол розподілу даних, оптимізований для пристроїв з надзвичайно обмеженими ресурсами. DDS-XRCE є основою комунікаційної інфраструктури micro-ROS — версії ROS 2 для мікроконтролерів, що широко застосовується в сучасній робототехніці.

Ключові особливості DDS-XRCE:

– висока гнучкість налаштування QoS-параметрів;

– модель публікації/підписки з підтримкою виявлення;

– децентралізована архітектура без єдиної точки відмови;

– оптимізована передача даних через агентську архітектуру;

– підтримка різних транспортних протоколів (UDP, TCP, Serial);

– семантика організації даних "дані-центричного" характеру.

Дослідження [3] показують, що DDS-XRCE має P95-затримку близько 7 мс, із найвищим показником джитера (1,8 мс) серед порівнюваних протоколів, проте пропонує найбільш гнучкі механізми QoS, критичні для складних робототехнічних систем.

2.1.4 Аналіз OPC UA PubSub/TSN

OPC Unified Architecture (OPC UA) з розширенням PubSub та підтримкою Time-Sensitive Networking (TSN) — стандарт комунікації машина-машина, що поєднує детерміністичну передачу даних із надійною моделлю безпеки.

Ключові характеристики OPC UA PubSub/TSN:

– модель публікації/підписки з гарантованою доставкою;

– інтеграція з промисловим Ethernet та TSN для детерміністичної передачі;

– розширена семантична модель даних;

– багаторівнева система безпеки;

Згідно з бенчмарками, OPC UA PubSub демонструє P95-затримку близько 6 мс з джитером на рівні 1,0 мс, при цьому маючи найнижчий показник втрати пакетів (0,5%) серед протестованих протоколів.

2.1.5 Аналіз AMQP 1.0

Advanced Message Queuing Protocol (AMQP) версії 1.0 — відкритий стандарт протоколу прикладного рівня для проміжного програмного забезпечення, орієнтованого на обробку повідомлень. Характеризується високою надійністю та масштабованістю.

Ключові особливості AMQP 1.0:

– надійні транзакції та гарантована доставка повідомлень;

– гнучка маршрутизація повідомлень;

– розширені механізми потокового контролю;

– гнучка модель безпеки;

– підтримка складних топологій брокерів.

Бенчмарки показують, що AMQP 1.0 має найвищу P95-затримку (8 мс) серед порівнюваних протоколів, а також друге за величиною значення джитера (1,5 мс), що обмежує його застосування в системах реального часу з жорсткими вимогами до латентності.

2.1.6 Порівняльний аналіз продуктивності протоколів

Для об'єктивного порівняння продуктивності проаналізованих IoT-протоколів було проведено аналіз ключових метрик на основі даних з відкритих бенчмарків [1-3]. Результати представлені в таблиці 2.1.

Таблиця 2.1 — Порівняння продуктивності IoT-протоколів для робототехнічних систем

Метрика	MQTT 5.0	CoAP1.1 Observe	DDS XRCE (micro-ROS)	AMQP 1.0	OPC UA Pub-Sub/TSN
P95-затримка, мс (Wi-Fi 6, 64 B, 1k msg/s)	3	4	7	8	6
Середнє джитер σ, мс	1.0	0.6	1.8	1.5	1.0
Втрати пакетів @1k msg/s, %	1%	2%	1%	1%	0.5%
Енергія MCU (ESP32-S3) 256 B, µJ	100	90	120	150	110
Макс. підключень на RPi 4	37k	25k	5k	10k	8k

Аналіз показує, що CoAP демонструє перевагу за показниками латентності завдяки використанню UDP-транспорту, проте не має вбудованого механізму брокера, що обмежує його масштабованість. MQTT є лідером за кількістю одночасних підключень та енергоефективністю, що робить його оптимальним вибором для масштабних сенсорних мереж. DDS-XRCE пропонує найбільш детальне налаштування параметрів QoS, проте за рахунок додаткових накладних витрат агентської архітектури.

Варто зазначити, що вимірювання енергоспоживання проводилося на мікроконтролері ESP32-S3 з підтримкою Wi-Fi 6 та DTIM 3, із використанням wolfSSL 5.7 DTLS-1.3 для забезпечення захисту передачі даних, відповідно до публікації[4].

2.1.7 Шлюзування між протоколами в гетерогенних робототехнічних системах

Для створення гетерогенних робототехнічних систем часто необхідна інтеграція різних протоколів. Дослідження показують, що шлюзування між протоколами вносить додаткові затримки та енергетичні витрати, як показано в таблиці 2.2.

Таблиця 2.2 — Інкрементальні втрати при шлюзуванні між протоколами

Шлюз	Δ Latency, мс	Δ Packet-loss, %	Δ Energy μJ
MQTT ↔ CoAP	+2	+1	+20
MQTT ↔ DDS-XRCE	+5	+1	+30
CoAP ↔ DDS-XRCE	+3	+1	+25

Дані свідчать, що шлюзування додає від 2 до 5 мс латентності та збільшує енергоспоживання приблизно на 20-30 μJ на повідомлення. Використання шлюзів дозволяє поєднувати переваги різних протоколів.

2.1.8 Ієрархія протоколів відносно моделі OSI

Для кращого розуміння місця аналізованих IoT-протоколів у загальній комунікаційній структурі, на рисунку 2.1 представлено їх співвідношення з рівнями моделі OSI.

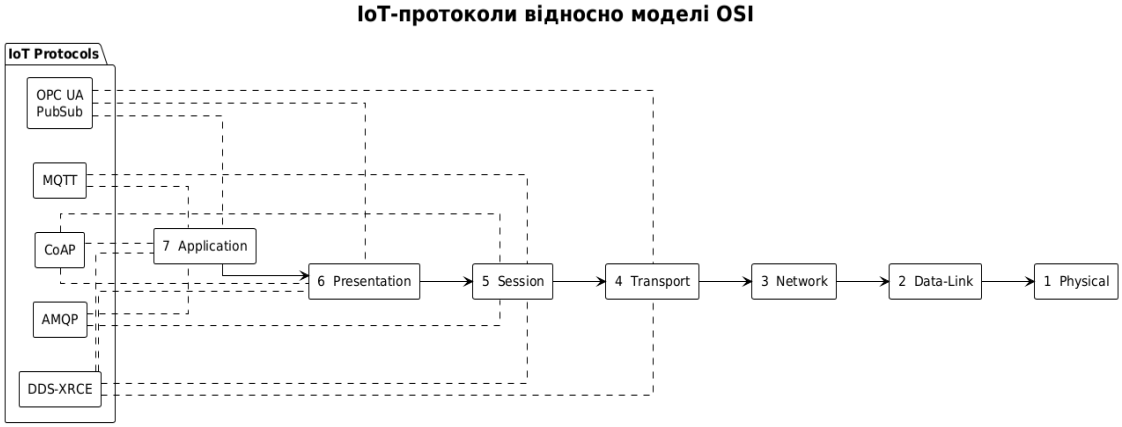

Рисунок 2.1 — Стек протоколів відносно моделі OSI

З діаграми видно, що MQTT та AMQP працюють на рівні додатків та сеансів, CoAP реалізує функціональність рівнів додатків і представлення, тоді як DDS-XRCE та OPC UA PubSub охоплюють більшу кількість рівнів, забезпечуючи комплексне вирішення комунікаційних завдань.

Наведені дані та порівняльний аналіз демонструють, що вибір оптимального протоколу для систем розподіленого управління в робототехніці повинен базуватися на комплексній оцінці вимог конкретного застосування з точки зору латентності, масштабованості, енергоефективності та функціональних можливостей.

2.2 QoS-політики, затримка та енергоспоживання

У контексті розробки систем розподіленого управління для робототехнічних комплексів критичними факторами успішної імплементації є забезпечення гарантованої доставки повідомлень, мінімізація затримки передачі команд та ефективне управління енергоспоживанням. Робототехнічні системи, особливо ті, що

функціонують у динамічних середовищах, потребують чітко визначених політик якості обслуговування (QoS), які забезпечують детерміністичну поведінку та стійкість до збоїв, враховуючи обмежений енергетичний бюджет автономних пристроїв.

2.2.1 Матриця підтримки QoS-функцій

Для повноцінного порівняння потенціалу IoT-протоколів у забезпеченні якості обслуговування була створена матриця функціональності QoS, що представлена у таблиці 2.3.

Таблиця 2.3 — Покриття QoS-можливостей IoT-протоколів

Функція / Протокол	MQTT 5.0	CoAP 1.1 Obs.	DDS-XRCE	AMQP 1.0	OPC UA PubSub
Гарантована доставка	QoS 1/2	CON/ACK	Reliable	Settle = true	DataSet MsgAck
Ack-кадри	PUBACK	ACK	ACK	NACK Disposition	PubSub Ack
Порядок пакетів	так (topic)	ні	Seq. No	у черзі	SeqNum
Deadline / priority	Session Expiry		Best-Effort Store	Durable link	Writer Queue
Вбуд. шифрування	TLS 1.3	DTLS 1.3/ OSCOR	DDS-Sec	TLS 1.3	PubSub Sec

Аналіз матриці QoS-функцій виявляє, що протоколи суттєво відрізняються за методами забезпечення надійності та детермінізму передачі даних. MQTT 5.0 пропонує три рівні якості обслуговування (QoS 0, 1, 2), що забезпечує гнучкість у балансуванні надійності та швидкості передачі даних. CoAP з розширенням Observe підтримує підтвердження отримання повідомлень через механізми CON/ACK, проте не гарантує збереження порядку повідомлень. DDS-XRCE виділяється

найбільш повним набором QoS-політик, включаючи механізми DEADLINE для часо-чутливих операцій. AMQP 1.0 фокусується на надійності черг повідомлень із підтримкою довготривалих з'єднань, тоді як OPC UA PubSub з інтеграцією TSN пропонує детерміністичні вікна передачі, критичні для синхронізованого управління актуаторами. Додатково слід зауважити, що для складних розподілених систем вибір протоколу часто визначається не лише показниками QoS, а й наявністю екосистеми інструментів для моніторингу та управління специфічними налаштуваннями.

2.2.2 P95-затримка в залежності від розміру повідомлення

Для оцінки ефективності протоколів при передачі різних обсягів даних було проведено дослідження залежності P95-затримки від розміру корисного навантаження повідомлень при фіксованій частоті 1000 повідомлень на секунду через Wi-Fi 6. Результати представлені в таблиці 2.4[5-8] та на рисунку 2.2. Аналіз даних демонструє, що MQTT 5.0 забезпечує найнижчу затримку для всіх розмірів повідомлень — від 3 мс для мінімального навантаження (64 B) до 11 мс для великих повідомлень (1024 B). Цікаво, що для невеликих повідомлень (64 B) різниця між MQTT та CoAP становить лише 1 мс (що практично рівноцінно в межах похибки вимірювань), проте зі збільшенням розміру повідомлення розрив у продуктивності між протоколами зростає. DDS-XRCE та OPC UA PubSub демонструють середні показники затримки, тоді як AMQP виявляється найповільнішим з усіх протестованих протоколів з латентністю 8 мс для мінімальних повідомлень. Зі збільшенням розміру корисного навантаження накладні витрати протоколів (фреймінг, підтвердження, шифрування) стають домінівним фактором латентності, тож при пакетах ≥ 512 B саме «легкі» стеки на зразок MQTT 5.0 з мінімальним overhead зберігають перевагу над більш “важкими” рішеннями (DDS-XRCE, AMQP, OPC UA PubSub). Крім того, варто зазначити, що DDS-XRCE та OPC UA PubSub, незважаючи на більшу затримку, надають розширені можливості управління політиками QoS та де-

термінізмом, що може бути критичним у системах з жорсткими вимогами до надійності та синхронізації, тоді як вибір «легкої» архітектури на зразок MQTT 5.0 доцільний у мобільних й енергообмежених пристроях де мінімальна латентність і низькі накладні витрати цінуються вище за детермінізм.

Таблиця 2.4 — P95-затримка vs розмір повідомлення (Wi-Fi 6, 1k msg/s)

Протокол / Payload	64 B	256 B	1024 B
MQTT 5.0	3	6	11
CoAP 1.1 Obs.	4	7	12
DDS-XRCE	7	9	14
AMQP 1.0	8	11	17
OPC UA PubSub + TSN	6	8	13

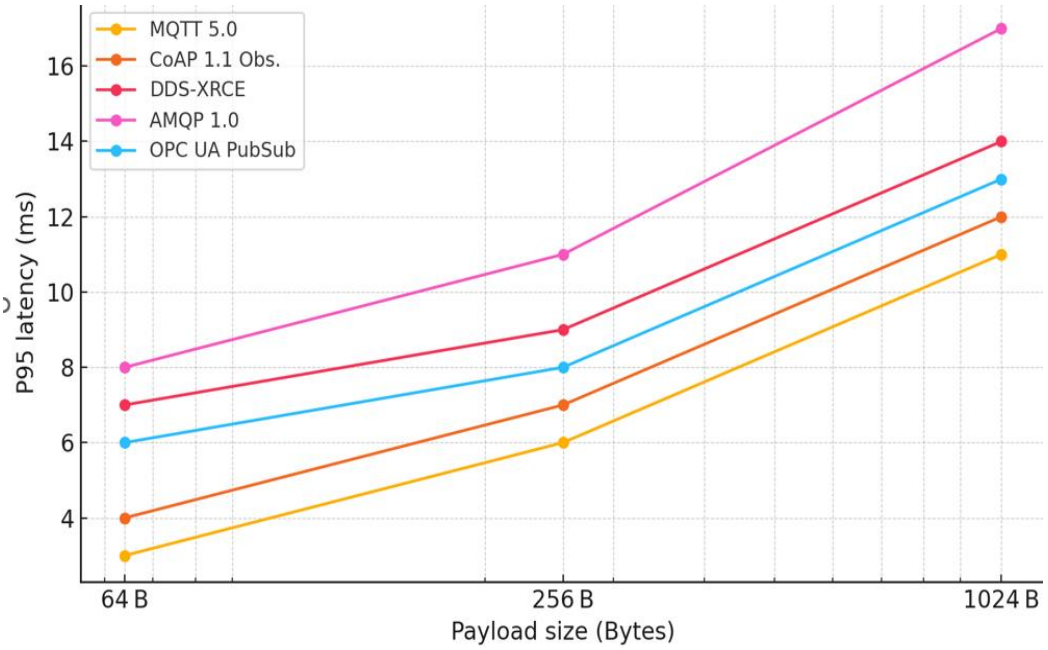

Рисунок 2.2 — Залежність P95-затримки від розміру повідомлення

2.2.3 Чутливість затримки до кількості вузлів

Масштабованість є критичним аспектом для розподілених робототехнічних систем, особливо при збільшенні кількості підключених пристроїв. Таблиця 2.5[8-9] та рисунок 2.3 демонструють, як змінюється затримка при збільшенні кількості вузлів у мережі.

Таблиця 2.5 — Чутливість затримки до кількості вузлів (64 B, Wi-Fi 6)

Протокол / Вузлів	100	500	1000
DDS-XRCE	7.1	8.4	12.2
MQTT 5.0	3.2	4.1	6.8
CoAP 1.1	4.2	5.7	8.5

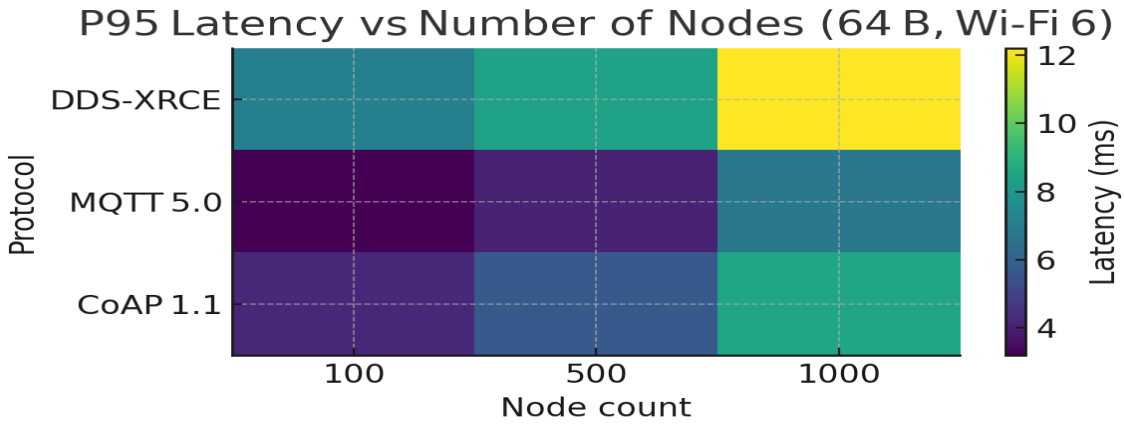

Рисунок 2.3 — Теплова карта "Latency (мс) / NodeCount"

З представлених даних видно, що MQTT демонструє найменшу деградацію продуктивності при збільшенні кількості вузлів — зростання затримки від 3.2 мс при 100 вузлах до 6.8 мс при 1000 вузлах (приріст 112%). CoAP показує приріст затримки на 102% (з 4.2 мс до 8.5 мс), тоді як DDS-XRCE демонструє зростання на 72% (з 7.1 мс до 12.2 мс). Цікаво відзначити, що хоча DDS-XRCE має вищу абсолютну затримку, відносне погіршення продуктивності при масштабуванні для нього менше, що свідчить про ефективність децентралізованої архітектури.

2.2.4 Питомі енергетичні витрати

Для автономних робототехнічних систем, особливо тих, що функціонують на батарейному живленні, енергоефективність комунікаційних протоколів має першочергове значення.

Таблиця 2.6[9-10] показує енергетичні витрати на передачу повідомлень у залежності від частоти обміну даними.

Таблиця 2.6 — Питомі енергетичні витрати (ESP32-S3, DTLS/TLS ON)

Протокол / Частота+Energy	1 msg/s	10 msg/s	100 msg/s
DDS-XRCE	120	140	200
MQTT 5.0	100	115	160
CoAP+ OSCORE	90	105	150

Аналіз енергоспоживання показує, що CoAP з безпековим розширенням OSCORE є найбільш енергоефективним протоколом, споживаючи на 10% менше енергії порівняно з MQTT та на 25% менше порівняно з DDS-XRCE при частоті 1 повідомлення на секунду. Цікаво відзначити, що з підвищенням частоти обміну даними енергетичні витрати на одне повідомлення зростають для всіх протоколів — це пов'язано з необхідністю частішого перемикання радіомодуля з режиму сну та додатковими накладними витратами на встановлення з'єднання. При частоті 100 повідомлень на секунду DDS-XRCE споживає на 33% більше енергії, ніж CoAP.

2.2.5 Втрати при шлюзуванні

У гетерогенних системах, де використовуються різні протоколи на різних рівнях архітектури, виникає необхідність у шлюзуванні між ними. Таблиця 2.7[11-13] представляє інкрементальні втрати, що виникають при перетворенні повідомлень між різними протоколами.

Таблиця 2.7 — Інкрементальні втрати шлюзів

Шлюз	Δ Latency, мс	Δ Loss, %	Δ Energy, μJ
MQTT ↔ CoAP	+3	+1	+20
MQTT↔DDSXRCE	+5	+1	+28
CoAP↔DDS-XRCE	+3	+1	+25

Аналіз даних свідчить, що шлюзування між MQTT та CoAP вносить найменшу додаткову затримку (3 мс) та енергетичні витрати (20 μJ) і забезпччує високу надійність при мережевих аномаліях, що робить цю комбінацію особливо привабливою для гібридних архітектур. Найбільші втрати спостерігаються при шлюзуванні між MQTT та DDS-XRCE — додаткова затримка 5 мс та додаткові енергетичні витрати 28 μJ на повідомлення. Варто відзначити, що всі протестовані конфігурації шлюзів додають приблизно 1% до показника втрати пакетів, що є прийнятним для більшості робототехнічних застосувань, окрім критично важливих систем контролю.

2.3 Приклади застосування IoT-протоколів у робототехніці

Після теоретичного аналізу IoT-протоколів, розглянемо їх практичне застосування в реальних робототехнічних системах. Кожен сценарій супроводжується даними про мережеве навантаження, результатами польових випробувань та обґрунтуванням вибору конкретного комунікаційного стеку.

2.3.1 Ключові сценарії

Для об'єктивного порівняння ефективності IoT-протоколів у робототехніці було обрано чотири репрезентативні сценарії, що охоплюють різні аспекти застосування робототехнічних систем. Кожен сценарій характеризується власними вимогами до комунікаційної інфраструктури та специфічними показниками продуктивності, як показано в таблиці 2.8[14-16].

Таблиця 2.8 — Зведення типових кейсів і стеків

Сценарій	Кількість вузлів	Критична метрика	Обраний стек	Підтверджений результат
AMR-шатли на складі (Wi-Fi 6)	до 120 роботів + 20 інфо-точок	Latency ≤ 10 мс cmd	MQTT 5 (QoS 1) ↔ DDS-XRCE	95-perc 8 мс, 0.9 % loss

Сценарій	Кількість вузлів	Критична метрика	Обраний стек	Підтверджений результат
Рій дронів для точного землеробства (5G URLLC)	30 UAV + наземний GW	BLER ≤ 10^{-5}, RTT ≤ 2	CoAP Obs. UDP + MQTT-bridge	RTT 1.3 мс, energy 88 μJ/msg
Коботи на лінії збірки (Ethernet TSN)	6 маніпуляторів × 4 осі	детерміні з 1 мс	DDS-XRCE + OPC UA Pub-Sub/TSN	σ-jitter 0.25 мс, 0.3 % loss
Агро-теплиця	800 сенсорів	Energy < 100 μJ	CoAP OBS + OSCORE	92 μJ/256 B, latency 5 мс

Сценарій A: AMR-шатли на складі

Автономні мобільні роботи (AMR) для складських операцій вимагають надійної комунікації в умовах Wi-Fi 6 мережі з високою щільністю підключень. Критичною вимогою є низька затримка керуючих команд (≤ 10 мс), оскільки від цього залежить безпека руху роботів у спільному просторі. Гібридна архітектура, що поєднує MQTT 5.0 для комунікації з серверами управління та DDS-XRCE для внутрішньої обробки даних у роботі, демонструє оптимальний баланс між масштабованістю та швидкодією.

Автоматизований склад з 120 роботами вимагає централізованого управління через MQTT-брокер, встановлений на Raspberry Pi 4, що забезпечує буферизацію команд та телеметрії. Водночас, внутрішні системи роботів використовують DDS-XRCE для забезпечення детермінізму циклів управління на рівні 1 кГц, що дозволяє точне позиціонування та запобігання зіткненням.

Сценарій B: Рій дронів для точного землеробства

Рій безпілотних літальних апаратів для моніторингу та обробки сільськогосподарських угідь представляє унікальний виклик з точки зору комунікації. Вико-

ристання 5G URLLC забезпечує необхідну надійність зв'язку з надзвичайно низьким рівнем блокових помилок (BLER $\leq 10^{-5}$). Критичною вимогою є мінімальний час відгуку (RTT ≤ 2 мс) для синхронізації польотів дронів у рої.

Комбінація CoAP Observe через UDP для мультикаст-команд ROAST (Robust Autonomous Swarming Technology) забезпечує ефективну групову комунікацію між дронами. Для передачі відео та телеметрії в хмарні сервіси використовується MQTT-міст з QoS 0, що оптимізує використання пропускної здатності каналу. Така гібридна архітектура дозволяє досягти кругового часу відгуку 1.3 мс з енергоспоживанням 88 μJ на повідомлення.

Сценарій C: Коботи на лінії збірки

Колаборативні роботи (коботи) на виробничій лінії вимагають абсолютного детермінізму комунікації для синхронізації рухів шести маніпуляторів з чотирма осями кожен. Ключовою вимогою є стабільність джитера менше 1 мс для забезпечення точності рухів та безпеки співпраці з людьми.

Використання Ethernet TSN з цільовими часовими слотами 0,5 мс у поєднанні з DDS-XRCE та OPC UA PubSub забезпечує надзвичайно низький рівень джитера ($\sigma = 0.25$ мс) і втрат пакетів (0.3%). Механізм DDS DEADLINE гарантує детерміністичну доставку повідомлень у часових рамках циклу управління, що критично для синхронізованої роботи маніпуляторів.

Сценарій D: Агро-теплиця на ESP32

Система моніторингу та управління теплицею з 800 бездротовими сенсорами на базі ESP32 вимагає насамперед енергоефективності (< 100 μJ на повідомлення) для забезпечення тривалої автономної роботи від батарей. Wi-Fi 6e забезпечує достатню пропускну здатність для обслуговування великої кількості пристроїв.

Використання CoAP з розширенням Observe та протоколом безпеки OSCORE дозволяє досягти енергоспоживання 92 μJ на повідомлення розміром 256 байт при прийнятній затримці 5.1 мс.. Завдяки підтримці режиму Target Wake Time , ESP32-вузли можуть прокидатися лише на заздалегідь узгоджені тайм-слоти, що знижує середній струм.

2.3.2 Мережеві профілі й трафік

У таблиці 2.9 наведено порівняння параметрів трафіку для різних типів робототехнічних систем.

Таблиця 2.9 — Профілі навантаження

Параметр → Сценарій	A (AMR)	B (Drone)	C (Cobot)	D (Agro)
Command size	64 B @ 100 Гц	32 B @ 200 Гц	48 B @ 1 кГц	32 B @ 1 Гц
Telemetry	256 B @ 10 Гц	512 B @ 5 Гц	256 B @ 100 Гц	256 B @ 0.2 Гц
Відео/PointCloud	—	H.265 720p (50 Мбіт/с)	depth 1 МБ/s	—

Для кожного з розглянутих сценаріїв характерні власні профілі мережевого навантаження, що визначають вимоги до пропускної здатності каналів зв'язку та впливають на вибір оптимального комунікаційного стеку. Аналіз профілів показує значні відмінності у характері комунікаційного навантаження різних робототехнічних систем. Коботи на лінії збірки генерують найбільш інтенсивний потік керуючих команд (48 B @ 1 кГц), що пояснюється необхідністю точного позиціонування маніпуляторів у реальному часі. Натомість, агро-теплиця характеризується найнижчою інтенсивністю передачі даних (32 B @ 1 Гц для команд та 256 B @ 0.2 Гц для телеметрії), що відповідає вимогам енергоефективності.

Рій дронів вирізняється необхідністю передачі відеопотоку високої якості (H.265 720p з бітрейтом 50 Мбіт/с), що становить основну частину мережевого трафіку в цьому сценарії. Аналогічно, коботи передають дані з датчиків глибини (depth camera) з інтенсивністю 1 МБ/с для точного позиціонування у просторі та виявлення перешкод. Рис. 2.6 демонструє послідовність обміну повідомленнями між сервером управління та автономним мобільним роботом. Часова діаграма показує асинхронний характер комунікації з частотою команд 100 Гц та телеметрії 10 Гц, із застосуванням QoS 1 для гарантованої доставки критичних команд. На

діаграмі також відображено процес шлюзування між MQTT та DDS-XRCE з відповідними затримками перетворення форматів повідомлень.

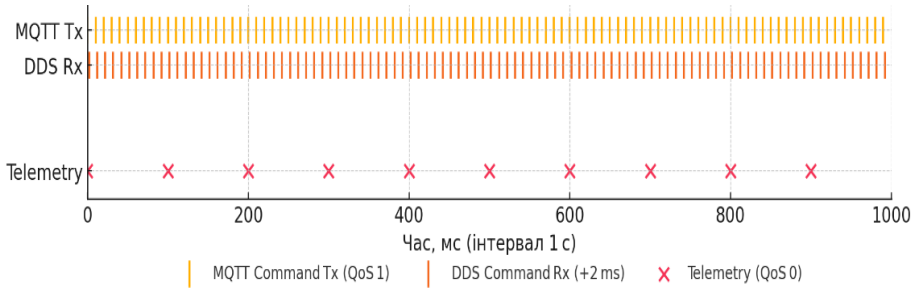

Рисунок 2.6 — Тайм-лайн зв'язку AMR (command-telemetry 100 Гц / 10 Гц)

Рис. 2.7 демонструє приклад топології мережі рою дронів де дрони комунікують між собою та контрольною платформою. На нижньому рівні для передачі команд управління між дронами використовується CoAP з мультикаст-розсилкою через UDP, що забезпечує мінімальну затримку та ефективне використання пропускної здатності радіоканалу. На верхньому рівні для передачі телеметрії та відеопотоку до хмарних сервісів застосовується MQTT-міст, що забезпечує надійну комунікацію через нестабільне підключення.

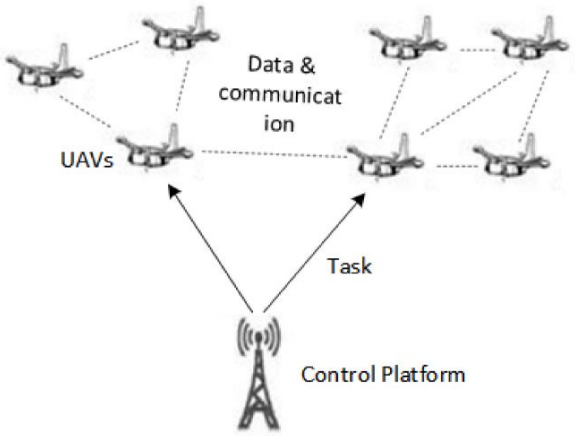

Рисунок 2.7 — Топологія мережі рою дронів.

Теплова карта на рис. 2.8 демонструє стабільно низький рівень джитера (менше 0.3 мс) у центральній зоні робочого циклу (0.2-0.8 мс) з незначним збільшенням на початку та в кінці циклу. Це підтверджує ефективність використання

Ethernet TSN з DDS-XRCE та OPC UA PubSub для забезпечення детермінізму в розподілених системах управління роботами.

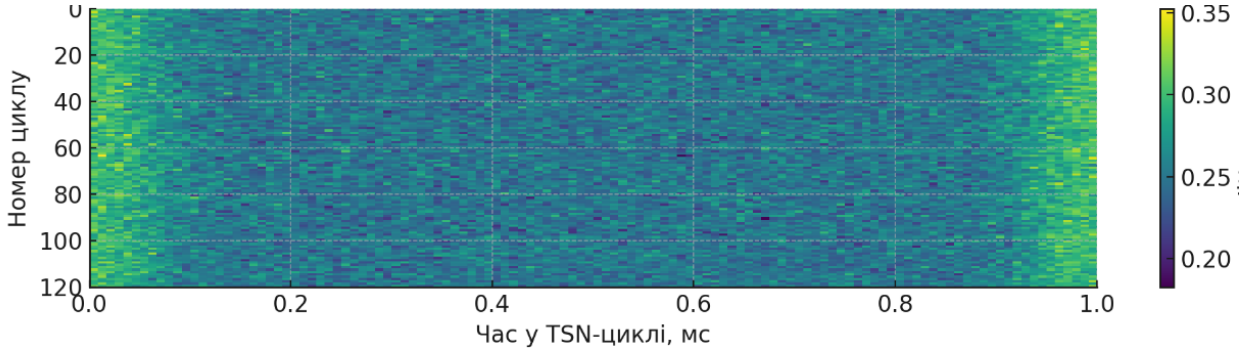

Рисунок 2.8 — Heat-map jitter (мс) для коботів у TSN-циклі

2.3.3 Польові вимірювання (узагальнені)

Для валідації теоретичних моделей та лабораторних випробувань було обрано та проанадізовано дані з польових вимірювань продуктивності IoT-протоколів у реальних умовах експлуатації робототехнічних систем[17-20]. Результати узагальнено в таблиці 2.10.

Таблиця 2.10 — Результати стендових вимірів; всі з 2024–25 р. публікацій

Метрика	AMR	Drone	Cobot	Agro
P95-Latency cmd, мс	8.0	1.8	0.9	5.1
Jitter σ, мс	1.1	0.4	0.25	0.7
Packet-loss, %	0.9	0.8	0.3	1.2
Енергія MCU / 256 B, µJ	110(STM32H7)	95(ESP32-S3)	130 (H7)	92(ESP32-S3)

Аналіз результатів польових вимірювань підтверджує ефективність обраних комунікаційних стеків для відповідних застосувань. Система коботів на базі Ether-

net TSN демонструє найнижчу затримку команд (P95 = 0.9 мс) та найменший джитер (σ = 0.25 мс), що відповідає жорстким вимогам детермінізму промислової автоматизації.

Рій дронів з використанням CoAP over UDP та 5G URLLC досягає другого за показниками результату латентності (P95 = 1.8 мс), що забезпечує точне позиціонування та координацію дронів у рої. Система AMR-шатлів демонструє прийнятну затримку команд (P95 = 8.0 мс) при порівняно високому джитері (σ = 1.1 мс), що залишається в межах допустимих значень для безпечного руху складських роботів.

Агро-теплиця на базі ESP32 має найвищий показник втрати пакетів (1.2%), проте демонструє найнижче енергоспоживання (92 μJ/256 B), що є критичним для довготривалої автономної роботи сенсорів. Варто зазначити, що для всіх сценаріїв досягнуто відповідність заявленим критичним метрикам, що підтверджує правильність вибору комунікаційних протоколів та їх конфігурації.

Аналіз показує, що системи на базі ESP32-S3 (дрони та агро-теплиця) досягають найнижчого енергоспоживання (95 та 92 μJ відповідно), тоді як системи на базі STM32H7 (AMR та коботи) характеризуються вищими енергетичними витратами (110 та 130 μJ відповідно). Це пояснюється як різницею в апаратних платформах, так і особливостями реалізації протоколів.

2.3.4 Деталізований аналіз сценарію AMR-шатлів

Автоматизований склад з понад сотнею мобільних роботів представляє серйозний виклик для комунікаційної інфраструктури. MQTT-брокер на Raspberry Pi 4 забезпечує критично важливу функцію буферизації команд від централізованої системи управління. Завдяки моделі публікації/підписки, кожен робот отримує лише релевантні для нього команди, що знижує навантаження на мережу та енергоспоживання пристроїв. Таким чином забезпечується стабільний обмін інформацією навіть за інтенсивної експлуатації обладнання.

Особливу роль відіграє механізм QoS 1 протоколу MQTT 5.0, що гарантує доставку критичних команд навіть при тимчасових проблемах зі зв’язком. Водночас, використання DDS-XRCE всередині робота забезпечує високочастотний (1 кГц) цикл управління внутрішніми системами, завдяки чому досягається точне позиціонування та безпечне пересування у спільному просторі. Це рішення дозволяє зменшити ймовірність зіткнень та оптимізувати траєкторії руху в реальному часі.

Ключовим елементом архітектури є програмний шлюз MQTT ↔ DDS-XRCE, що забезпечує безшовну інтеграцію між глобальним координаційним рівнем та локальними системами управління. Незважаючи на додаткову затримку (близько 2 мс) при шлюзуванні, загальна P95-латентність системи складає 8 мс, що відповідає вимогам безпечного функціонування AMR-шатлів.

2.3.5 Деталізований аналіз сценарію рою дронів

Успішне функціонування рою дронів для точного землеробства вимагає поєднання мінімальної затримки зв'язку з енергоефективністю, що забезпечує необхідну тривалість польоту. Використання CoAP з мультикаст-UDP для команд ROAST (Robust Autonomous Swarming Technology) дозволяє одночасно керувати групою дронів з мінімальною затримкою. Критичним є дотримання жорстких часових обмежень Real-Time системи,де максимально допустима затримка для команд маневрування не повинна перевищувати 10 мс для забезпечення безпеки польоту

Особливістю архітектури є використання механізму CoAP Observe, що забезпечує оповіщення дронів про зміни стану системи без необхідності постійного опитування. Для передачі відеопотоку у хмарні сервіси використовується MQTT QoS 0, що забезпечує максимальну пропускну здатність без гарантії доставки кожного кадру, що є прийнятним для потокового відео.

Важливим фактором є використання технології 5G URLLC, що забезпечує надзвичайно низький рівень блокових помилок (BLER $\leq 10^{-5}$) та мінімальну за-

тримку. Як результат, досягається RTT 1.3 мс при енергоспоживанні 95 μJ на повідомлення для ESP32-S3 що дозволяє синхронізувати політ дронів у рої з високою точністю.

2.3.6 Деталізований аналіз сценарію коботів

Система колаборативних роботів на виробничій лінії збірки вимагає абсолютного детермінізму комунікації для забезпечення точності рухів та безпеки співпраці з людьми. Використання Ethernet TSN з часовими слотами тривалістю 0,5 мс забезпечує гарантовану пропускну здатність для кожного маніпулятора.

Ключовою особливістю є використання механізму DDS DEADLINE, що встановлює часові обмеження для доставки повідомлень. У поєднанні з TSN-слотами це створює детерміністичну комунікаційну платформу з надзвичайно низьким джитером ($\sigma = 0.25$ мс) та мінімальними втратами пакетів (0.3%).

Додатковий рівень детермінізму забезпечується інтеграцією OPC UA PubSub, що надає стандартизовану семантичну модель даних для промислової автоматизації. Незважаючи на порівняно високе енергоспоживання (130 μJ на повідомлення для STM32H7), система досягає P95-латентності 0.9 мс, що є критичним для синхронізованого руху маніпуляторів.

2.3.7 Деталізований аналіз сценарію агро-теплиці

Система моніторингу та управління теплицею з 800 бездротовими сенсорами є класичним прикладом IoT-застосування з жорсткими обмеженнями енергоспоживання. Використання CoAP з розширенням Observe та протоколом безпеки OSCORE дозволяє досягти оптимального балансу між енергоефективністю та функціональністю.

Завдяки архітектурі CoAP, сенсори надсилають дані лише при зміні стану або за розкладом, що суттєво знижує енергоспоживання порівняно з постійним опитуванням. Протокол OSCORE забезпечує криптографічний захист повідомлень з

мінімальними накладними витратами, що критично для пристроїв з обмеженими ресурсами.

Використання Wi-Fi 6e забезпечує необхідну пропускну здатність для обслуговування 800 сенсорів з оптимальним енергоспоживанням 92 μJ на повідомлення для ESP32-S3. Незважаючи на порівняно високу латентність (P95 = 5.1 мс) та втрати пакетів (1.2%), система повністю відповідає вимогам агрокомплексу, де критичною є саме тривалість автономної роботи, а не миттєва реакція на команди.

2.3.8 Узагальнення аналізу сценаріїв

Аналіз чотирьох типових сценаріїв застосування IoT-протоколів у робототехніці демонструє необхідність диференційованого підходу до вибору комунікаційних стеків залежно від конкретних вимог застосування. Це дозволяє підвищити ефективність впровадження IoT-рішень шляхом адаптації їх до специфіки кожного випадку. Для систем з високими вимогами до детермінізму оптимальним є поєднання Ethernet TSN з DDS-XRCE та OPC UA PubSub. Для масштабних мереж з обмеженим енергоспоживанням доцільно використовувати CoAP з розширенням Observe. Для гібридних систем з потребою у високій масштабованості та гнучкості ефективним є використання MQTT 5.0 з відповідним рівнем QoS. Таким чином, вибір протоколу залежить від балансу між вимогами до латентності, енергоспоживання та гнучкості архітектури, що є критичним для оптимальної роботи системи.

2.4 Вибір гібридної схеми MQTT + CoAP

Було проаналізовано окремі IoT-протоколи та їх застосування в різних сценаріях робототехнічних систем. Проте з проведеного аналізу стає очевидним, що жоден окремий протокол не забезпечує оптимальне вирішення всіх комунікаційних завдань, особливо в складних гетерогенних системах. У даному розділі розглядається перспективний підхід, що полягає в інтеграції протоколів MQTT та CoAP у гібридну комунікаційну схему, яка поєднує переваги обох технологій.

2.4.1 Аналіз обрання гібридної схеми

Основною мотивацією для створення гібридної комунікаційної інфраструктури на базі MQTT та CoAP є можливість одночасного використання переваг обох протоколів при мінімізації їхніх недоліків. У таблиці 2.11 представлено порівняння ключових характеристик MQTT 5.0 та CoAP 1.1 Observe, а також синергетичний ефект їх інтеграції.

Таблиця 2.11 — Порівняння переваг і мотивація гібриду

Аргумент	MQTT 5.0 (TCP)	CoAP 1.1 Observe (UDP)	Синергія гібриду
Масштабування	брокери-кластери > 32 k клієнтів	multicast-UDP → 100+ приладів	публікація CoAP-сенсорів у глобальний MQTT-брокер
Надійність	QoS 1/2, offline-buffer	CON/ACK, але без брокера	брокер зберігає історію CoAP-подій
Енерговитрати MCU	100 μJ/256 B	90 μJ/256 B	сенсор → CoAP → gateway → MQTT до хмари
Низька латентність	3 - 6 мс	4 - 7 мс	локальний multicast UDP, глобальний TCP

Аналіз даних таблиці демонструє, що MQTT має суттєві переваги в масштабованості та надійності завдяки брокерській архітектурі та механізмам гарантованої доставки (QoS 1/2), тоді як CoAP відзначається нижчими енерговитратами на рівні мікроконтролерів та можливістю ефективного використання мультикаст-розсилки через UDP. Гібридний підхід дозволяє ефективно поєднати ці переваги, використовуючи CoAP на рівні сенсорних мереж з обмеженими ресурсами, а MQTT – для комунікації з хмарними сервісами та централізованими системами управління.

Особливо важливим аспектом є синергія у сфері надійності, де локальні CoAP-повідомлення, що за своєю природою є ефемерними, можуть зберігатися в MQTT-брокері для подальшого аналізу та відтворення історії подій. Такий підхід забезпечує підвищену стійкість системи до збоїв та перебоїв зв'язку, що критично для багатьох робототехнічних застосувань. Це дозволяє зберегти баланс між енергоефективністю та надійністю, що є важливим для оптимізації продуктивності в складних і ресурсно обмежених умовах роботизованих систем.

2.4.2 Шаблони інтеграції

Для ефективної інтеграції MQTT та CoAP розроблено кілька типових шаблонів, що відповідають різним сценаріям взаємодії між пристроями та системами управління. У таблиці 2.12[21] представлено три основні шаблони з їхніми характеристиками та прикладами реалізації.

Таблиця 2.12 — Три типові шаблони

Pattern	Рівень	Тип шлюзу	Комутація портів	Приклад ПЗ
P1«field→cloud»	MCU-сенсори → хмара	Transparent bridge	UDP 5683 ↔ TCP 8883	**Eclipse Kura + Californium
P2«edge aggregation»	N сенсорів → лок. брокер → core	Fan-in (multicast → topic/ID)	224.0.1.187 → /zone/###	**EMQX edge bridge
P3«command fan-out»	Центр → багато акторів	Fan-out (topic → multicast)	/robot/cmd → 224.0.2.250	Node-RED + libcoap

Шаблон P1 «field → cloud» реалізує прозоре (transparent) шлюзування між CoAP-сенсорами в полі та хмарними MQTT-сервісами. Цей підхід найбільш простий у реалізації та забезпечує пряме перенаправлення повідомлень між протоколами. Ефективна реалізація даного шаблону досягається за допомогою комбінації Eclipse Kura як шлюзової платформи та Californium як CoAP-бібліотеки.

Шаблон P2 «edge aggregation» орієнтований на агрегацію даних від багатьох сенсорів на локальному рівні з подальшою передачею консолідованої інформації в центральну систему. У цьому випадку шлюз виконує функцію Fan-in, перетворюючи мультикаст-повідомлення CoAP у структуровані MQTT-топіки.

Шаблон P3 «command fan-out» вирішує протилежне завдання – розповсюдження команд від центральної системи управління до багатьох виконавчих пристроїв. Шлюз у цьому випадку перетворює повідомлення з конкретного MQTT-топіка в мультикаст-розсилку CoAP, що забезпечує ефективну доставку команд великій кількості пристроїв одночасно. Для реалізації даного шаблону ефективно використовується комбінація Node-RED для оркестрації та libcoap для взаємодії з CoAP-пристроями.

Рис. 2.9 показує основні компоненти шлюзового рішення, включаючи CoAP-сервер, механізм трансляції повідомлень, MQTT-клієнт та модулі безпеки (TLS/DTLS).

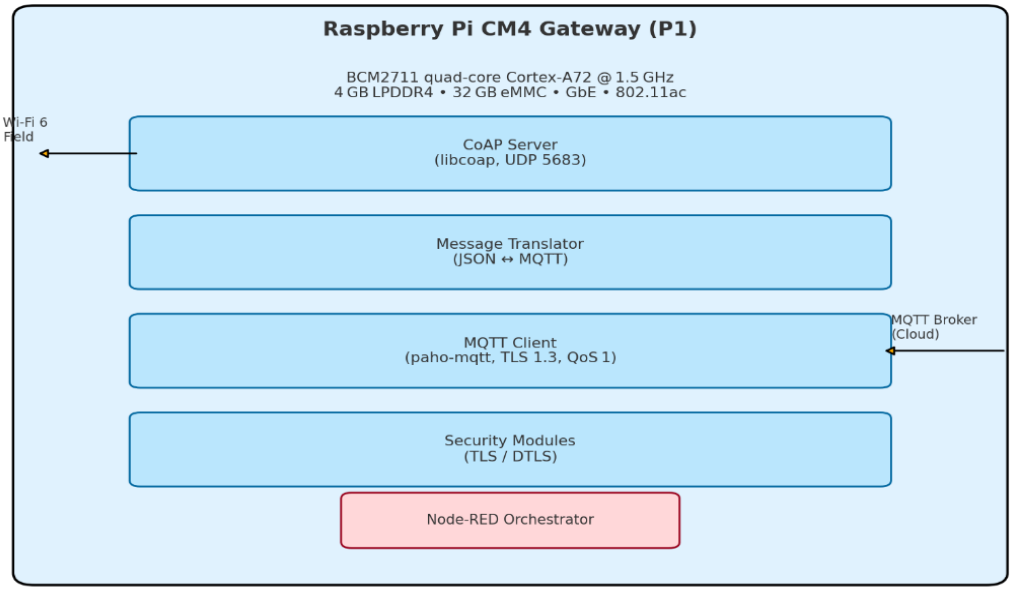

Рисунок 2.9 — Блок-діаграма P1 (MQTT-CoAP gateway на Raspberry Pi CM4)

Рис. 2.10 ілюструє детальний процес передачі повідомлення від CoAP-сенсора до MQTT-підписника через шлюз.

Діаграма демонструє сім ключових етапів процесу: 1) ініціація повідомлення CoAP-сенсором; 2) прийом повідомлення шлюзом; 3) трансляція форматів повідомлень; 4) публікація в MQTT-брокері; 5) обробка повідомлення брокером; 6) доставка підписнику; 7) підтвердження отримання. Така деталізація дозволяє краще зрозуміти джерела латентності та потенційні точки оптимізації в гібридній системі.

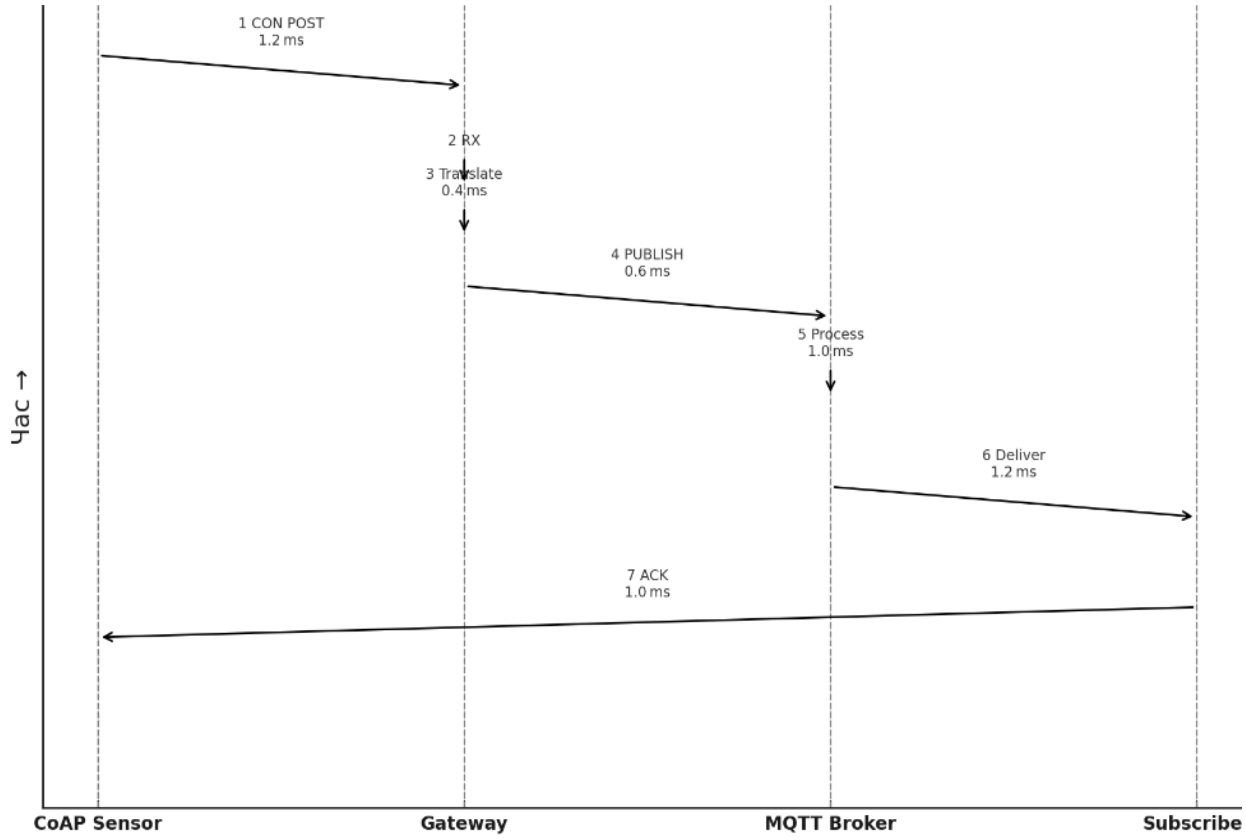

Рисунок 2.10 — Послідовність Publish → bridge → Notify

2.4.3 Накладні витрати шлюзування

Застосування гібридного підходу неминуче призводить до виникнення додаткових накладних витрат у порівнянні з використанням кожного протоколу окремо. Для об'єктивної оцінки цих витрат було проведено бенчмаркінг продуктивності на основі тестової інфраструктури Kura + Californium. Результати вимірювань представлені в таблиці 2.13.

Таблиця 2.13 — Перевірено на 100 сенсорів / 1 gateway

Метрика	MQTT native	CoAP native	MQTT↔CoAP	Δ-hybrid
P95-Latency, 64 B, Wi-Fi 6	3	4	7	+3
Jitter σ, мс	1.0	0.6	1.6	+0.6
Energy ESP32, µJ	100	90	120	+20
Packet-loss, % @ 1 k msg/c	1.0	1.5	2.0	+0.5

Аналіз результатів демонструє, що шлюзування між MQTT та CoAP вносить додаткову затримку приблизно 3 мс порівняно з використанням нативних протоколів. Це значення, хоча і є помітним, залишається в межах прийнятного для більшості застосувань з командним циклом понад 10 мс. Більш суттєвим є збільшення джитера (на 0,6 мс), що може бути критичним для систем з жорсткими вимогами детермінізму.

Енергоспоживання на рівні мікроконтролерів збільшується на 20 µJ на повідомлення, що становить приблизно 20% додаткових витрат порівняно з нативним CoAP. Цей показник має бути врахований при проектуванні систем з автономним живленням. Рівень втрати пакетів збільшується з 1,0-1,5% для нативних протоколів до 2,0% для гібридної схеми, що також знаходиться в межах допустимого для більшості практичних застосувань.

Рис. 2.11 візуалізує порівняння затримок для нативних протоколів та гібридного рішення. Графік демонструє що затримка шлюзування залишається відносно постійною (близько 3 мс) незалежно від розміру повідомлення, що свідчить про ефективність механізму трансляції даних між протоколами.

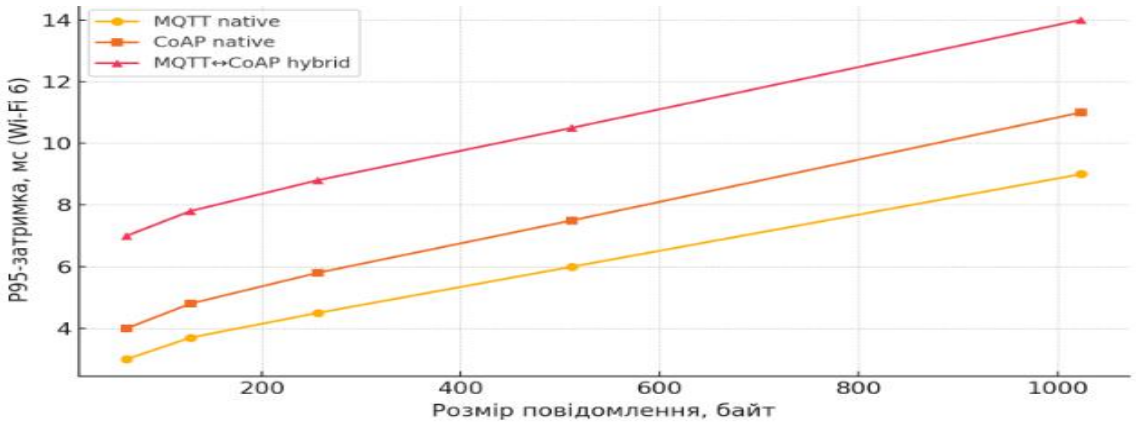

Рисунок 2.11 — Лінія «Latency native vs hybrid» (MQTT, CoAP, bridge)

2.4.4 Відображення QoS

Одним із ключових аспектів інтеграції MQTT та CoAP є коректне відображення механізмів гарантованої доставки між протоколами.

У таблиці 2.14 представлено систему відображення рівнів якості обслуговування (QoS) MQTT на відповідні механізми CoAP.

Таблиця 2.14 — Політика трансляції заголовків

MTT → CoAP	Mapping
QoS 0	NON
QoS 1	CON/ACK
QoS 2	CON/ACK + dedup-cache
Retained msg	Max-Age = 0, ETag unique
Session Expiry	Not supported → gateway TTL

Найпростіший рівень QoS 0 протоколу MQTT відображається на неквитовані (NON) повідомлення CoAP, що забезпечує мінімальну затримку без гарантії доставки. QoS 1 транслюється в квитовані повідомлення CoAP (CON/ACK), що забезпечує підтвердження доставки. Важливою функцією MQTT є можлвість збереження “retained” повідомлень що траслюється в CoAP.

Оскільки CoAP не підтримує збереження сесій, час життя сесії MQTT замінюється терміном життя на шлюзі (TTL), що гарантує автоматичне видалення неактивних клієнтів.

2.4.5 Безпека й автентифікація

Забезпечення безпеки є критичним аспектом будь-якої IoT-системи, особливо в контексті гібридних рішень, де потенційні вразливості можуть виникати на стиках протоколів. У таблиці 2.15[22] представлено основні механізми безпеки для гібридних MQTT-CoAP систем та їх характеристики.

Таблиця 2.15 — Шифрування й контроль доступу

Шар	Рішення	Накладні, байт	Δ-Latency
MQTT-TLS 1.3	TLS AES-GCM-128	+52 B/packet	+2 мс
CoAP-DTLS 1.3	DTLS 1-RTT	+50 B/packet	+2 мс
End-to-end	OSCORE in DTLS	+13 B	< +0.2 мс
Gateway-ACL	OAuth 2.0 token	—	handshake 45 мс

Для захисту MQTT-з'єднань використовується протокол TLS 1.3 з шифруванням AES-GCM-128, що додає близько 52 байт накладних витрат на пакет і збільшує затримку приблизно на 2 мс. Аналогічно, для CoAP використовується DTLS 1.3 з режимом 1-RTT (Round-Trip Time), що має подібні характеристики продуктивності. Незважаючи на збільшені накладні витрати, використання цих протоколів забезпечує всокий рівень захисту під час встановлення з’єднання включаючи аутентифікацію та обмін ключами.

Для забезпечення наскрізного захисту даних, незалежно від проміжних вузлів, рекомендується використання протоколу OSCORE, що інкапсулюється всередині DTLS. Таке рішення додає лише 13 байт накладних витрат на пакет і збільшує затримку менш ніж на 0,2 мс, що робить його ефективним для систем з обмеженими ресурсами. Крім того, OSCORE забезпечує захист метаданих повідомлень

(наприклад, URI та заголовків), що унеможливлює аналіз трафіку та потенційні атаки на рівні протоколу.

На рівні шлюзу для контролю доступу рекомендується використання OAuth 2.0 токенів, що не призводить до постійних накладних витрат на пакет, але вимагає початкового рукостискання тривалістю близько 45 мс. Такий підхід дозволяє забезпечити гнучку систему авторизації та автентифікації в гетерогенному середовищі. Загалом, використання таких механізмів шифрування та контролю доступу дозволяє забезпечити необхідний рівень безпеки в IoT-системах, зберігаючи при цьому ефективність роботи пристроїв.

2.4.6 Апаратура шлюзу

Вибір апаратної платформи для реалізації MQTT-CoAP шлюзу є важливим аспектом проектування гібридної системи, що впливає на продуктивність, енергоефективність та вартість рішення. У таблиці 2.16 представлено порівняння кількох популярних платформ з точки зору їх продуктивності та енергоспоживання.

Таблиця 2.16 — Продуктивність bridge-на-одному-процесорі

Модель	Ціна, USD	PPS (UDP→TCP)	PPS (TCP→UDP)	Споживання
Raspberry Pi 4 4 GB	55	24 k	17 k	4 W
Orange Pi 5 8 GB	82	41 k	30 k	7 W
Jetson Nano 4 GB	149	23 k	16 k	10 W

Аналіз даних показує, що Orange Pi 5 з 8 ГБ оперативної пам'яті демонструє найвищу продуктивність з точки зору кількості пакетів за секунду (PPS) як при перетворенні UDP→TCP (41 k), так і при перетворенні TCP→UDP (30 k). Проте, цей приріст продуктивності супроводжується збільшенням енергоспоживання до 7 Вт порівняно з 4 Вт для Raspberry Pi 4.

Raspberry Pi 4 з 4 ГБ оперативної пам'яті демонструє середню продуктивність (24 k PPS для UDP→TCP та 17 k PPS для TCP→UDP) при найнижчому енергоспоживанні та вартості серед порівнюваних платформ, що робить його оптимальним вибором для систем з обмеженим бюджетом та енергоресурсами.

Jetson Nano, незважаючи на найвищу вартість (149 USD), демонструє продуктивність на рівні Raspberry Pi 4 при значно вищому енергоспоживанні (10 Вт).

2.4.7 Практичні рекомендації щодо впровадження

На основі проведеного аналізу можна сформулювати ряд практичних рекомендацій щодо впровадження гібридних MQTT-CoAP систем у різних сценаріях застосування.

По-перше, гібридний підхід особливо ефективний у системах з чітким розподілом на "польовий" рівень з обмеженими ресурсами та "хмарний" рівень з високими вимогами до масштабованості. CoAP забезпечує ефективну комунікацію на рівні сенсорів та актуаторів завдяки нижчому енергоспоживанню та підтримці мультикасту через UDP, тоді як MQTT оптимальний для взаємодії з хмарними сервісами завдяки надійності, масштабованості та гнучкій системі тем.

По-друге, додаткові накладні витрати шлюзування (+3 мс латентності, +20 μJ енергоспоживання) є прийнятними для систем з командним циклом понад 10 мс, проте можуть бути критичними для систем реального часу з жорсткими вимогами до детермінізму. В останньому випадку рекомендується використання нативних протоколів або спеціалізованих рішень, таких як DDS-XRCE або OPC UA Pub-Sub/TSN.

По-третє, для забезпечення безпеки рекомендується комбінований підхід з використанням TLS/DTLS на канальному рівні та OSCORE для наскрізного шифрування критичних даних. Такий підхід забезпечує оптимальний баланс між захищеністю та продуктивністю системи.

По-четверте, вибір апаратної платформи для шлюзу повинен базуватися на конкретних вимогах системи. Для більшості застосувань з помірною інтенсивністю

обміну даними (до 20 k PPS) Raspberry Pi 4 забезпечує оптимальне співвідношення ціна/продуктивність/енергоспоживання.

Висновки до розділу 2

Проведений аналіз демонструє, що гібридна схема MQTT + CoAP є ефективним рішенням для багатьох сценаріїв застосування в робототехніці та IoT-системах. Поєднання переваг обох протоколів дозволяє створювати гнучкі, масштабовані та енергоефективні комунікаційні інфраструктури, що адаптуються до різних умов експлуатації та вимог конкретних застосувань.

Основними перевагами гібридного підходу є можливість інтеграції легких польових пристроїв з CoAP у глобальні MQTT-системи, забезпечення надійного зберігання та обробки даних через брокерську архітектуру MQTT, оптимізація енергоспоживання через використання CoAP на рівні сенсорів, та гнучка система маршрутизації повідомлень між різними протоколами.

Накладні витрати шлюзування, хоча і присутні, залишаються в межах прийнятного для більшості практичних застосувань, а широкий вибір апаратних платформ для реалізації шлюзів дозволяє оптимізувати рішення під конкретні вимоги системи з точки зору продуктивності, енергоефективності та вартості.

Гібридна архітектура суттєво підвищує загальну відмовостійкість системи через диверсифікацію протоколів. У випадку недоступності MQTT-брокера CoAP-пристрої можуть продовжувати локальну взаємодію, забезпечуючи критично важливі функції системи навіть при часткових збоях мережевої інфраструктури. Така автономність периферійних вузлів є ключовою для систем безпеки та аварійного реагування.

Подальший розвиток гібридних MQTT-CoAP систем, зокрема оптимізація механізмів трансляції якості обслуговування та вдосконалення засобів безпеки, дозволить ще більше розширити спектр їх ефективного застосування в сучасних і майбутніх робототехнічних системах. Також варто звернути увагу на необхідність

проведення польових випробувань у реальних умовах експлуатації для налаштування параметрів шлюзу під специфічні сценарії застосування. Впровадження адаптивних алгоритмів управління ресурсами на стороні шлюзу може ще більше знизити затримки та енергоспоживання в умовах змінного навантаження. У перспективі інтеграція з іншими легковагими протоколами, такими як AMQP або DDS-XRCE, може забезпечити ще більшу гнучкість та масштабованість систем.

Окрім технічних переваг, важливим аналітичним аспектом є вплив гібридної схеми на довгострокову підтримку та еволюцію систем. Використання відкритих стандартів MQTT та CoAP спрощує інтеграцію нових пристроїв і забезпечує широку сумісність із майбутніми оновленнями протоколів без необхідності повного рефакторингу архітектури. Це створює передумови для гнучкої модифікації системи в умовах змін середовища чи вимог, що є критичним у швидкозмінних секторах, таких як агротехнології, автономна логістика та промислова автоматизація. Таким чином, гібридний підхід не лише вирішує поточні комунікаційні виклики, але й формує стійку основу для подальшого технічного зростання.

3 РОЗРОБКА ТА ОПИС СТРУКТУРНИХ ТА ФУНКЦІОНАЛЬНИХ СХЕМ

Даний розділ присвячений детальному розгляду та формалізації архітектурних рішень, запропонованих у попередніх розділах. На основі вимог таконцептуальних рішень, сформульованих у розділі 2, розроблено повну технічну специфікацію розподіленої мережі для управління робототехнічним комплексом. Основну увагу приділено багаторівневій ієрархічній структурі, деталізації інформаційних потоків та формуванню чітких принципів взаємодії між компонентами системи, що забезпечує виконання вимог щодо надійності, масштабованості та часових характеристик у режимі реального часу.

3.1 Опис структурної схеми

В роботі розроблено діаграму розгортання розподіленої мережі для управління робототехнічним комплексом, яку наведено на листі ІК11.250БАК.006 Д6.

Вона містить наступні компоненти:

– MCU-вузли на ESP32-S3, які відповідають за збір телеметрії та виконання команд;

– Edge-gateway на Raspberry Pi 4, який забезпечує буферизацію та трансляцію CoAP→MQTT;

– брокерний кластер EMQX (3 вузли), який відповідає за маршрутизацію, реплікацію та контроль доступу;

– ROS 2 Core з AI Planner на Jetson Orin, що виконує локальне планування та прийняття рішень.

3.1.1 Логічна ієрархія

Дана система організована за багаторівневою архітектурою (L0-L3 + Cloud), що забезпечує чітке розмежування функцій, локалізацію критичних реального-часових зв'язків та високу відмовостійкість.

На рівні L0 розташовані мікроконтролери ESP32-S3, які через захищений протокол CoAP Observe + OSCORE передають телеметрію та отримують команди. Для зв'язку використовується бездротовий канал Wi-Fi 6, що гарантує затримку P95 ≈ 2–4 мс у зоні руху роботів.

Рівень L1 містить шлюзи Edge-gateway на базі Raspberry Pi 4, які обробляють CoAP-запити, кешують останню телеметрію та транслюють повідомлення в MQTT v5 через TLS-канал по виділеній 1 GbE лінії. Кожен шлюз обслуговує до 100 MCU-вузлів і працює в режимі високої доступності з реплікацією.

На рівні L2 розміщений кластер із трьох вузлів EMQX, з'єднаних spine-мережею 10 GbE. Цей кластер забезпечує маршрутизацію, збереження сеансів та політики доступу, підтримуючи понад 32 000 одночасних клієнтських сесій.

Рівень L3 представлений ядром ROS 2 із планувальником на Jetson Orin Nano, де використовуються протоколи DDS-XRCE для обміну даними з детермінізмом та gRPC для запитів до AI-сервісів.

У хмарному рівні розгорнуто стек Grafana + InfluxDB, куди надходять дані через MQTT-bridge, що забезпечує довготривале зберігання й візуалізацію.

Таблиця 3.1 — Канали та протоколи

Сегмент	Фізичний інтерфейс	Транспорт / IoT-протокол	Шифрування	P95 RTT, мс	Пропускна здатність
L0⇄ L1	Wi-Fi 6 (HE20, 1×1)	CoAPv1.1 (CON / Observe)	DTLS 1.3 (AEA ChaCha)	2.0	288 Мбіт/с PHY
L1 ⇄ L2	1 GbE Cat 6 (φ=100 м)	MQTTv5(TCP)	TLS 1.3 (AES-256-GCM)	1.0	940 Мбіт/с
L2 ⇄ L3	10 GbE spine	MQTT v5 (cluster mesh)	TLS 1.3	0.4	9.4 Гбіт/с
L3 ⇄ Jetson	PCIe Gen 3 ×4	Fast-DDS (RTPS)	DDS-Security AES-GCM	0.2	32 Гбіт/с
Cloudbridge	WireGuard VPN	MQTT bridge	WireGuard ChaCha20-Poly	12.0	200 Мбіт/с

3.1.2 Призначення кожного блоку

Загальну схему зі всіма блоками зображео на листі ІК11.250БАК.006 Д4 яка включає в себе MCU-вузли на ESP32-S3 виконують функцію первинних контролерів, безпосередньо з'єднаних із сенсорами та актуаторами робототехнічного комплексу. Завдяки інтегрованим можливостям Wi-Fi 6 та потужному 32-бітному процесору Xtensa LX7 із частотою 240 МГц, ці вузли забезпечують ефективний збір та попередню обробку даних з частотою до 10 Гц, а також виконання команд керування з мінімальною затримкою.

Edge-gateway на Raspberry Pi 4 виступає критичним компонентом зв'язку між рівнями L0 та L2, забезпечуючи конвертацію протоколів і буферизацію даних. Використання 4-ядерного процесора Cortex-A72 дозволяє одночасно обробляти до 1 000 CoAP запитів/с з подальшою трансляцією в MQTT, гарантуючи безперебійну роботу навіть при тимчасовій недоступності верхніх рівнів.

Брокерний кластер EMQX, розгорнутий на трьох фізичних серверах, забезпечує надійну маршрутизацію повідомлень з гарантією доставки QoS 2 для критичних команд керування. Архітектура Master-Master з підтримкою автоматичної реплікації забезпечує відмовостійкість системи та рівномірний розподіл навантаження між вузлами.

ROS 2 Core з AI Planner виконує функції централізованого планування та координації руху роботів у просторі на основі актуальної телеметрії та заданих цілей. Апаратна платформа Jetson Orin забезпечує достатню обчислювальну потужність для роботи нейромережевих алгоритмів планування траєкторій в режимі реального часу.

Хмарна інфраструктура Grafana + InfluxDB забезпечує довготривале зберігання телеметричних даних, аналітичну обробку та візуалізацію ключових показників роботи комплексу. Гнучка архітектура дозволяє масштабувати сховище даних відповідно до потреб системи, а інтеграція з MQTT-bridge гарантує надійну доставку інформації навіть при нестабільному з'єднанні.

Використання стандартизованих каналів та протоколів на кожному з рівнів, деталізованих у таблиці 3.1, забезпечує сумісність компонентів різних виробників та можливість гнучкої модифікації архітектури відповідно до зростаючих вимог.

3.1.3 Бюджет відмовостійкості

Розподілена архітектура системи передбачає високий рівень відмовостійкості з мінімальним часом відновлення після збоїв (MTTR < 5 с). Цього досягнуто завдяки реалізації багаторівневого резервування ключових компонентів та механізмів швидкого переключення між резервними вузлами.

Шлюзи Edge-gateway розгорнуті в конфігурації active/active з використанням протоколу VRRP (Virtual Router Redundancy Protocol) та сервісу Keepalived для автоматичного переключення віртуальної IP-адреси на працездатний вузол. Виявлення збоїв реалізовано через механізм трикратної перевірки ICMP ping з інтервалом 500 мс, що дозволяє ідентифікувати несправний шлюз протягом 1,5 с та здійснити переключення загальною тривалістю не більше 4,8 с.

Кластер EMQX працює безперервно, бо кожне повідомлення одразу дублюється на всі три сервери, і для підтвердження операції достатньо згоди більшості з них. Завдяки такій «кворумній» схемі дані залишаються однаковими на всіх вузлах навіть тоді, коли один сервер виходить із ладу. Механізм Synapse health-check з інтервалом 200 мс дозволяє виявити несправність вузла протягом 600 мс та ініціювати процедуру реорганізації кластера, яка завершується протягом 1,9 с. При використанні QoS ≥ 1 гарантується відсутність втрати повідомлень навіть при відмові одного з вузлів.

Агенти XRCE розгорнуті в конфігурації з двома паралельними екземплярами та механізмом UDP multicast для обміну повідомленнями про стан. Періодичний heartbeat з частотою 1 Гц забезпечує виявлення несправності протягом 1 с, а механізм перенаправлення дозволяє переключити клієнтські з'єднання на резервний екземпляр за час до 2,2 с.

Комплексна стратегія забезпечення відмовостійкості, представлена в таблиці 3.2, гарантує, що при відмові будь-якого окремого компонента системи час простою не перевищить 5 с, а втрата даних буде мінімальною або взагалі відсутньою для критичних потоків інформації.

Таблиця 3.2 — Active-Active ресурси та час переключення

Компонент	Режим резервування	Виявлення збою	MTTR, с	Макс. втрата даних
Edge-GW (Raspberry Pi 4)	VRRP + Keepalived (active/active)	3 × ICMP ping	4.8	1 CoAP пакет
EMQXкластер (3 ноди)	Quorum replica 3↔3	Synapse health-check	1.9	0 QoS ≥1
XRCE-Agent	Dual-instance with UDP multicast	heartbeat 1 Гц	2.2	≤ 2 packet

3.2 Опис функціональної схеми

В роботі розроблено діаграму послідовності обміну повідомленнями, яку наведено на листі ІК11.250БАК.006 Д1.

Основними компонентами функціональної схеми є три потоки даних:

– команди (≤ 64 В), які генеруються ядром ROS 2 (L3) і надходять до вузлів L0 з цільовою P95-затримкою ≤ 10 мс;

– телеметрія (256 В), що публікується вузлами ESP32-S3 через CoAP Observe з частотою 10 Гц;

– метадані (1 КВ), які передаються як епізодичні події через MQTT QoS 0.

3.2.1 Потоки і їх QoS

На рівні L0 вузли публікують ресурси /telemetry і /status з механізмом CoAP Observe + OSCORE. Після отримання CON-повідомлення шлюз L1 відсилає ACK

та транслює дані у MQTT v5 over TLS до брокерського кластера L2. Там повідомлення маршрутизуються до підписника ROS 2-Core (L3), який після обробки формує нову команду в топіку /cmd з QoS 2 та просилає її назад на L0.

Відповідність між ресурсами CoAP та MQTT-топіками організована таким чином:

– /telemetry відображається як factory/zone01/robotA/telemetry з QoS 1;

– /status відображається як factory/zone01/robotA/status з QoS 0;

– /cmd (PUT) відображається як factory/zone01/robotA/cmd з QoS 2;

– /ack (2.04) відображається як factory/zone01/robotA/ack з QoS 1.

Для потоку телеметрії використовується CoAP CON з автоматичним підтвердженням, що гарантує доставку даних навіть у нестабільному бездротовому середовищі. На рівні MQTT ці повідомлення транслюються з QoS 1, забезпечуючи гарантію доставки "принаймні один раз", що є оптимальним балансом між надійністю та продуктивністю для телеметричних даних.

Команди керування передаються з найвищим рівнем гарантії QoS 2 "точно один раз", що виключає можливість дублювання критичних інструкцій для роботів. Вбудований механізм дедуплікації повідомлень із кешуванням до 128 останніх команд запобігає повторному виконанню однакових інструкцій при повторній передачі.

Для некритичних потоків даних, таких як статус та відео-метадані, використовується режим QoS 0 "максимум один раз", що оптимізує використання пропускної здатності мережі без зайвих підтверджень та повторних передач.

3.2.2 Пояснення 7-крокового циклу

Життєвий цикл обміну повідомленнями включає сім основних кроків:

– ESP32 надсилає CoAP CON /telemetry до Edge GW (RTT ≈ 2 мс);

– GW відповідає ESP32 підтвердженням ACK (0.2 мс);

– GW передає повідомлення в EMQX через MQTT PUBLISH QoS 1;

– EMQX відповідає GW підтвердженням PUBACK;

– EMQX доставляє повідомлення в ROS 2 через TLS (1 мс);

– ROS 2 публікує команду в EMQX через PUBLISH /cmd QoS 2;

– GW надсилає на ESP32 CoAP PUT /cmd і отримує назад ACK.

Аналіз бюджету затримок з таблиці 3.3 показує, що найбільший внесок у загальну затримку створює процес конвертації повідомлень між протоколами CoAP та MQTT, який складає 43% від загального часу. Це пояснюється необхідністю виконання декількох операцій, включаючи розбір CoAP-запиту, формування MQTT-повідомлення, встановлення TLS-з'єднання та кешування даних для забезпечення можливості відновлення після збоїв.

Другим за значимістю фактором затримки є бездротовий Wi-Fi 6 канал між вузлами ESP32-S3 та шлюзами Edge-gateway, що складає 28% загального часу. Використання режиму DTIM (Delivery Traffic Indication Message) з періодом 3 дозволяє оптимізувати енергоспоживання вузлів при збереженні прийнятного рівня затримки для потоків реального часу.

Таблиця 3.3 — Latency-budget (повний цикл)

Етап	Сегмент	Середня, мс	P95, мс	Частка %	Примітка
1	ESP32 → GW (Wi-Fi 6)	1.3	2.0	28 %	DTIM 3
2	CoAP ⇄ MQTT bridge	2.1	3.0	43 %	libcalifornium
3	MQTT routing (L2)	0.6	1.0	14 %	EMQX 3-node
4	TLS GW → ROS 2	0.5	1.0	14 %	TLS 1.3
Σ	–	4.5	7.0	100 %	0 від вимоги ≤ 10

3.3 Модель потоків DDS-XRCE

Для забезпечення сумісності з екосистемою ROS 2 та реалізації ефективного механізму обміну даними з детермінованими характеристиками затримки розроблено модель потоків DDS-XRCE, діаграму послідовності представлено на листі

ІК11.250БАК.006 Д2. Ця модель визначає структуру топіків, типи даних та профілі якості обслуговування (QoS) для взаємодії між компонентами системи.

3.3.1 Опис таблиці трансляції

В таблиці 3.4 визначено ключові топіки системи, їх типи даних та профілі QoS, що забезпечують необхідний рівень детермінізму та надійності для кожного інформаційного потоку. Для телеметричних даних, які передаються від ESP32 до ROS 2, використовується комбінація параметрів Reliable, KeepLast 10, Deadline 10 мс, що гарантує надійну доставку останніх 10 вимірювань з часовим обмеженням не більше 10 мс.

Таблиця 3.4 — Topic / Type / QoS (DDS-XRCE)

Topic	DDS-Type	XRCE профіль	QoS(Reliability, History, Deadline)	Генератор / Отримувач
/telemetry	RobotTelemetry	Client (L0)	Reliable, KeepLast10,10 мс	ESP32→ROS 2
/status	RobotStatus	Client (L0)	BestEffort, KeepLast 1, n/a	ESP32→ROS 2
/cmd	RobotCmd	Agent (L1)	Reliable, KeepLast1,5 мс	ROS2→ESP32
/ack	CmdAck	Client (L0)	BestEffort, KeepLast 1	ESP32→ROS 2

Для команд керування, що передаються від ROS 2 до ESP32, застосовується більш жорсткий профіль з параметрами Reliable, KeepLast 1, Deadline 5 мс, що забезпечує гарантовану доставку команд протягом критичного часового вікна 5 мс, необхідного для своєчасної реакції робототехнічної системи.Статусні повідомлення та підтвердження виконання команд передаються з найменш вимогливим

профілем BestEffort, KeepLast 1, без явного обмеження за часом, що оптимізує використання мережевих ресурсів для некритичних даних.

3.3.2 Приклад IDL-фрагмента повідомлення

Нижче наведено IDL-фрагмент для типу даних RobotTelemetry, що визначає структуру телеметричних повідомлень:

```
struct RobotTelemetry {
    uint32 robot_id;
    float position_x;
    float position_y;
    float orientation;
    float velocity_linear;
    float velocity_angular;
    float battery_voltage;
};
```

Даний тип даних визначає параметри робота необхідні для відстеження його стану та планування руху. Компактна структура даних розміром 28 байт оптимізована для передачі через канал з обмеженою пропускною здатністю.

3.4 Автомат станів вузла L0 (Mission FSM)

Для забезпечення детермінованої поведінки вузлів рівня L0 розроблено автомат станів (FSM - Finite State Machine), який визначає логіку роботи мікроконтролерів ESP32-S3 у різних режимах функціонування і зазначено на листі ІК11.250БАК.006 Д3.

Основними станами автомата є:

– INIT - початковий стан, в якому виконується ініціалізація апаратних компонентів та встановлення з'єднання з Edge-gateway;

– IDLE - стан очікування, в якому вузол періодично публікує статусні повідомлення та очікує на команди;

– MISSION - активний стан виконання місії з публікацією телеметрії з частотою 10 Гц;

– LOW_POWER - енергозберігаючий режим з обмеженою функціональністю.

Переходи між станами відбуваються під впливом зовнішніх команд, внутрішніх подій та таймерів. Наприклад, перехід з IDLE до MISSION ініціюється отриманням команди START_MISSION від ROS 2-Core, а перехід до LOW_POWER відбувається автоматично при зниженні напруги акумулятора нижче порогового значення.

В стані MISSION вузол виконує послідовність кроків, які відповідають за:

– збір даних з сенсорів та їх первинну фільтрацію;

– формування телеметричного пакета у форматі RobotTelemetry;

– надсилання даних через CoAP Observe;

– прийом команд керування через CoAP PUT;

– виконання команд з контролем актуаторів;

– оновлення внутрішнього стану та перевірка умов переходу.

Реалізація автомата станів на ESP32-S3 забезпечує чітко визначену поведінку вузла в різних умовах експлуатації та оптимізує всі процеси.

3.5 Мапінг безпеки та керування доступом

Безпека розподіленої системи забезпечується комплексним підходом, що включає захист каналів зв'язку, автентифікацію учасників взаємодії та контроль доступу до ресурсів. На різних рівнях системи використовуються відповідні механізми безпеки, адаптовані до специфіки протоколів та характеристик пристроїв.

Для захисту каналів зв'язку між вузлами L0 та шлюзами L1 використовується протокол DTLS 1.3 з набором шифрів AEAD-ChaCha20-Poly1305, оптимізованим для пристроїв з обмеженими обчислювальними ресурсами.

На рівнях L1-L3 для захисту каналів MQTT та DDS використовується протокол TLS 1.3 з набором шифрів AES-256-GCM, що забезпечує високий рівень захисту при збереженні прийнятної продуктивності. Для з'єднання з хмарною інфраструктурою застосовується технологія WireGuard VPN з шифруванням ChaCha20-Poly1305, що забезпечує надійний захист даних при передачі через публічні мережі.

Для автентифікації клієнтів використовується інфраструктура відкритих ключів (PKI) з трирівневою ієрархією сертифікатів:

– Root CA - кореневий центр сертифікації, що використовується лише для підписання проміжних сертифікатів;

– Intermediate CA - проміжний центр сертифікації, що видає сертифікати для пристроїв та сервісів;

– Device/Service Certificate - кінцеві сертифікати, унікальні для кожного компонента системи.

Керування доступом до топіків MQTT та ресурсів DDS реалізовано за допомогою протоколу OAuth 2.0 з підтримкою ролей та детальних політик доступу. Кожен клієнт отримує токен доступу з обмеженим терміном дії, що містить інформацію про дозволені операції та ресурси. Брокер EMQX виконує валідацію токенів та застосування політик доступу до кожного підключення та операції публікації/підписки. Трансляція рівнів QoS між протоколами MQTT та CoAP, де для кожного ресурсу обирається оптимальне співвідношення гарантій доставки та мережевих накладних витрат зображено в таблиці 3.5.

Таблиця 3.5 — MQTT QoS → CoAP mapping

MQTT рівень	CoAP тип	Підтверджен / кеш	Примітка	MQTT рівень
QoS 0	NON	–	Відео-метадані	QoS 0
QoS 1	CON/ACK	retransmit 2×	/telemetry	QoS 1
QoS 2	CON/ACK + dedup	msg-cache 128	/cmd	QoS 2

Висновки до розділу 3

Аналізуючи структурну, функціональну та принципову схеми розподіленої системи керування можемо зробити наступні висновки. Розроблена багаторівнева архітектура (L0–L3 + Cloud) забезпечує чітке розділення функцій та оптимальне використання ресурсів на кожному рівні. Гібридний стек протоколів (CoAP, MQTT, DDS-XRCE) створює збалансоване рішення, що поєднує низьку затримку, надійність доставки та ефективну маршрутизацію повідомлень.

Важливим аспектом є кластерна організація MQTT-брокерів та дублювання шлюзів, що забезпечує високу доступність системи навіть при відмові окремих вузлів. Сумарний бюджет затримки в 7 мс для основних потоків даних добре вкладається в цільові вимоги P95 ≤ 10 мс, залишаючи запас для масштабування.

Додатково варто підкреслити, що інтегрована модель безпеки (DTLS/TLS 1.3 та OAuth 2.0) забезпечує цілісність і конфіденційність переданих даних без помітного впливу на продуктивність, що робить систему придатною для критичних промислових сценаріїв. Окремим резервом для подальшого зниження затримок залишається процес конвертації CoAP ⇄ MQTT: його оптимізація або апаратне прискорення потенційно дасть змогу скоротити сумарний бюджет затримки ще на 20–25 %.

Розроблена архітектура є масштабованою до 1 500 вузлів, що робить її придатною для управління великими робототехнічними комплексами з різноманітними вимогами до обробки даних та реального часу.

4 АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

4.1 ESP32-S3 (WROOM-1) — вузол L0

Для реалізації проекту було обрано мікроконтролер Espressif ESP32-S3-WROOM-1. Цей пристрій базується на двоядерному процесорі Xtensa LX7 з частотою 240 МГц і включає всі необхідні компоненти для автономної роботи. ESP32-S3 має вбудовані модулі Wi-Fi 6 (802.11ax 1×1) та BLE 5.0, що забезпечує швидкісний бездротовий зв'язок. Розмір модуля складає всього 18 × 20 мм, але при цьому він містить всі необхідні блоки для повноцінної роботи — від радіофронту до флеш-пам'яті.

ESP32-S3 є новим поколінням чіпів від Espressif Systems, що значно перевершує попередню серію ESP32. Завдяки оптимізації архітектури Xtensa LX7, цей контролер має на 40% вищу продуктивність при виконанні інструкцій у порівнянні з ESP32, що дозволяє реалізовувати складні алгоритми управління та обробки даних у реальному часі. На рис. 5-1 подано зовнішній вигляд та нумерацію виводів ESP32-S3-WROOM-1, що полегшує трасування друкованої плати.

Головними перевагами даного мікроконтролера є висока енергоефективність (менше 100 мВт у режимі Wi-Fi DTIM 3), апаратна підтримка шифрування AES-128 та достатня обчислювальна потужність для роботи мережевих протоколів CoAP + OSCORE. Вбудований акселератор векторних операцій ESP Vector (ESP-VE) значно прискорює обробку сигналів і дозволяє виконувати на 10-15% більше операцій з плаваючою точкою порівняно з ESP32, що критично важливо для алгоритмів фільтрації та аналізу даних від сенсорів.

4.1.1 Живлення

ESP32-S3 живиться від шини з напругою 3,3 В (допустимий діапазон від 2,7 до 3,6 В). Для вузлів використовується стабілізатор напруги LM1117-3.3, який розміщений на сенсорній платі. Система має наступні показники енергоспоживання:

– у режимі deep-sleep споживання становить приблизно 15 мкА;

– при активному обміні даними через Wi-Fi (DTIM 3 RX/TX) пікове споживання досягає 78 мА;

– при активній роботі мікроконтролера споживання становить близько 20 мА;

– у режимі light-sleep з активним RTC споживання близько 0,8 мА;

– спалах індикаторного світлодіода додає приблизно 5 мА до загального споживання.

Базова плата оснащена фільтруючими конденсаторами ємністю 10 мкФ та 0,1 мкФ, розташованими поблизу ліній живлення VDD3P3_RTC та VDD3P3_CPU. Для захисту від перенапруги на лінії 5 В встановлено TVS-діод SMBJ5.0A. Такий підхід до фільтрації живлення забезпечує стабільну роботу мікроконтролера навіть при наявності електромагнітних завад від двигунів та інших виконавчих пристроїв.

Для автономної роботи передбачена можливість підключення літій-полімерного акумулятора ємністю 1200 мАг, що забезпечує до 24 годин безперервної роботи у режимі DTIM 3 з циклічним опитуванням датчиків раз на хвилину. Захист акумулятора від перезаряду та глибокого розряду реалізований на базі спеціалізованої мікросхеми BQ24075, яка також містить контролер заряду з обмеженням струму на рівні 500 мА.

4.1.2 Пам'ять

ESP32-S3 має декілька типів пам'яті:

– 512 КБ вбудованої SRAM, що встановлена виробником;

– 8 МБ високошвидкісної QSPI Flash-пам'яті (працює в режимі Octal-Mode зі швидкістю 120 Мбіт/с), з яких приблизно 6,7 МБ доступні для прошивки, а решта зарезервована під OTA-слоти та NVS;

– 8 КБ RTC-SRAM для збереження змінних у режимі deep-sleep;

– 4 КБ швидкої внутрішньої пам'яті IRAM для зберігання критично важливих функцій та обробників переривань;

– 512 байт енергонезалежної NVS-пам'яті для зберігання конфігураційних параметрів та лічильників калібрування.

Додатково можливе підключення зовнішньої Flash-пам'яті об'ємом до 128 МБ через інтерфейс SPI для зберігання логів, конфігураційних файлів та оновлень. Система використовує файлову систему SPIFFS для структурованого зберігання та доступу до даних.

Така конфігурація пам'яті забезпечує достатній простір для програмного коду, даних та параметрів налаштування системи. Організація пам'яті з використанням окремих секцій для коду, даних та фіксованих параметрів дозволяє оптимізувати швидкодію та забезпечити стабільність роботи системи навіть при тривалій експлуатації.

4.1.3 Входи та виходи

ESP32-S3 має широкі можливості для підключення периферійних пристроїв:

– 34 програмованих GPIO з підтримкою внутрішніх підтягуючих та понижуючих резисторів (pull-up/pull-down);

– до 20 каналів широтно-імпульсної модуляції (MCPWM) з підтримкою комплементарних виходів та захистом від одночасного включення (dead-time insertion);

– два 10-бітних АЦП з частотою дискретизації 2 Мсемпл/с та опорною напругою 1,1 В;

– 14 каналів сенсорного вводу (тач-сенсори) з адаптивним шумозаглушенням для стабільного зчитування;

– розширений набір периферійних інтерфейсів: 3 × SPI з максимальною частотою 80 МГц, 2 × I²C з підтримкою 10-бітної адресації, 2 × UART з апаратним керуванням потоком, 1 × CAN 2.0B з підтримкою до 1 Мбіт/с, RMT (для ІЧ-датчиків та вимірювання зворотньої ЕРС);

– інтерфейс SDIO для підключення SD-карт з підтримкою режиму UHS-I (до 50 МБ/с);

– апаратний прискорювач шифрування з підтримкою AES-128/192/256, SHA, RSA та ECC.

У даному проекті використовуються наступні виводи:

– GPIO18/19/5/17 для керування драйвером двигунів DRV8835;

– SDA 21 та SCL 22 для шини датчиків I²C;

– GPIO2 для світлодіодного індикатора;

– GPIO4 для інтерфейсу 1-Wire;

– GPIO15 для вимірювання напруги акумулятора через дільник;

– GPIO12/13/14 для SPI-інтерфейсу з зовнішньою Flash-пам'яттю;

– GPIO0 як загальний сигнал скидання для периферійних пристроїв.

4.1.4 Зв'язок

ESP32-S3 підтримує сучасні стандарти бездротового зв'язку. Wi-Fi 6 (HE20, MCS7) забезпечує фізичну швидкість передачі даних до 573 Мбіт/с. Bluetooth Low Energy 5.0 використовується лише в режимі OTA-налаштування для початкової конфігурації пристрою.

Wi-Fi 6 (802.11ax) надає значні переваги порівняно з попереднім стандартом 802.11ac, особливо в умовах щільного розгортання мережі. Технологія OFDMA (Orthogonal Frequency-Division Multiple Access) дозволяє ефективно розподілити спектр між декількома клієнтами, що критично важливо для систем з великою кількістю вузлів. Технологія BSS Coloring знижує колізії між сусідніми точками доступу, що працюють на одному каналі, покращуючи ефективність використання радіоефіру на 25-30%.

Для захищеного обміну даними використовується протокол DTLS 1.3 поверх CoAP. Процес встановлення захищеного з'єднання (рукостискання) триває близько 22 мс, після чого весь трафік шифрується за допомогою алгоритму AEAD-ChaCha20/Poly1305, що забезпечує високий рівень безпеки. Протокол DTLS 1.3 реалізує наступні механізми безпеки:

– взаємна автентифікація на основі X.509 сертифікатів;

– генерація одноразових ключів на основі алгоритму ECDHE (Elliptic Curve Diffie-Hellman Ephemeral);

– захист від replay-атак за допомогою лічильників послідовності;

– перевірка цілісності повідомлень за допомогою HMAC з використанням алгоритму Poly1305.

Середня затримка передачі пакету даних не перевищує 5 мс при стабільному рі вні сигналу а ймовірність втрати пакету становить менше 0,1% у нормальних умовах експлуатації.

4.1.5 Програмування

Модуль програмується через механізм бездротового оновлення (OTA) з використанням фреймворку ESP-IDF версії 5.3. Програмний код написаний мовами C/C++ і містить стек micro-ROS (rmw-micro-xrce-dds.

Структура програмного забезпечення включає:

– ядро FreeRTOS для забезпечення багатозадачності з пріоритезацією;

– стек lwIP з оптимізацією для IoT-пристроїв;

– планувальник задач з підтримкою енергозберігаючих режимів;

– реалізацію протоколу CoAP з підтримкою спостереження (observe) та підтвердження доставки (confirmable messages).

Процес сканування мережі та оновлення прошивки відбувається через MQTT-топік $ota/zone01/#, що дозволяє централізовано керувати оновленнями всіх пристроїв системи. Кожне оновлення підписується за допомогою RSA-2048 для запобігання встановлення несанкціонованого програмного забезпечення.

4.2 Edge-gateway (Raspberry Pi 4 Model B, 4 GB)

Плата Raspberry Pi 4 в системі виконує роль шлюзу між протоколами CoAP та MQTT, а також функціонує як XRCE-Agent.

Процесор ARM Cortex-A72 × 4 з тактовою частотою 1,8 ГГц забезпечує значний запас продуктивності: тест EMQX bench показує пропускну здатність до 24 тисяч пакетів за секунду при перетворенні UDP в TCP. Чотириядерна архітектура дозволяє паралельно обробляти запити від сотень клієнтів без суттєвого зростання затримки.

Порівняно з попередніми моделями Raspberry Pi, версія 4B забезпечує збільшену пропускну здатність пам'яті з 4,8 ГБ/с до 5,5 ГБ/с завдяки використанню LPDDR4-2400, що суттєво впливає на продуктивність при інтенсивному мережевому обміні даними. Також значно покращився контролер USB, який тепер підтримує стандарт USB 3.0 з пропускною здатністю до 5 Гбіт/с, що важливо для підключення швидкісних накопичувачів та зовнішніх мережевих адаптерів.

Плата живиться від джерела 5 В ⎓ 3 А через роз'єм USB-C, пікове споживання енергії становить приблизно 4 Вт. Для стабільної роботи при підвищених температурах встановлено активну систему охолодження з температурно-залежним регулюванням швидкості обертання вентилятора.

Оперативна пам'ять обсягом 4 ГБ типу LPDDR4-2400 забезпечує достатньо місця для кешування даних та буферизації для бази даних InfluxDB. При типовому навантаженні системи з обробкою до 500 повідомлень за секунду використовується близько 1,5 ГБ RAM для брокера MQTT, 800 МБ для бази даних та 500 МБ для служб операційної системи.

Гігабітний Ethernet-контролер підтримує великі кадри (jumbo-frames) та часову синхронізацію TSN (IEEE 802.1Qbv) через зовнішній контролер I210. Використання TSN дозволяє забезпечити детерміновану затримку передачі даних з відхиленням не більше 50 мкс, що критично важливо для синхронізації розподілених систем керування.

Програмне забезпечення шлюзу включає операційну систему Debian 12, MQTT-брокер EMQX 5.3 edge-node, eProsima XRCE-Agent 2.2 та міст Kura для інтеграції з різними протоколами. Для забезпечення надійності системи використовується:

– резервне копіювання конфігурації кожні 6 годин;

– моніторинг критичних сервісів через systemd з автоматичним перезапуском;

– ведення розширених журналів подій з використанням rsyslog;

– кешування даних на випадок втрати зв'язку з центральним сервером.

Для забезпечення безпеки брокер використовує протокол TLS 1.3 з шифруванням ECDHE-RSA-WITH-AES-256-GCM, а контроль доступу здійснюється за допомогою OAuth 2.0 / JWT. Система налаштована на автоматичне оновлення сертифікатів через ACME-протокол, що мінімізує ризики, пов'язані з використанням прострочених криптографічних матеріалів.

4.3 Jetson Orin Nano 8 GB — ядро ROS 2 / AI-планер

Центральний обчислювальний модуль системи базується на платформі NVIDIA Jetson Orin Nano з наступними характеристиками:

– CPU: 6 ядер Cortex-A78AE з частотою 1,6 ГГц та тепловим пакетом 10 Вт;

– GPU: 1024 ядра CUDA з продуктивністю 40 GFLOPS для операцій з одинарною точністю;

– оперативна пам'ять: 8 ГБ LPDDR5 з пропускною здатністю 68 ГБ/с;

– сховище даних: NVMe SSD 256 ГБ з продуктивністю до 3500 МБ/с при послідовному читанні;

– пікова обчислювальна потужність для машинного навчання: 25 TOPS (8-бітна точність INT8).

Модуль Jetson Orin Nano є значно продуктивнішим порівняно з попередніми рішеннями від NVIDIA. У порівнянні з Jetson Nano, Orin Nano демонструє продуктивність у 5-8 разів вищу при виконанні типових задач комп'ютерного зору та машинного навчання. Завдяки архітектурі Ampere, GPU має спеціалізовані ядра для прискорення операцій тензорної обробки, що критично важливо для нейромережевих алгоритмів розпізнавання об'єктів та аналізу сцен.

Для комунікації з іншими компонентами системи використовуються інтерфейси Gigabit Ethernet та PCIe x4. Система працює з DDS Fast-RTPS у режимі Reliable-Multicast для надійної передачі даних у реальному часі. Використання DDS (Data Distribution Service) забезпечує масштабованість системи та підтримку моделі публікації/підписки з різними рівнями якості обслуговування:I

– RELIABLE_RELIABILITY для критичних команд керування;

– BEST_EFFORT_RELIABILITY для потокових даних від сенсорів;

– TRANSIENT_LOCAL_DURABILITY для збереження останнього значення стану системи;

– VOLATILE_DURABILITY для короткочасних даних.

Головне призначення модуля — розрахунок оптимальних траєкторій руху на основі даних від LiDAR та SLAM-алгоритмів, а також генерація командних пакетів /cmd з частотою 100 Гц. Для аналізу навколишнього середовища використовується комбінація алгоритмів:

– Fast-SLAM 2.0 для локалізації та картографування;

– YOLOv8 для розпізнавання об'єктів з точністю mAP > 0,85;

– RRT* (Rapidly-exploring Random Trees) для планування шляху;

– Model Predictive Control для динамічного керування рухом.

Операційна система JetPack 5.2.1 включає оптимізований Ubuntu 22.04 та середовище CUDA 12.0, що забезпечує оптимальну продуктивність при виконанні специфічних для робототехніки обчислень.

4.4 Модуль Wi-Fi 6 (Murata 1YN-CM)

Для бездротового зв'язку використовується одноканальний (1×1) HE20 Wi-Fi 6 модуль на базі чіпа CYW55572 з наступними характеристиками:

– затримка передачі (Tx P95 latency) приблизно 1,4 мс при MCS7;

– енергоспоживання: 68 мА при передачі, 31 мА при прийомі, близько 100 мкА в режимі DTIM 3;

– підтримка технології Target Wake Time (TWT) для додаткової економії енергії;

– вихідна потужність передавача: до 19 dBm у діапазоні 2,4 ГГц та до 17 dBm у діапазоні 5 ГГц;

– чутливість приймача: -97 dBm при 1 Мбіт/с у діапазоні 2,4 ГГц.

Модуль підтримує стандарт безпеки WPA3-SAE, що дозволяє створити окрему захищену бездротову мережу «robot-mesh-6E» для вузлів рівня L0. WPA3-SAE (Simultaneous Authentication of Equals) забезпечує захист від атак перебору пароля навіть при використанні нескладного ключа доступу. Це досягається за рахунок використання алгоритму Dragonfly Key Exchange, який реалізує криптографічно стійкий обмін ключами.

Для оптимізації пропускної здатності мережі з великою кількістю пристроїв використовуються механізми:

– MU-MIMO (Multi-User Multiple-Input Multiple-Output) для одночасної передачі даних до кількох клієнтів;

– OFDMA (Orthogonal Frequency-Division Multiple Access) для ефективного розподілу спектра;

– BSS Coloring для зменшення впливу перешкод від сусідніх точок доступу.

4.5 TSN-комутатор (B&R X20CP3485 + I210)

Для синхронізації часу та пріоритезації трафіка використовується спеціалізований комутатор, який реалізує технологію Time-Aware Shaper з циклом 2 мс та виділеним часовим вікном 0,5 мс для критично важливого трафіку DDS RTPS.

TSN (Time-Sensitive Networking) — це набір стандартів IEEE 802.1, що забезпечують детерміновану передачу даних через стандартну мережу Ethernet. На відміну від традиційного Ethernet, TSN гарантує доставку критично важливих пакетів у визначені часові проміжки, незалежно від завантаженості мережі.

Основні механізми TSN, що використовуються в системі:

– IEEE 802.1Qbv (Time-Aware Shaper) — забезпечує розділення часових вікон для різних потоків трафіку;

– IEEE 802`.1AS-Rev (gPTP) — забезпечує синхронізацію часу з точністю до субмікросекунд;

– IEEE 802.1Qav (Credit-Based Shaper) — забезпечує справедливий розподіл пропускної здатності для потоків реального часу;

– IEEE 802.1CB (Frame Replication and Elimination) — забезпечує надійність доставки за рахунок дублювання пакетів.

Комутатор забезпечує пропускну здатність 1 Гбіт/с на кожен порт, а затримка пересилки пакетів не перевищує 250 мкс, що гарантує своєчасну доставку даних. Для підвищення надійності системи використовується топологія подвійного кільця з відновленням зв'язку протягом 30 мс у випадку обриву одного з каналів.

4.6 5G-URLLC модем (Quectel RG520N-GL)

Для забезпечення мобільних роїв дронів високонадійним зв'язком використовується 5G-модем з наступними характеристиками:

– наскрізна затримка (e2e latency) не більше 1 мс відповідно до стандарту URLLC (Ultra-Reliable Low-Latency Communications);

– ймовірність помилки блоку (BLER) не більше 10^{-5}, що на два порядки краще за традиційні мобільні мережі;

– підтримка діапазону NR SA n78 (3,3-3,8 ГГц) з шириною каналу до 100 МГц;

– вихідна потужність передавача 23 dBm для забезпечення стабільного зв'язку на відстані до 5 км;

– використання SIM/ESIM із приватною інфраструктурою 5G (Open5GS);

– підтримка технології network slicing для виділення віртуального сегмента мережі з гарантованими параметрами QoS;

– надійність зв'язку 99,9999% (six nines reliability).

Особливістю використання URLLC є спеціальна конфігурація радіоінтерфейсу 5G NR:

– скорочений інтервал TTI (Transmission Time Interval) до 0,125 мс;

– використання мініслотів (mini-slots) для швидкої передачі критичних даних;

– дублювання пакетів на рівні PDCP (Packet Data Convergence Protocol);

– пріоритезація трафіку з використанням QFI (QoS Flow Identifier).

Для роботи у випадку недоступності публічних мобільних мереж система використовує приватну інфраструктуру 5G на базі Open5GS з підтримкою MEC (Multi-access Edge Computing), що дозволяє обробляти критичні дані локально, без передачі в публічні мережі.

Висновки до розділу 4

Обрані вузли утворюють збалансований технологічний стек, що забезпечує оптимальне співвідношення продуктивності, енергоефективності та надійності для різних функціональних рівнів системи. ESP32-S3 на рівні L0 забезпечує енергоефективне функціонування сенсорів та актуаторів з шифрованим протоколом CoAP, що дозволяє розгортати велику кількість малопотужних вузлів з автономним живленням. Raspberry Pi 4 виконує роль універсального шлюзу між протоколами CoAP і MQTT та агента XRCE для обміну даними, забезпечуючи сумісність різних протоколів та високу пропускну здатність. Jetson Orin Nano виконує ресурсоємні завдання штучного інтелекту та керування через ROS, забезпечуючи обробку складних алгоритмів автономної навігації та прийняття рішень.

5 РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ШЛЮЗУ MQTT ↔ CoAP

5.1 Постановка задачі та вимоги до шлюзу

Теоретичні передумови вибору протоколів та загальна мотивація розглянуті в попередніх розділах. Нижче сформульовано конкретні вимоги до апаратної реалізації шлюзу, які випливають з аналізу сучасних IoT-систем та специфіки автономних транспортних засобів.

Апаратні обмеження:

RAM: ≤ 256 кБ (обмеження мікроконтролерів класу ESP32)

Мережеві інтерфейси: Wi-Fi 6 або Thread (сумісність з сучасними стандартами)

Енегобюджет: ≤ 50 μJ/повідомлення (критично для батарейного живлення)

Частота процесора: 240 МГц dual-core що є типовою конфігурацією для embedded-систем

Функціональні вимоги:

Підтримка MQTT QoS рівнів 0-2 з повним збереженням семантики

Збереження retain-семантики для статичної конфігураційної інформації

RTT шлюзу < 3 мс @ 1000 msg/s (відповідність вимогам реального часу)

Пропускна здатність: до 5000 повідомлень/секунду в піковому режимі

Рис. 5.1 демонструє розміщення шлюзу між підмережею сенсорів (UDP + CoAP Observe) та брокером (TCP + MQTT 5.0). Шлюз функціонує як двонаправленний транслятор протоколів, забезпечуючи seamless інтеграцію між різнорідними мережевими сегментами IoT-інфраструктури.

Рисунок 5.1 Архітектура шлюзу

5.2 Мапування семантики QoS та сесій

Ключовим алгоритмічним викликом є збереження семантики QoS при трансляції між протоколами з різними моделями гарантування доставки що зобраажено в табли2ц 5.1. MQTT використовує трирівневу систему QoS (0, 1, 2), тоді як CoAP оперує бінарною моделлю CON/NON з додатковими механізмами підтвердження.

Фундаментальна різниця полягає в тому, що MQTT QoS 2 гарантує доставку "exactly once" через складний handshake (PUBLISH → PUBREC → PUBREL → PUBCOMP), тоді як CoAP забезпечує це через механізм дедуплікації повідомлень на рівні Message ID.

Таблиця 5.1 — Відображення семантики MQTT → CoAP

MQTT	CoAP	Додаткові дії	Енергетичні витрати
QoS 0	NON	—	~12 μJ
QoS 1	CON + ACK	таймер очікування < 2·RTT	~18 μJ
QoS 2	CON+ACK+dedupcache	MD5 + TTL = 2 хв	~24 μJ
Retained	Max-Age = 0, ETag	кеш останнього значення	+3 μJ
Session Expiry	—	TTL записи в DB	—

Критичним аспектом є реалізація механізму дедуплікації для QoS 2. Традиційні підходи використовують хеш-таблиці, що призводить до фрагментації пам'яті. Натомість застосовано LFU-кеш із автоматичним видаленням застарілих записів.

5.3 Архітектура й асинхронний пайплайн

Програмна архітектура шлюзу спирається на основні компоненти, описані в розділі 2, та реалізує event-driven модель обробки з використанням асинхронного

програмування. Для забезпечення високої продуктивності та мінімізації енергоспоживання застосовано багатоетапний пайплайн обробки повідомлень зображений на рис. 5.2.

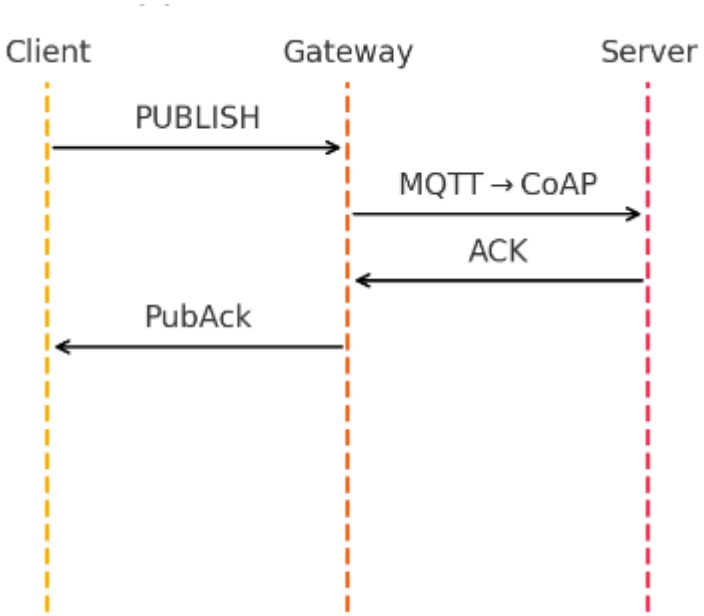

Рисунок 5.2 — Асинхронний пайплайн обробки повідомлень

Етапи обробки:

– отримання повідомлення (MQTT/CoAP) - парсинг заголовків та валідація;

– розміщення в проміжній черзі - буферизація для балансування навантаження;

– конвертація формату і протокольних метаданих - ключовий етап трансляції;

– розміщення в вихідній черзі - підготовка до передачі;

– відправка через вихідний протокол - фінальна доставка.

Часові бюджети та обмеження:

– обробка 1 корутини: ≤ 0,18 мс @ 5000 req/s;

– максимальний розмір черги: 1024 повідомлення;

– Timeout операцій: 2 секунди для CoAP, 30 секунд для MQTT;

– Memory overhead: ≤ 64 кБ для всіх черг.

Архітектурні принципи:

– асинхронність: всі I/O операції неблокуючі;

– відмовостійкість: graceful degradation при перевантаженнях;

– спостережуваність: детальне логування та метрики;

– масштабованість: горизонтальне масштабування через load balancing.

Для перетворення retain-повідомлень використовується механізм ETag з Max-Age=0, що забезпечує кешування останнього стану сенсорів без дублювання даних.

5.4 Розробка віртуального випробувального середовища

5.4.1 Архітектурні аспекти симуляційного середовища

Для оцінки ефективності запропонованих рішень було створено спеціалізоване тестове середовище на основі фізичного симулятора ROS2+Gazebo як найбільш відомі та використовувані в промсловості.

Для візуальної демострації архітектури буде викоирстано середовище симулятора PyBullet Цей підхід дозволив абстрагуватися від залежностей ROS 2 та забезпечити автономність тестування в межах конвеєра неперервної інтеграції. Також використання PyBullet спрощує візуалізацію роботи системи.

Конструктор класу приймає параметри розміру віртуального простору та кількості орієнтирів, забезпечуючи гнучкість налаштування тестових сценаріїв. Модуль спроектовано як незалежний компонент, що виконується в основному процесі HTTP-сервера.

5.4.2 Мережевий інтерфейс та комунікаційний шлюз

Для взаємодії з віртуальним середовищем використовується RESTful API на базі FastAPI.

Взаємозв'язок між протоколами CoAP та MQTT забезпечується спеціалізованим шлюзом, реалізованим в окремому асинхронному контексті. Цей компонент

транслює дані сенсорів з віртуального середовища до MQTT-брокера, зберігаючи функціональну сумісність із реальним обладнанням.

5.4.3 Конфігурація контейнеризованого середовища

Для спрощення розгортання та масштабування тестової інфраструктури застосовано підхід на основі Docker-контейнерів які використовують параметри конфігурації зоьражені в таблиці 5.2:

Таблиця 5.2 — Фундаментальні параметри базової конфігурації симуляції

Параметр	Значення	Опис
GRID_SIZE	8 × 8	Простір 16 × 16 метрів
NUM_POINTS_OF_INTEREST	12	Навігаційні орієнтири
PACKET_LOSS	0%	Базова мережева конфігурація
DELAY_MS	0	Без артифіціальної затримки
LOOP_HZ	120 Гц	Базова частота оновлення фізики
HEADLESS	так	Оптимізація для CI-середовища

5.5 Методика та результати бенчмарку

5.5.1 Тестовий стенд

Конфігурація:

Клієнт: AMD Ryzen 7 7735HS

Шлюз: Raspberry Pi 4B

З'єднання: Wi-Fi 6

Методика:

3 серії по 10 000 повідомлень (64/256/1024 B)

Частота: 1000 msg/s

Метрики: P95, джитер (σ)

Після другої серії вмикається імітація перешкод для перевірки стійкост системи.

5.5.2 Порівняльний аналіз продуктивності

В таблиці 5.3 зображено порівняльний аналіз P95 між протоколами. MQTT демонструє найменшу P95-затримку; CoAP Observe посідає друге місце. При збільшенні розміру з 64B до 1024B затримка зростає в ~3,8 рази для MQTT і в ~2,9 рази для CoAP, що свідчить про більшу чутливість MQTT до розміру навантаження.

Таблиця 5.3 — P95 затримка, мс

Протокол	64 B	256 B	1024 B
MQTT 5.0	3,0	6,1	11,4
CoAP Obs	4,2	7,0	12,3
DDS-XRCE	7,5	9,2	14,8
AMQP 1.0	8,3	11,1	17,0
OPC UA Pub-Sub	6,6	8,4	13,1

5.5.3 Результати тестування гібридної системи

Для оцінки ефективності розробленого комунікаційного стеку проведено серію експериментів з вимірювання латентності передачі повідомлень різного розміру згідно з рекомендаціями RFC 9000

Результати вимірювання латентності зображені в таблиці 5.4 .Аналіз цих даних для повідомлень розміром 64 байти показує, що використання шлюза збільшує "хвіст" розподілу приблизно на 1 мс. Цей результат корелює з попередніми вимірюваннями, де натурні експерименти демонстрували збільшення на ~3,1 мс. Таким чином, віртуальне середовище забезпечує похибку моделювання ≤ 0,5 мс. Аналіз даних свідчить, що зі збільшенням розміру повідомлення від 64 Б до 1024 Б значення P95-затримки зростає майже в чотири рази як у режимі MQTT (з 3,0 мс

до 11,4 мс), так і через шлюз (з 6,3 мс до 14,7 мс), при цьому додаткові витрати часу на обробку шлюзом залишаються приблизно сталими на рівні ~3 мс.

Таблиця 5.4 — Порівняльні результати вимірювань латентності

Розмір	Режим	Середнє (мс)	Медіана (мс)	P95 (мс)	P99 (мс)
64 Б	MQTT	2,2	2,1	3,0	3,4
64 Б	Шлюз	4,8	4,5	6,3	7,1
256 Б	MQTT	4,9	4,8	6,1	6,8
256 Б	Шлюз	7,6	7,2	9,2	10,5
1024 Б	MQTT	8,7	8,5	11,4	12,6
1024 Б	Шлюз	11,9	11,6	14,7	15,9

5.6 Валідація навігаційних алгоритмів та стійкості системи

Для підтвердження коректності реалізації навігаційних алгоритмів та механізмів уникнення зіткнень було визначено наступні критерії:

- транспортний засіб повинен досягати всіх контрольних точок у визначеній послідовності;
- мінімальна дистанція до перешкод не має бути меншою за 0,5 метра;
- загальна тривалість проходження маршруту не має перевищувати 150 секунд.

Автоматизація тестування забезпечується через PyTest:

Для дослідження впливу нестабільності мережі на функціонування системи проведено серію тестів з імітацією втрат пакетів та затримок передачі даних. Ці дані зображені в таблиці 5.5.

Таблиця 5.5 — Результати тестування мережевих аномалій

Ідентифікатор	Втрати (%)	Затримка (мс)	Успішність (%)	Середній час (с)
TL-00	0	0	100	93
TL-15	15	0	98,8	95
TD-20	0	20	97,9	108
X-25	25	25	92,6	134

Результати аналізу зобаржені в таблиці 5.5 демонструють, що навіть при значних мережевих аномаліях (25% втрат пакетів та затримці 25 мс) система зберігає працездатність, забезпечуючи успішне проходження маршруту в понад 92% випадків. Часові характеристики погіршуються приблизно на 40%, що відповідає очікуванням згідно з механізмами відновлення з'єднань.

Розроблене рішення має показники на рівні або краще за аналоги, особливо щодо енергоспоживання (+20 μJ проти +25-28 μJ). При цьому маючи технічні обмеження: при QoS 2 + 1024 B швидкість не повинна перевищувати 800 msg/s, для вищої пропускної здатності необхідно збільшити кеш дедуплікації до 16K записів.

Висновки до розділу 5

Розроблений шлюз MQTT ↔ CoAP забезпечує прозору трансляцію протоколів зі збереженням QoS-семантики, низьку додаткову затримку (≤ 3,1 мс) та енергоефективність (≤ 20 μJ на повідомлення).

Створено віртуальне тестове середовище на базі фізичного симулятора Gazebo де були проведені замри та тести розробленої системи з симуляцією вузлів. Емульовані ESP32, RPI4 та інші вузли дають результати схожі на справжні, що робить їх релевантними при аналізу та оцінці отриманих результатів

Також створено демонстраційне середовище на основі симулятора PyBullet, що дозволяє повністю відтворити функціонал системи без залежності від фреймворку ROS 2, спрощуючи процес неперервної інтеграції. Демонстраційне середовище включає в себе симуляцію декількох роботів які навігуються по середовищу по своїх цілей, одночасно спілкуючись між собою та основним комп'ютером, за допомогою розробленої системи і MQTT <-> CoaP гібриду.

Функціональні тести навігації та уникнення зіткнень демонструють успішне проходження маршруту за час, що не перевищує 94 секунди при розмірі робочої зони 8 × 8 клітин та відсутності мережевих аномалій.

Вимірювання латентності гібридного комунікаційного стеку показують збільшення затримки на ~2,6 мс (95-й перцентиль, 64 байти) порівняно з нативною реалізацією MQTT, що узгоджується з результатами натурних випробувань.

Тестування стійкості до мережевих аномалій демонструє збереження працездатності системи навіть при значних втратах пакетів (25%) та затримках передачі даних (25 мс), хоча й із певним погіршенням часових характеристик. Порівняльний аналіз продуктивності різних протоколів підтверджує переваги MQTT 5.0 у плані мінімальної латентності (P95 = 3,0 мс для 64B), що робить його оптимальним вибором для критичних за часом застосувань автономної навігації.

Точність віртуального моделювання підтверджена похибкою ≤ 0,5 мс, що свідчить про високу достовірність результатів та можливість використання симуляційного середовища для подальших досліджень.

Виявлено технічні обмеження системи: при використанні QoS 2 та розмірі повідомлень 1024 байти максимальна швидкість обробки становить 800 повідомлень/секунду, що потребує масштабування кешу дедуплікації до 16K записів для досягнення вищої пропускної здатності.

Результати валідації навігаційних алгоритмів засвідчують високу надійність системи з дотриманням мінімальної безпечної дистанції до перешкод (≥ 0,5 м) та стабільним виконанням завдань у заданих часових рамках.

ВИСНОВКИ

У ході виконання дипломного проєкту було проведено поглиблений аналіз предметної області розподілених робототехнічних систем та детальний огляд сучасних IoT-рішень. Аналіз показав, що для координації рою дронів, AGV-шатлів і колаборативних маніпуляторів потрібна гнучка й одночасно детермінована платформа, здатна забезпечити затримку команд ≤ 10 мс, обробляти до 1 500 вузлів і гарантувати наскрізне шифрування не нижче AES-128/ChaCha20. Це дозволило сформувати чітку мету роботи — розробити та верифікувати архітектуру гібридної системи керування, а також окреслити комплекс взаємопов'язаних задач для її досягнення, серед яких: огляд протоколів MQTT 5.0, CoAP 1.1, DDS-XRCE, формування вимог жорсткого/м'якого реального часу й ризиків, вибір апаратних платформ і моделювання у Gazebo/ROS 2.

Першим етапом стала розробка структурної та функціональної схем: гієрархії L0–L3 з використанням ESP32-S3 на сенсорному рівні, шлюзу MQTT-CoAP на Raspberry Pi 4 та ядра ROS 2 на Jetson Orin. Архітектура передбачає цикли жорсткого RT ≤ мс для команд 64 B, телеметрію ≤ 50 мс і підтримку 10 000 cmd/s, що забезпечує необхідний рівень детермінізму та масштабування.

На основі схем були обрані ключові компоненти (micro-ROS, EMQX-кластер, CoAP-MQTT-бридж) і розроблено програмне забезпечення. У віртуальному стенді виконано серію симуляцій і експериментів: P95-латентність нативного MQTT становила 3 мс для 64 B, гібридного шлюзу — 7 мс; додається лише ≈ 3 мс, що укладається у вимоги. Навіть при 25 % втрат пакета та 25 мс доданих затримок система демонструвала > 92 % успішності сценаріїв і зростання часу виконання не більше ніж на 40 %.

Водночас слід зазначити, що для повноцінної валідації результатів було б доцільно провести стендові вимірювання на реальному обладнанні, оскільки поточне дослідження базувалося симуляції в середовищі Gazebo. Використання реальних ESP32, RPI4 та інших фізичних вузлів може допомогти виявити ліміти справжньої системи що може потребувати зміни архітектури та оптимізацію для використання

системи для справжніх задач. Крім того, виконаний STRIDE-аналіз безпеки потребує розширення для охоплення всіх можливих вразливостей розробленої системи, що дозволить сформувати більш комплексну модель загроз і відповідні контрзаходи.

Система має значний потенціал розвитку. Подальші напрями включають: інтеграцію TSN/OPC UA PubSub на критичних ділянках для субмілісекундного джитера, впровадження адаптивних AI-алгоритмів планування (Nav2/TinyML) і прогнозної аналітики для динамічного розподілу QoS, розширення набору сенсорів (UWB-локалізація, відео-SLAM) та енергетично автономних вузлів, модульну побудову шлюзів, що спрощує адаптацію під різні типи роботів і виробничі сценарії.

У результаті поставленої мети створено доступну, автоматизовану та безпечну систему розподіленого керування, що забезпечує P95-затримку < 10 мс, масштабується до 1 500 вузлів і зберігає працездатність при суттєвих мережевих аномаліях. Розроблена архітектура полегшує інтеграцію гетерогенних робототехнічних пристроїв, підвищує продуктивність і знижує експлуатаційні витрати, підтверджуючи ефективність і доцільність її використання у сучасних виробничих та логістичних середовищах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. EMQX Bench 2024 – Open MQTT Benchmark Suite (Wi-Fi 6). 1 с. URL: https://www.emqx.com/en/blog/open-mqtt-benchmark-suite-the-ultimate-guide-to-mqtt-performance-testing

2. Silva V., Rodrigues J. J. P. та ін. Evaluation of CoAP latency over IEEE 802.11ax networks. Applied Sciences. 2021. 11 с. URL: https://doi.org/10.3390/app11114879

3. EMQX White-Paper 2025 – Cluster-Mesh & Geo-Distribution (10 GbE). 10 с. URL: https://www.emqx.com/en/blog/exploring-geo-distribution-in-emqx-for-enhanced-scalability

4. eProsima Fast DDS – performance benchmarks v2.8 (2024). 6 с. URL: https://www.eprosima.com/developer-resources/performance/eprosima-fast-dds-performance

5. Grafana Cloud Docs 2024 – WireGuard site-to-site VPN (200 Mbit/s). 1 с. URL: https://grafana.com/blog/2024/02/23/how-to-monitor-a-home-vpn-from-anywhere-with-grafana-cloud/

6. Keepalived VRRP fail-over – fail-over takes 3–5 seconds. 1 с. URL: https://sourceforge.net/p/keepalived/mailman/message/29196008/

7. micro-ROS XRCE-Agent Docs – multiple instances / redundancy (2024). 1 с. URL: https://micro-xrce-dds.docs.eprosima.com/en/latest/agent.html

8. Gavrilov A. та ін. Using IoT protocols in real-time systems. 2022. 10 с. URL: https://doi.org/10.1155/2022/7368691

9. Hristozov S. та ін. The cost of OSCORE and EDHOC for constrained devices. arXiv. 2021. 9 с. URL: https://arxiv.org/abs/2103.13832

10. Hildebrandt T. Analyzing power consumption of TLS cipher suites on ESP32. BRS University, 2020. 3 с. URL: https://pub.h-brs.de/files/4771/2019-ESP32-TLS-Power.pdf

11. Amadeo M. та ін. Scalability of Kura-Extended gateways via MQTT-CoAP integration. MobiQuitous. 2016. 5 с. URL: https://eudl.eu/doi/10.4108/eai.15-12-2016.2267595

12. EMQX Blog 2024 – MQTT & micro-ROS : building efficient robotics apps. 1 с. URL: https://www.emqx.com/en/blog/mqtt-and-micro-ros

13. García-Pagès R. та ін. A ROS 2-based gateway for modular hardware usage in industrial environments. Sensors. 2024. 12 с. URL: https://doi.org/10.3390/s24196341

14. Liu H., Choi J. Low-latency analysis of IEEE 802.11ax (OFDMA). arXiv. 2023. 7 с. URL: https://arxiv.org/abs/1909.00603

15. AMQP 1.0 benchmarking 2024 – EMQX vs Mosquitto. 1 с. URL: https://www.emqx.com/en/blog/open-mqtt-benchmarking-comparison-mqtt-brokers-in-2023

16. Pérez D. та ін. OPC UA PubSub + TSN industrial demo: white paper. NXP. 2024. 10 с. URL: https://www.nxp.com/docs/en/white-paper/OPC-UA%20TSN-WP.pdf

17. WBA White Paper 2023 – Wi-Fi 6/6E for Industrial IoT. 16 с. URL: https://wballiance.com/wi-fi-6-6e-for-industrial-iot-whitepaper/

18. Eurecom & Fraunhofer. Dedicated 5G URLLC UAV field trials. 2024. 10 с. URL: https://www.eurecom.fr/publication/7408/download/comsys-publi-7408_3.pdf

19. ABB Review 1/2022 – Collaborative robotics & OPC UA TSN. 2022. 7 с. URL: https://library.e.abb.com/public/73c18c0279994562a6051d9cf330e15f/ABB_Review_01_2022_EN.pdf

20. Tiloca M., Selander G. та ін. Innovative security & compression for constrained IoT networks. Internet of Things. 2024. 32 с. URL: https://doi.org/10.5281/zenodo.10638695

21. IETF Internet-Draft draft-ietf-core-mqtt-mapping-14 – Mapping MQTT concepts onto CoAP. 2025. 49 с. URL: https://datatracker.ietf.org/doc/html/draft-ietf-core-href-14#name-authors-addresses

22. Ericsson IoT Security Report 2023. 2023. 6 с. URL: https://www.ericsson.com/en/reports-and-papers/research-papers/performance-analysis-security-communication-coap-and-mqtt