ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ДОСЛІДЖЕННЯ СТАНДАРТІВ ТА ПРОТОКОЛІВ В ОБЛАСТІ IoT»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

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

_____ *Олексій МАРКІН*

Виконав:
здобувач вищої освіти Олексій МАРКІН
група ІСДМ-61

Керівник: Оксана ТКАЛЕНКО
науковий ступінь, к.т.н., доцент
вчене звання

Рецензент:
науковий ступінь, _____
вчене звання Ім’я, ПРІЗВИЩЕ

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
Маркіну Олексію Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: «Дослідження стандартів та протоколів в області IoT»
керівник кваліфікаційної роботи Оксана ТКАЛЕНКО, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «19» жовтня 2023 року № 145.

2. Строк подання кваліфікаційної роботи: 29 грудня 2023 року.

3. Вихідні дані до кваліфікаційної роботи: Інструменти, розроблені за допомогою протоколу MQTT;
Технічні вимоги;
Науково-технічна література з питань, пов'язаних з технологіями IoT.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Стандартизація в області інформаційних технологій. Стандартизація Інтернету речей.

2. Дослідження засобів взаємодії протоколів IoT.
3. Впровадження та реалізація протоколу MQTT.

5. Перелік ілюстративного матеріалу: *презентація*
   1. Характеристика технологій для IoT/M2M та порівняння технологій.
   2. Стек протоколів IoT.
   3. Використання прикладних протоколів Інтернету речей.
   4. Програмне забезпечення та сервіси для моделі IC.

6. Дата видачі завдання: 19 жовтня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	19.10 – 25.10.23	
2	Аналіз перспектив стандартизації IoT	26.10 – 02.11.23	
3	Дослідження стандартів по забезпеченню безпеки промислових IoT	03.11 – 11.11.23	
4	Аналіз IoT-протоколів	12.11 – 16.11.23	
5	Дослідження протоколів CoAP, MQTT	17.11 – 20.11.23	
6	Реалізація інформаційної системи	21.11 – 12.12.23	
7	Оформлення роботи: вступ, висновки, реферат	13.12 – 18.12.23	
8	Розробка демонстраційних матеріалів	19.12 – 28.12.23	

Здобувач вищої освіти _____ Олексій МАРКІН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ Оксана ТКАЛЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 84 стор., 39 рис., 14 табл., 23 джерела.

Мета роботи – дослідити стандарти та протоколи Інтернету речей.

Об'єкт дослідження – процес стандартизації в області Інтернету речей на міжнародному, регіональному та національному рівнях.

Предмет дослідження – протоколи та стандарти IoT.

Короткий зміст роботи: У роботі проаналізовано процеси стандартизації в області Інтернету речей на міжнародному, регіональному та національному рівнях, принципи роботи протоколів передавання даних для IoT. Розроблено модель, яка надає можливість контролювати дані зі сторонніх сервісів в режимі реального часу. Досліджено та розглянуто протоколи CoAP, MQTT. Розглянуто актуальні галузі впровадження та використання протоколу MQTT у сфері Інтернету речей. Проведено порівняння брокерів MQTT, їх можливостей, клієнтських бібліотек. Проаналізовано роботу кожного модуля, що використовується у роботі протоколу MQTT та досліджено всі можливі фази передавання даних від брокера до клієнта.

КЛЮЧОВІ СЛОВА: ПРОТОКОЛ MQTT, СЕРВЕР, КЛІЄНТ, БАЗА ДАНИХ, ПЛАТФОРМА ІНТЕРНЕТУ РЕЧЕЙ, БРОКЕР, МОДЕЛЬ, СЕРЕДОВИЩЕ Node-RED, ПАРАМЕТРИ, ОПТИМІЗАЦІЯ, ТОПІК.

ABSTRACT

Text part of the master`s qualification work: 84 pages, 39 pictures, 14 table, 23 sources.

The purpose of the work is to investigate the standards and protocols of the Internet of Things.

Object of research is the process of standardization in the field of the Internet of Things at the international, regional and national levels.

Subject of research is IoT protocols and standards.

Summary of the work: The work analyzed the processes of standardization in the field of the Internet of Things at the international, regional and national levels, the principles of data transmission protocols for IoT. A model was developed that provides an opportunity to monitor data from third-party services in real time. CoAP, MQTT protocols were investigated and considered. Current areas of implementation and use of the MQTT protocol in the field of the Internet of Things were considered. A comparison of MQTT brokers, their capabilities, and client libraries was made. The operation of each module used in the operation of the MQTT protocol was analyzed and all possible phases of data transmission from the broker to the client were investigated.

KEYWORDS: MQTT PROTOCOL, SERVER, CLIENT, DATABASE, INTERNET OF THINGS PLATFORM, BROKER, MODEL, NODE-RED ENVIRONMENT, PARAMETERS, OPTIMIZATION, TOPIC.

ЗМІСТ

ВСТУП

Актуальність теми: Стандарти та протоколи визначають спосіб взаємодії між різними пристроями та системами IoT. Це сприяє створенню єдиного стандарту, який дозволяє різним пристроям розуміти один одного та ефективно взаємодіяти, навіть якщо вони вироблені різними виробниками. Застосування стандартів дозволяє легше масштабувати IoT-системи. Зростання кількості пристроїв та систем вимагає стандартизації, щоб забезпечити ефективне та стабільне функціонування. Стандарти та протоколи допомагають забезпечити безпеку в IoT-системах. Визначення загальних стандартів безпеки сприяє уникненню вразливостей та захищає дані, які обмінюються між пристроями та системами.

Використання стандартів полегшує розробку та впровадження нових пристроїв та систем, оскільки розробники можуть використовувати вже існуючі стандартизовані рішення, не створюючи все з нуля. Стандарти дозволяють різним компаніям та розробникам працювати разом над інноваційними проектами. Вони надають загальну основу, на якій можна будувати нові ініціативи та розробки. Використання стандартів управління мережею спрощує контроль та підтримку IoT-систем. Це може включати у себе засоби моніторингу, діагностики та керування пристроями.

Огляд та дотримання стандартів та протоколів є важливим етапом для створення ефективних, надійних та безпечних IoT-систем, які можуть широко використовуватися у різних галузях та застосуваннях. Тому тема роботи є актуальною.

Метою роботи є дослідження стандартів та протоколів Інтернету речей.

Для досягнення поставленої мети необхідно *виконати наступні завдання*:

- дослідити стандарти та протоколи IoT, перспективи стандартизації IoT у міжнародних організаціях зв’язку;
- розглянути актуальні галузі впровадження та використання протоколів CoAP, MQTT у сфері Інтернету речей;

- розробити модель, яка надає можливість контролювати дані із сторонніх сервісів в режимі реального часу;
- провести порівняння брокерів MQTT, їх можливостей;
- проаналізувати роботу кожного модуля, що використовується у роботі протоколу MQTT.

Об'єкт дослідження – процес стандартизації в області Інтернету речей на міжнародному, регіональному та національному рівнях.

Предметом дослідження є протоколи та стандарти IoT.

Апробація результатів магістерської роботи:

Маркін О.С. «Технологія SIGFOX». Тези доповіді на І Всеукраїнській науково-технічній конференції «Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і Світу». – Київ, 28 листопада 2023 року.

РОЗДІЛ 1. СТАНДАРТИЗАЦІЯ В ОБЛАСТІ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ. СТАНДАРТИЗАЦІЯ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Стандартизація у сфері інформаційних технологій

Сучасне поняття інформаційних технологій (ІТ) визначається як сукупність методів, виробничих процесів і програмно-технічних засобів, об'єднаних у технологічний ланцюжок з метою збору, обробки, зберігання, розповсюдження і використання інформації в інтересах її користувача.

Стандарти в області ІТ містять вимоги до засобів обчислювальної техніки і мереж, інформаційного забезпечення та баз даних, програмного забезпечення, інформаційних систем. До них відносяться стандарти життєвого циклу, взаємозв'язку відкритих систем, середовища відкритих систем, а також стандарти на документацію програмного забезпечення і сферу безпеки ІТ.

Мережний протокол – набір правил, який дозволяє здійснювати з’єднання та обмін даними між двома та більше включеними в мережу комп’ютерами.

Internet Protocol (IP, досл. «міжмережний протокол») — маршрутизуємий протокол мережного рівня стеку TCP/IP. Саме IP став тим протоколом, який об’єднав окремі комп’ютерні мережі у всесвітню мережу Інтернет.

Інтернет речей *(Internet of Things, скорочено IoT)* - це глобальна мережа підключених до Інтернету фізичних пристроїв - «речей», оснащених сенсорами, датчиками і пристроями передавання інформації. Ці пристрої об'єднані за допомогою підключення до центрів контролю, управління і обробки інформації.

Ще одне визначення IoT — це мережа, що складається із взаємозв’язаних фізичних об’єктів (речей) або пристроїв, які мають вбудовані датчики, а також програмне забезпечення, що дозволяє здійснювати передавання та обмін даними між фізичним світом і комп’ютерними системами за допомогою використання стандартних протоколів зв’язку. Крім датчикі, мережа може мати виконавчі пристрої, вбудовані у фізичні об’єкти і пов’язані між собою через проводові і безпроводові мережі. Ці взаємопов’язані об’єкти (речі) мають можливість

зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

Щоб стати *"річчю",* будь-який пристрій повинен відповідати визначеним критеріям:

- пристрій повинен посилати деякі сенсорні дані, наприклад, тиск, температураабо вологість;
- пристрій повинен мати унікальний ідентифікатор, щоб його можна було ідентифікувати в процесі спілкуванні;
- пристрій повинен бути здатним обмінюватися інформацією з подібними собігаджетами, а також мати доступ до проводового інтернету та Wi-Fi.

Сьогодні технології Інтернету речей активно впроваджуються в усі сфери життя суспільства, дозволяючи використовувати різні пристрої, не обов'язково фізичні, для створення конкретних рішень, здатних полегшити життя людства. Пристрої стають здатними чути, бачити, думати, в деяких випадках діяти. Для правильної і ефективної роботи пристрої повинні коректно спілкуватися і координувати свої дії з іншими для того, щоб приймати рішення, які можуть бути настільки критичні, як порятунок життів або будівель. Технології розподілених обчислень, вбудовані датчики, сучасні безпроводові технології дозволяють Інтернету речей виконувати поставлені завдання. Однак беручи до уваги сьогоднішню різноманітність даних технологій і пристроїв, величезна кількість виробників, виникає безліч проблем їх взаємодії і необхідність в створенні і прийнятті спеціалізованих *стандартів і протоколів* зв'язку.

В області ІТ найбільш значущі з точки зору практики стандарти публікуються наступними організаціями.

Інститут інженерів з електротехніки та електроніки (IEEE, www.ieee.org) протягом багатьох років залишається провідною науково-технічною організацією, в тому числі, у створенні стандартів документації програмного забезпечення.Більшість стандартів розроблені різними комітетами, що складаються з досвідчених і відповідальних інженерів-професіоналів. Деякі із

стандартів IEEE стали також стандартами ANSI (American National Standards Institute).

Міжнародна організація по стандартизації (ISO) має великий вплив у всьому світі, особливо серед організацій виробників, що мають справу з Євросоюзом (ЄС). На теперішній час практично фактично всі сучасні стандарти в області IT - це стандарти, підготовлені ISO спільно з міжнародною електротехнічною комісією - МЕК (IEC). Особлива увага приділяється питанням забезпечення якості продукції на міжнародному рівні (ISO 9000 - серія стандартів, що регламентують управління якістю (менеджмент якості) на підприємствах) - обов'язкова умова для отримання держзамовлення.

Інститут технологій розробки програмного забезпечення (Software Engineering Institute - SEI, sei.cmu.edu) був заснований Міністерством оборони США в університеті Карнегі-Меллон для підняття рівня технології програмного забезпечення у підрядників Міністерства оборони. Робота SEI також була прийнята багатьма комерційними компаніями, які вважають поліпшення процесу розробки програмного забезпечення своїм стратегічним корпоративним завданням.

Консорціум по технології маніпулювання об'єктами (Object Management Group, www.omg.org) є некомерційною організацією, в яку в якості членів входять близько 700 компаній. OMG встановлює стандарти для розподілених об'єктно- орієнтованих обчислень. Слід зазначити, що OMG використовує уніфіковану мову моделювання UML в якості свого стандарту для опису проєктів.

1.2 Перспективи стандартизації Інтернету речей у міжнародних організаціях зв’язку

На теперішній час спостерігається активне зростання як попиту на послуги Інтернету речей (IoT) з боку індивідуальних і бізнес-споживачів, так і пропозицій з боку виробників обладнання і сервіс-провайдерів послуг IoT. Він

підтримується процесами стандартизації в області Інтернету речей на міжнародному, регіональному та національному рівнях у вигляді розробок різних рекомендацій, технічних специфікацій і нормативно-правових актів, що визначають вимоги до обладнання, додатків, мереж та послуг IoT, а також до мереж доступу.

Основними організаціями, залученими до стандартизації IoT на глобальному рівні, є:

- *Сектор стандартизації Міжнародного союзу електрозв'язку (МСЕ-Т),* в рамках якого у 2015 році була створена нова дослідницька комісія - ДК20 "IoT та його додатки, включаючи "розумні" міста і спільноти (SC&C)";

- *Партнерський проект oneM2M*, який стартував у 2012 році за ініціативою шести регіональних органів стандартизації (ETSI, ARIB, TTA, CCSA, TIA і TTC), які створили Партнерський проект 3GPP, і американської асоціації ATIS;

- *Партнерський проект 3GPP (SA, RAN, GERAN),* що займається розвитком мереж мобільного зв'язку під потреби мереж, послуг і пристроїв IoT/M2M (Machine-to-Machine).

На регіональному європейському рівні питаннями стандартизації мереж і послуг Інтернету речей займається *Європейський інститут стандартизації електрозв'язку (ETSI)*, в якому створено спеціальний технічний комітет SmartM2M.

Діяльність МСЕ-Т

Дослідження в області Інтернет речей були розпочаті у Секторі стандартизації МСЕ в Дослідницькій комісії ДК13 ("Нове покоління мереж"), а потім у 2015 році були передані у вище згадану ДК20 SC&C, яка є відповідальною за міжнародні стандарти, що забезпечують скоординований розвиток технологій IoT, включаючи міжмашинні комунікації та всеохоплюючі сенсорні мережі. ДК20 на теперішній час відкрила 57 питань з IoT на дослідний період 2017-2020 років. До складу ДК20 входять дві робочі групи (рис.1.1):

WP1: Інтернет речей;

WP2: "Розумні" міста і спільноти.

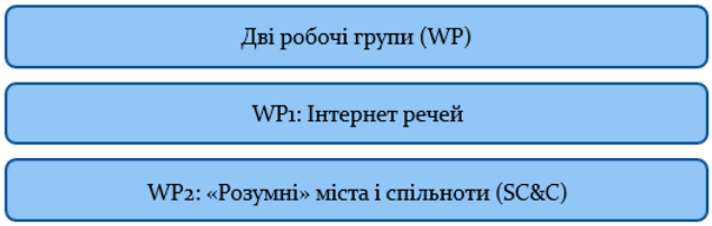


Рис. 1.1 Структура та області відповідальності ДК20 МСЕ-Т

Першими двома Рекомендаціями МСЕ-Т, розробленими ДК20, стали схвалені МСЕ в березні 2016 року:

- Y.4702 (ex. Y.IoT-DM-reqts) "Загальні вимоги і можливості керування пристроями в Інтернеті речей";
- Y.4553 (ex. Y.IoT-SPSN) "Вимоги до смартфонів, як вузлів споживання додатків та послуг Інтернету речей".

Крім того, для досліджень питань розвитку мереж і послуг M2M у квітні 2012 року відповідно до Рекомендації МСЕ-Т A7 у складі Сектору стандартизації була створена фокус-група FG з питань сервісного рівня M2M (Focus Group on M2M service layer). Її цілями були визначення мінімального набору загальних вимог M2M для вертикальних ринків, з орієнтацією в першу чергу на ринок охорони здоров'я; прикладні програмовані інтерфейси (API) і протоколи з підтримкою додатків і послуг електронної охорони здоров'я, а також підготовка технічних звітів у цих сферах. В кінці 2013 року FG M2M була закрита і питання IoT/M2M були передані до ДК20.

Останні за часом збори ДК20 проходили з 25 липня по 5 серпня 2016 року у Женеві. В процесі їх роботи на засіданнях WP1 і WP2 були розглянуті та погоджені проекти наступних нових Рекомендацій:

- Y.4113 (ex. Y.IoT-network-reqts) "Вимоги до мереж Інтернету речей" (питанняQ2/20);
- Y.4451 (ex. Y.IoT-cdn) "Структура організації мережі пристроїв з обмеженнями в середовищі Інтернету речей" (питання Q3/20);
- Y.4452 (ex. Y.WoO-fw) "Функціональна структура веб-об'єктів" (питання

Q4/ 20);

- Y.4453 (ex. Y.IoT-ASF) "Структура адаптивного програмного забезпечення для пристроїв Інтернету речей" (питання Q4/20);

- Y.4454 (ex. Y.SC-platform) "Платформа між мережної взаємодії для "розумних "міст" (питання Q6/20);

- Серії від Y.Suppl.42 до Y.4100 (ex Y.UCS-usecase) "Сценарії використання клієнтоцентричного простору (UCS)" (питання Q2/20).

Консультативна група по стандартизації електрозв'язку TSAG схвалила назви нових Рекомендацій серії Y, розробляємих ДК20.

Діяльність Партнерського проекту oneM2M

При створенні Партнерської проекту oneM2M у 2012 році передбачалося, що в сферу його діяльності будуть входити виключно питання, які пов'язані із стандартизацією технологій та послуг M2M. Структура Партнерського проекту oneM2M включає декілька робочих груп.

У січні 2015 року було закінчено і опубліковано Реліз 1 oneM2M, що включав 12 технічних специфікацій з вимогами до системної архітектурі мереж M2M. У січні 2016 року було ратифіковано уточнений Реліз 1, який фактично можна трактувати як Реліз 3 ETSI, але який не має сумісності з Технічними специфікаціями oneM2M.

У липні 2016 року на пленарному засіданні TP25 oneM2M, що відбулося в США, було оголошено про завершення роботи над Релізом 2 oneM2M, що включає 17 технічних специфікацій, які встановлюють вимоги до системної архітектурі мереж M2M, безпеки мереж M2M у ланцюжку E2E для будь-яких пристроїв і серверів. На TP25 oneM2M була відзначена важлива роль промислових застосувань в IoT і представлені плани розробки 14 нових додаткових технічних специфікацій, що розширюють дію Релізу 2 oneM2M на сферу IoT. Ці нові специфікації oneM2M призначені для розширення можливостей міжмережної взаємодії мереж IoT і M2M, а також створення безшовних мереж IoT у різних секторах промисловості та сфери послуг, у тому числі автомобільному, медицини, "розумних" будинків і міст.

Звіт про діяльність Партнерського проекту oneM2M, представлений пленарним зборам TP25, показує, що з 228 членів проекту більша частина учасників - це члени ETSI - 122 (більше 50% від одного з семи головних партнерів oneM2M), що свідчить про істотну інноваційну активність європейської частини проекту стандартизації M2M.

Діяльність Партнерського проекту 3GPP

Перші дослідження 3GPP по стандартизації рішень в області мереж і послуг M2M були розпочаті в рамках розробки Релізу 10. У документах Релізу 10 були сформульовані загальні вимоги до послуг M2M, підтримуваним мережами доступу на основі технології LTE Advanced, які сфокусовані на забезпеченні контролю і управління перевантаженнями трафіку M2M-додатків у мережі доступу.

Подальший розвиток технічних вимог до мереж доступу і послуг M2M в Релізі 11 3GPP призвело до розробки нового функціонального елементу MTC-IWF архітектури мережі LTE, що виконує функцію міжмережної взаємодії між мережею M2M і мережею доступу. Були розширені загальні вимоги до підтримуваним мережею LTE послуг M2M, сформульовані вимоги до функції виведення M2M-пристрою із сплячого режиму (пробудження), введені вимоги використання IP-адресації для M2M-пристроїв замість E.164, що розширюють загальний адресний простір, а також підтримці M2M - пристроїв, що працюють тільки з пакетною комутацією.

У Релізі 12 основною метою робіт 3GPP з розглянутих питань було створення недорогих абонентських пристроїв M2M на базі технології LTE - пристроїв класу 1. У цьому релізі додатково розширені вимоги до M2M-послуг, сформульовані вимоги до функцій групової доставки повідомлень, групових обмежень, групової тарифікації, функціям передавання малого обсягу даних і відправки повідомлень з низькою активністю, до моніторингу подій, до низької мобільності і до функцій контролю часу передавання та прийому даних, а також вимоги щодо зниження енергоспоживання пристроїв M2M.

Робочі групи Партнерського проекту 3GPP у 2015 році завершили

дослідження з оптимізації технології LTE (TR 36.888) з метою забезпечення створення рішень, конкурентних за вартістю з M2M-пристроями на основі стандарту GSM (2G).

На теперішній час Партнерський проект 3GPP стандартизує три окремі технології для частотних діапазонів з ліцензованим використанням спектру:

- LTE-M - як еволюцію технології LTE, яка оптимізована під IoT в мережі RAN. Перша версія з'явилася в релізі 12 в IV кварталі 2014 року, в подальшому вона оптимізована в релізі 13. Повна специфікація LTE-M з'явилася в I кварталі 2016 року. Для цієї технології використовується канал шириною 1,4 МГц;

- EC-GSM (з розширеним GSM-покриттям) - технологія представляє собою еволюційний підхід, стандартизована для вирішення GERAN в Релізі 13. Повна її специфікація з'явилася також у I кварталі 2016 року;

- Clean Slate Cellular IoT (або NB-IoT/Narrow Band Internet of Things) - технологія нового вузькосмугового радіоінтерфейсу, яка створюється в рамках розвитку RAN (Реліз 13). Стандартизація розпочата у IV кварталі 2015 року. Дане рішення на основі вузькосмугових FDMA-сигналів в лінії вгору і вузькосмуговогоOFDMA-сигналу в лінії вниз використовує канали шириною 200 кГц.

Найбільш перспективні економічні і технічні параметри забезпечує технологія NB-IoT. Специфікаціями 3GPP в Релізі 13 для технології радіодоступу NB-IoT, що розроблена спеціально для підключення абонентських пристроїв Інтернету речей до стільникових мереж технологій LTE (LTE-Advanced, LTE Pro), була реалізована можливість поєднувати такі характеристики LTE, як сигнали OFDM, дальність зв'язку і великий термін служби акумуляторів. Бюджет радіолінії NB- IoT був поліпшений на 20 дБ у порівнянні з LTE-Advanced. У лінії вгору технологія NB-IoT для передавання невеликих обсягів даних дозволяє абонентським пристроям працювати паралельно. Очікується, що технологія NB-IoT завдяки зниженню витрат, збільшення покриття і продовження терміну служби акумуляторів підключених пристроїв допоможе приєднати мільярди абонентських пристроїв Інтернету речей.

Діяльність Технічного комітету SmartM2ETSI

На європейському економічному і технологічному просторі стандартизація мережевих рішень, послуг і бізнес-моделей M2M здійснюється через інститут ETSI, у складі якого в 2009 році був створений спеціальний Технічний комітет M2M/ETSI. Технічним комітетом M2M ETSI завершена розробка двох Релізів - 1 і2 - стандартів, що визначають базові вимоги до мереж M2M.

Після створення міжнародного проекту oneM2M технічний комітет M2M/ ETSI втратив частину своїх повноважень, але в той же час отримав і ряд додаткових. В результаті правлінням ETSI було прийняте рішення зберегти комітет з урахуванням нових повноважень, спрямованих на реалізацію директив ЄС з питань M2M та IoT (Smart Grid, Smart metering, eCall та ін.), і назвати перетворенуструктуру Технічним комітетом SmartM2M.

Для стандартизації вимог до мереж і послуг Інтернету речей в Технічному комітеті SmartM2M ETSI в листопаді 2015 року було створено спеціальну групу STF505. Вона вже провела шість засідань з підготовки технічного звіту SmartM2M про розробку стандартів Інтернету речей на основі аналізу існуючих досягнень міжнародних організацій та індустріальних альянсів в галузі стандартизації IoT.

Крім цього, комітетом SmartM2M вирішуються питання проведення семінару з безпеки IoT та оцінки Європейського розриву у стандартизації технологій Інтернету речей. Для цього групою STF505 проводяться дослідження, результати яких увійдуть у два технічних звіти: TR 103 375 "Стандарти IoT і майбутня еволюція" і TR 103 376 "Сценарії використання IoT при реалізації великомасштабних пілотних проектів і розрив у стандартизації".

Сорокове, останнє, засідання Технічного комітету SmartM2M (28-30 листопада 2016 року, м. Софія Антиполіс) було присвячено стандартизації питань онтології пристроїв IoT та M2M, що відносяться до класу пристроїв (appliances), яка в розумінні SmartM2M визначає формальну специфікацію концептуалізації прямого використання семантики для визначеної реальності.

Засідання включало розгляд проектів наступних стандартів та звітів:

- ETSI TS 103 268 "Онтологія "розумних" пристроїв споживчого сегменту (appliances) і структура тестування зв'язку;

- ETSI TS 103 410 "Розширення розумних пристроїв до еталонної онтології пристроїв споживчого сегменту SAREF; Частина 1: Енергетичний домен";

- ETSI TR 103 410 "Дослідження розширення розумних пристроїв з еталонною онтологією SAREF".

Онтологія SAREF (Semantic Annotations for the Internet of Things) використовується в якості еталонної (референсної) для пристроїв IoT та M2M, що відносяться до класу пристроїв споживчого сегменту (appliances), що використовуються в побутовій електроніці на рівні домашнього або офісного навколишнього середовища. SAREF - це онтологія, розроблена з метою створення стандартизованого та семантично визначеного опису об'єктів інтернету речей (IoT). Онтологія визначає способи взаємодії між різними пристроями та системами в IoT за допомогою стандартних семантичних анотацій. SAREF визначає концепції та відносини між різними об'єктами, які можуть бути присутні в сенсорах, пристроях, додатках та системах IoT. Онтологія дозволяє пристроям та системам IoT розуміти одне одного, використовуючи спільну мову для опису своїх можливостей та функцій.

Отже, зусилля світового телекомунікаційного співтовариства спрямовані на розвиток нових сегментів ІКТ-ринку, пов'язаних з Інтернетом речей і міжмашинних комунікацій M2M. Нормативно-технологічною базою цього розвитку в умовах подальшої лібералізації ринків стають відкриті стандарти, що створюються на глобальному і регіональному рівнях. Основними організаціями, залученими у стандартизацію IoT та M2M на глобальному рівні, є: Сектор стандартизації МСЕ-Т (ДК20), Партнерський проект oneM2M, Партнерський проект 3GPP, а на європейському - Європейський інститут стандартизації електрозв'язку. Ці організації на основі розроблюваних стандартів забезпечують єдність технічної політики і технологічних рішень, підвищуючи можливості конкуренції виробників на етапі розробки стандартів.

Таблиця 1.1

Характеристики технологій для IoT/M2M та порівняння технологій

Характеристики	LTE-M (1,4 МГц)	NB-IoT (200 кГц)	EC-GSM (200 кГц)
Покращене покриття, включаючи всередині будівель	156 дБ MCL (покращення + 15 дБ)	164 дБ MCL (покращення + 20 дБ)	164 дБ MCL (покращення + 20 дБ)
Радіус дії (пряма видимість)	< 11 км	< 15 км	< 15 км
Ємність при масовому застосуванні	> 52 тис. АТ/сота/180 кГц	> 52 тис. АТ/сота/180 кГц	> 52 тис. АТ/сота/180 кГц
Швидкість передавання даних	< 1 Мбіт/с	< 200 кбіт/с	< 70 кбіт/с
Термін автономної роботи	> 10 років	> 10 років	> 10 років
Вартість IoT-модуля	5.0 $ (2016) / 3 $ (2022)	4.0 $ (2016) / 2-3 $ (2022)	5.5 $ (2016) / 2.5 $ (2022)
Сценарій використання спектру	У смузі ліцензованих частот 3GPP (In-band)	Три сценарії (In-band, Stand alone, Guard-band)	У смузі рефармингу ліцензованих частот 3GPP (Stand alone)
Необхідність оновлення мережі	Буде визначена	Так (HW/SW)	Так (HW/SW)

1.3 Стандарти по забезпеченню безпеки споживчих та промислових IoT

Із зростанням кількості пристроїв, що підключаються, збільшуються і проблеми в сфері їх безпеки, і ці проблеми ще в більшій мірі стосуються пристроїв, які не відповідають прийнятим стандартам. Є три основних побоювання, які пов'язані з використанням IoT:

• Всеохоплюючий збір даних.

• Потенціал несподіваного використання даних, призначених для користувача.

• Підвищені ризики безпеки.

Всі ці проблеми у сфері безпеки є результатом гонки на ринку: кожний виробник IoT пристроїв поспішає вийти на ринок, не думаючи про безпеку.

В якості прикладу типової проблеми з безпекою можна назвати ситуацію, коли компанії підключають у свої мережі незахищені пристрої, а потім не можуть оновити програмне забезпечення. Крім того, до мережі IoT підключається багато застарілих пристроїв, які не мають необхідного захисту

від зовнішніх загроз.

Стандарт *TS 103 645* забезпечує базовий рівень безпеки для підключених до інтернету споживчих продуктів, а також основу для майбутніх схем сертифікації IoT. TS 103 645 включає положення щодо забезпечення безпеки підключених до інтернету споживчих пристроїв і пов'язаних з ними сервісів.

Вимоги нового стандарту включають відмову від використання розповсюджених дефолтних паролів, які неодноразово ставали джерелом значної кількості проблем безпеки, а також реалізацію виробниками політики розкриття інформації про вразливості для надання експертам в галузі безпеки можливості повідомити про наявність тих чи інших проблем.

Крім того, стандарт дозволить забезпечити дотримання норм Загального регламенту із захисту даних (General Data Protection Regulation, GDPR) IoT-пристроями та сервісами, які зберігають і обробляють персональні дані користувачів.

Для забезпечення безпеки «розумного» виробничого обладнання потрібні «розумні» технології. Смарт-пристрої постійно перебувають під загрозою кібератак, і не тільки «Інтернет речей» (IoT) в цілому, а й окремі системи. У зв'язку з цим Міжнародна організація по стандартизації (ISO) розробила новий стандарт, покликаний посилити безпеку промислового IoT-обладнання.

Список загроз включає в себе навмисне збільшення швидкості роботи виробничого смарт-обладнання до критичного рівня і зниження температури термічної обробки продуктів харчування, що може призвести до розмноження хвороботворних мікроорганізмів. Тобто, кібератаки на промислові IoT-пристрої не тільки загрожують фінансовими втратами через вимушену зупинку виробництва, але також можуть завдати шкоди здоров'ю людини.

З метою посилення безпеки промислового IoT-обладнання ISO представила новий технічний стандарт *ISO/TR 22100-4* «Безпека виробничого обладнання - Зв'язок з ISO 12100 - Частина 4: Керівництво для виробників обладнання по розгляду відповідних аспектів інформаційної безпеки (кібербезпеки)».

Як зрозуміло з назви, новий стандарт доповнює вже існуючий стандарт

безпеки промислового обладнання ISO 12100 «Безпека виробничого обладнання - Основні принципи проектування - Оцінка і зниження ризиків».

Новий стандарт покликаний «опрацювати аспекти безпеки обладнання, яке може бути порушене кібератаками, пов'язаними з безпосереднім або віддаленим доступом та маніпуляціями з системами управління безпекою, що здійснювалися зі злим умислом (не за призначенням)». Для цієї мети стандарт містить вказівки з таких ключових аспектів ІБ, як шифрування і аутентифікація.

Стандарти ISO/IEC 29192

Серія стандартів *ISO/IEC 29192* містить характеристики легкої криптографії, яка ідеально підходить для малопотужних простих пристроїв. У ситуації з лампочками ізраїльські дослідники рекомендували особливу методику захисту даних, описану в ISO/IEC 29192-5, яка включає три хеш-функції, які підходять для додатків, що вимагають легких криптографічних реалізацій. Але, як і в будь- якій сфері, що розвивається, потрібними будуть нові стандарти.

На сьогоднішній день було опубліковано технічний звіт ISO/IEC TR 22417, Інформаційна технологія, випадки використання Інтернету речей, який містить інформацію для користувачів Інтернету речей. Керівництво містить такі важливі питання, як основні вимоги, сумісність і стандарти, що застосовуються користувачами. Наведені приклади показують, в якому випадку існуючі стандарти відіграють визначену роль, і наголошують про необхідність подальшої роботи із стандартизації.

Стандарт ISO/IEC 20924

Ще одним основним стандартом є *ISO/IEC 20924, Інформаційні технології. Інтернет речей (IoT). Словник.* ISO/IEC 20924 та ISO/IEC 30141 формують єдину мову.

Робочу групу, яка займалася розробкою ISO/IEC 30141, очолював д-р Цзе Шен (Dr Jie Shen) з Китаю за підтримки двох співредакторів, таких як Wei Wei

(Вей Вей) з Німеччини і Остен Фронберг (Östen Frånberg) зі Швеції. У сукупності керівники проектів володіють багаторічним досвідом роботи в цій галузі діяльності, який був поглиблений 50 іншими фахівцями, які внесли безпосередній внесок у розробку стандарту.

Більша частина деталей вже міститься у багатьох стандартах, опублікованих підкомітетами СТК 1, а ISO/IEC 30141 містить посилання, щоб об'єднати їх усіх з декількома новими стандартами, що розробляються СТК 1/ПК 41.

ISO/IEC 30141 містить загальну основу для дизайнерів і розробників Інтернету речей. Стандарт містить характеристики Інтернету речей поряд з концептуальною моделлю та еталонною архітектурою. Описи супроводжуються численними прикладами.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ЗАСОБІВ ВЗАЄМОДІЇ ПРОТОКОЛІВ IoT

2.1 Аналіз предметної області

Не тільки люди повинні генерувати або фіксувати та створювати дані, але комп'ютери та інші вбудовані пристрої повинні мати можливість збирати власну інформацію, взаємодіючи з їхнім внутрішнім станом або зовнішнім середовищем. Впровадження IoT, інших пристроїв або «речей» розширює традиційний Інтернет, роблячи мережні з'єднання більш актуальними та цінними, ніж будь-коли раніше, а також додати зовсім нове значення галузі інформаційних і комунікаційних технологій. Загалом, IoT можна назвати більш трансформаційним, ніж традиційний Інтернет, який він матиме вплив на спосіб життя людей.

IoT — це мережа фізичних об'єктів або «речей», які спілкуються один з одним. Вони оснащені електронікою, датчиками та виконавчими механізмами з обчислювальною потужністю, програмним забезпеченням і підключенням до мережі, що дозволяє користувачам стати його невід'ємною частиною. IoT пройшов через багато розробок і міркувань з різнимивизначеннями на основі Інтернету. Доктор Овідіу Вермезан і компанія у своїй роботі описали цей термін, розглядаючи більшу роботу Інтернету як Інтернет енергії (IoE), Інтернет медіа (IoM), Інтернет людей (IoP) та Інтернет послуг (IoS). Cisco вирішила ввести цей термін як Інтернет усього, де він розглядався як система, що складається з речей, де процеси, дані та люди разом утворили «Мережу мереж». На думку Cisco, IoT з'єднає людей, процеси, дані та «речі» разом, щоб сформувати мережу, придатну та корисну для допомоги у відстеженні «речей», а також для вирішення деяких глобальних викликів, таких як посуха, зміна клімату, джерела або питної води, і голод.

IoT швидко розширюється і сфери застосування включають розумні міста, розумну воду, розумне вимірювання, безпеку та надзвичайні ситуації, роздрібну торгівлю, логістику, промисловий контроль, розумне сільське господарство, розумне тваринництво, домашнє господарство. Однак, оскільки області

застосування охоплюють різні середовища, а задіяні пристрої різноманітні, це робить IoT дуже різнорідних і, отже, викликів і бар'єрів, таких як зв'язок, управління живленням, складність, швидкий розвиток, безпека та якість обслуговування, які завжди пов'язані зі стандартними проблемами бездротової сенсорної мережі (WSN), були перераховані Чейзом як перешкода розвитку IoT. Інші проблеми, на які Губбі та його колеги відзначили конфіденційність, зондування за участю, аналітику даних, візуалізацію на основі географічної інформаційної системи (ГІС) і хмарні обчислення. Крім того, проблема підключення IoT також супроводжується проблемами архітектури та протоколів.

Сучасні виробники промислового обладнання стикаються якщо не з усіма, то з більшістю вищезгаданих проблем під час підготовки продуктів для IoT. Таким чином, щоб IoT успішно працював і відповідав прогнозованому об'єму пристроїв, підключених до Інтернету до 2023 року, аналіз показує, що йому потрібно будувативідкриті, гнучкі апаратні, програмні та мережеві платформи, які здатні розвиватися та адаптуватися. Однак у цій дипломній роботі розглядаються проблеми протоколів шлюзів підключення IoT та їх взаємодію.

З тих пір, як галузі почали підключати практично всі пристрої та «речі» від сміттєвих баків до термостата для збору даних у режимі реального часу, сьогодні підприємства починають усвідомлювати, що справжня цінність Інтернету речей полягає не лише в зборі й обробці даних, а й це аналіз даних для отримання розуміння бізнесу.

Згідно з прогнозами International Data Corporation (IDC) світовий ринок IoT досягнув десяти мільярдів доларів (10065 мільярдів доларів) у 2023 році, що буде стрибком із двадцяти мільярдів доларів (20715 мільярдів доларів) у 2025 році.

На рис. 2.1 показано потенційний економічний вплив Інтернету речей до 2025 року за результатами аналізу Mc Kinsey Global Institute. Продуктивність IoT залежить від ефективності збору, передавання та обробки даних. Дані можуть передаватися безпосередньо від пристроїв до хмарних сервісів або проходити через проміжні вузли (міжмережеві шлюзи).

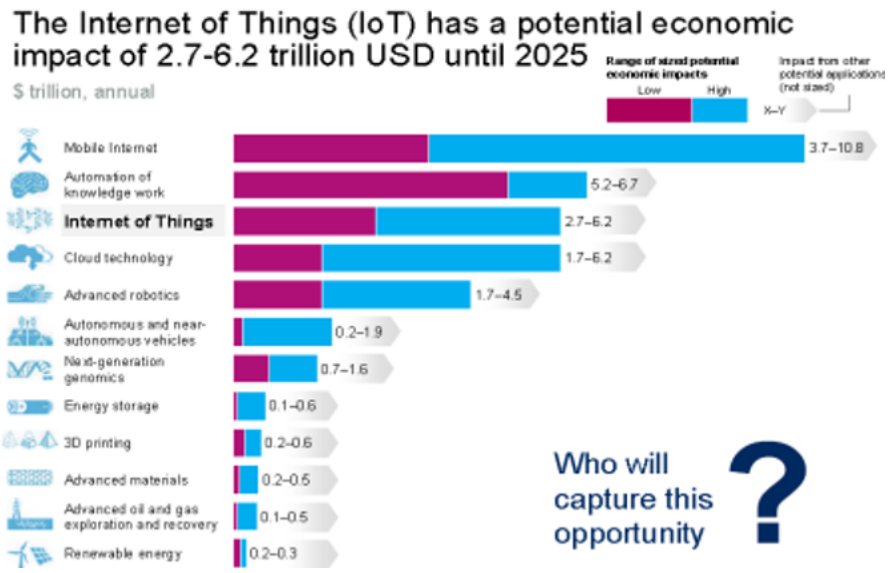


Рис 2.1 Економічний потенціал IoT

2.2 Архітектура IoT

Усвідомивши економічну важливість IoT у попередніх розділах, важливо також розглянути технологію, яка робить можливим реалізацію економічних цінностей. Архітектура Інтернету речей (IoT) визначає спосіб організації та взаємодії різних пристроїв, датчиків, програмних рішень та мереж у єдину систему, яка дозволяє збирати, обробляти і використовувати дані.

Рівень архітектури IoT має форму, подібну до еталонної моделі ISO/OSI ((ISO) & IEC).

Протокол управління передаванням (TCP) та Інтернет-протокол (IP), а також 4-рівнева модель Міністерства оборони США (DoD4). Рис.2.2 ілюструє вищезазначені моделі в архітектурі міжмережевого зв’язку.

Пристрої та датчики - це різноманітні фізичні об'єкти, які мають можливість взаємодії з навколишнім середовищем та здійснювати обмін інформацією через мережі. Мережний рівень відповідає за забезпечення з'єднання між пристроями та інфраструктурою Інтернету, включаючи бездротові та провідні технології зв'язку. Хмарний рівень та розподілені обчислення включає сервіси для зберігання та

обробки великих обсягів даних, а також надає можливості для аналізу та взаємодії з іншими системами.

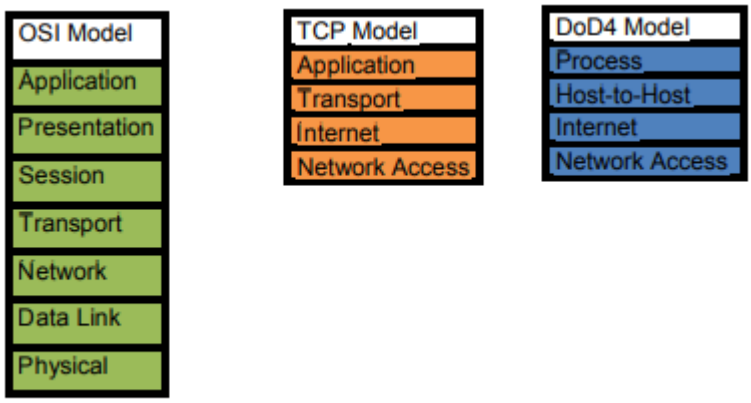


Рис.2.2 Семирівневий стек і чотирьохрівневі стеки або OSI, TCP і DoD4

З рис.2.2 видно, що Інтернет-модель (TCP) і модель DoD4 є 4-рівневими та відображаються одна на одну. Крім того, запропонована модель архітектури IoT була заснована на вищезгаданій моделі, тобто ISO, TCP і DoD4 також є 4-рівневою моделлю. Насправді, базуючись на ISO, TCP і DoD4, IoT можна було реалізувати без подальшого архітектурного моделювання, але їм не вдалося створити IoTфункції та проблеми, такі як підключення та зв'язок, збір і аналіз даних, керування пристроями, масштабованість, сумісність, інтеграція та безпека.

Таким чином, виникла необхідність реструктуризації всіх трьох моделей, щоб вони відповідали функціям і проблемам IoT. Архітектурна модель IoT складається з різних компонентів і є 4-рівневою орієнтованою архітектурою, де на кожному рівні можуть бути реалізовані певні технології. У таблиці 2 показано 4-рівневу модель IoT, яка показує, які компоненти реалізовані на кожному рівні.

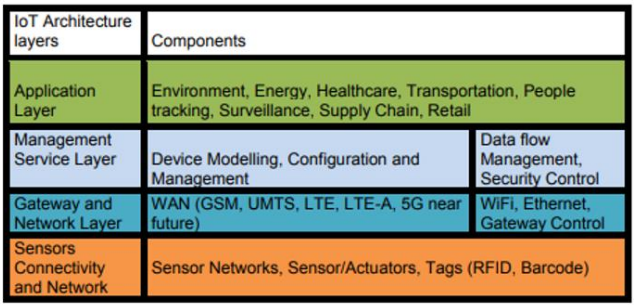


Рис.2.3 Архітектурна модель IoT

Підключення датчиків і мережа або рівень пристроїв

Рівень внизу в основному представляє пристрої IoT, і вони бувають різних типів і форм архітектури, властивостей і можливостей. Пристрій можна вважати пристроєм IoT, якщо такий пристрій має будь-яку форму зв'язку, яка може бути підключена до Інтернету прямо чи опосередковано. Рис.2.4 ілюструє деякі приклади пристроїв, які можна знайти на рівні підключення датчиків і мережі.

Пристрої на сенсорному рівні мають здатність сприймати та збирати інформацію в реальному часі для обробки. Вони мають низьку потужність і низьку швидкість передачі даних для підключення. Області застосування деяких із цих датчиків можна назвати датчиками тіла, датчиками навколишнього середовища та датчиками спостереження.

Sensors Layers	Technologies	Infographic example
LAN	WiFi, Ethernet	
PAN	Bluetooth, ZigBee, Z-Wave, 6LoWPAN, UWB, Wired	
Sensors or Actuators	Infrared, Solid State, GPS, Photoelectric, Accelerometer, Photochemistry, Catalytic, Gyroscope	
Tag	RFID and Barcode (1D, 2D)	

Рис. 2.4 Рівень сенсора IoT

Шлюз і мережний рівень

Шлюз і мережевий рівень, також відомий як комунікаційний рівень, підтримує підключення пристроїв на рівні датчика або на рівні пристрою. Він складається з різноманітних протоколів, які допомагають у спілкуванні між пристроями та хмарою. Найвідомішим із цих протоколів є протокол передачі гіпертексту (HTTP) із підхід RESTful, MessageQueueTelemetryTransport (MQTT) і ConstrainedApplicationProtocol (CoAP). Протоколи IoT будуть детальніше вивчені в наступних розділах. Рис. 2.5 показує шлюз і мережний рівень із задіяними технологіями.

Gateway Network	WAN 3G, LTE, LTE-A, M LoRa, Sigfox, future 5G	LAN WiFi, Ethernet
Gateway	Micro-Controllers, Radio Communication Module, Signal Processor, and Modulator, Access Point, Embedded/OS, SIM module Encryption.	

Рис.2.5 Шлюз Інтернету речей і мережний рівень

Одним з найважливіших аспектів шлюзу та мережевого рівня є його здатність агрегувати дані, а також розміщувати зв’язок з брокером. Посередник зв’язку та агрегації даних поєднує зв’язок і дані з різних пристроїв, а потім направляє інформацію на певний пристрій через службу шлюзу. Шлюз і мережевий рівень також здатніпідтримки, наприклад, сервера HTTP та брокера MQTT для забезпечення зв’язку між пристроями. Крім того, він служить мостом і перетворює між різними протоколами, такими як HTTP API на основі повідомлення MQTT на пристрій.

Рівень служби управління

Рівень служби управління складається з двох основних функціональних частин, як зазначено на рис.2.6. Двома основними функціональними частинами є

частина моделювання пристрою, конфігурації та управління потоком даних і контролю безпеки. Однак, перш ніж розглядати функції частин рівня служби керування, також важливо описати, що таке служба управління. На рис.1.6 зображено деякі послуги, які може запропонувати рівень обслуговування управління.

Management Service Layer	
Services	Components of the service
Operational Support System (OSS)	Device Management / Configuration / Management, Performance Management, Security Management
Service Analytic Platform	Statistical Analytics, Data Mining, Text Mining, In-Memory Analytics, Predictive Analytics
Billing Support System (BSS)	Billing Report
Security	Access Control, Encryption, Identify Access
Business Rules Management (BRM)	Rule Definition / Modelling / Simulation / Execution
Business Process Management (BPM)	Workflow Process Modelling / Simulation / Execution

Рис. 2.6 Компоненти рівня служби управління IoT

Рівень служби управління відіграє важливу роль в архітектурі IoT. Ролі можна розділити на дві частини. Управління службою даних відповідає за такі процеси, як інформаційна аналітика, контроль безпеки, моделювання процесів і управління пристроями. Управління даними має дві форми техніки: періодичні та аперіодичні схеми управління даними.

У періодичному управлінні даними IoT інформація або дані періодично збираються датчиком IoT для аналізу. Наприклад, монітор датчика температури за певний проміжок часу зафіксує ряд інформації про погоду або стан промислової машини. Однак не вся зібрана інформація буде необхідна для аналізу, отже, необхідне уточнення даних, зібраних датчиком, щоб відфільтрувати небажані та зберегти ті, які потрібні для фактичної мети збору даних.

У техніці аперіодичного збору даних датчик IoT збирає дані та вимагає негайної реакції або уваги на інформацію, щойно відбувається подія. Наприклад, якщо сенсорний пристрій IoT стежить за пацієнтом, якщо безпека контролюється,

доставка інформації має бути негайною та вимагатиме також негайна реакція. Окрім блоку управління даними, існує також блок управління даними, який забезпечує управління інформаційним потоком даних, доступом до інформації, інтеграцією та контролем даних. Існує також блок абстракції даних, який надає послуги, такі як обробка вилучення інформації, і може використовуватися як звичайний бізнес-режим.

Прикладний рівень

Рівень програми IoT є найвищим рівнем, який служить інтерфейсом між сенсорною програмою та кінцевими користувачами. Він складається з різних секторів застосування, таких як навколишнє середовище, промисловість, охорона здоров'я, розумний будинок відстеження та кілька інших, як показано в таблиці 6 нижче. Це також рівень, на якому розміщені протоколи прикладного рівня IoT, такі як протокол передачі гіпертексту (HTTP), MQTT, CoAP, розширений протокол черги повідомлень (AMQP), розширюваний протокол обміну повідомленнями та присутності (XMPP), простий протокол доступу до об'єктів (SOAP).

Крім того, оскільки різні додатки з різних галузей і секторів мають різні протоколи та класифікації на основі типу мережі, зони покриття, розміру, бізнес-моделі, систем реального часу або не реального часу, протоколи прикладного рівня можуть виділяти, з'єднувати та обмінюватися даними чи інформацією між іншими прикладними системами. Класифікація IoT базується на доменах додатків, таких як Особистий і Домашній, Корпоративний, Комунальний і Мобільний. Ці класифікації визначають розмір області застосування, а також визначають її характеристики.

Наприклад, домен програми Personal and Home представляє невеликий масштаб. Це означає обмежену кількість користувачів, окремих осіб або домівку. Корпоративний IoT представляє велику кількість користувачів на рівні спільноти. Утиліта IoT представляє набагато більший масштаб користувачів, таких як

національна чи регіональна підтримка IoT і MobileIoT, які зазвичай поширені в інших доменах через їх природу мобільності, а задіяні пристрої в основному працюють від акумуляторів і портативні. Рис.2.7 ілюструє деякі з основних доменів додатків, ринкових сфер і секторів, які може розміщувати рівень додатків.

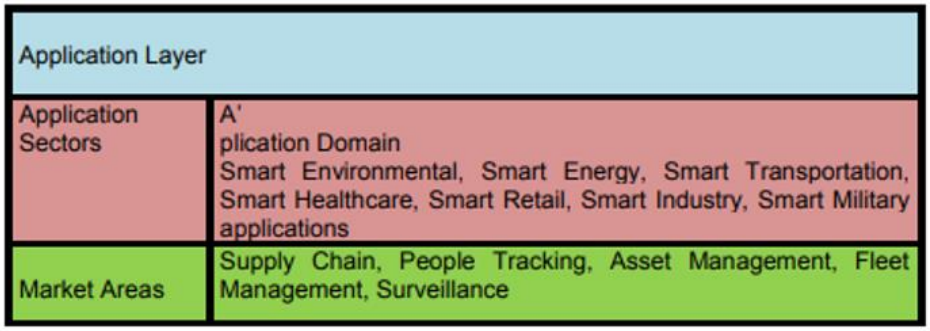


Рис. 2.7 Рівень програми IoT

2.3 Аналіз IoT-протоколів

Як і в будь-якій іншій формі зв'язку між людиною-людиною (H2H) або D2D (пристрій-пристрій), має бути протокол, який сприяє або допомагає людям чи пристроям розуміти один одного. У випадку IoT зв'язку між D2D або (Machine-to-Machine) M2M і хмарою існує широкий набір протоколів, які полегшують зв'язок. На рис. 2.8 показано стек протоколів IoT у порівнянні з моделлю ISO/OSI та стеком протоколів (TCP).

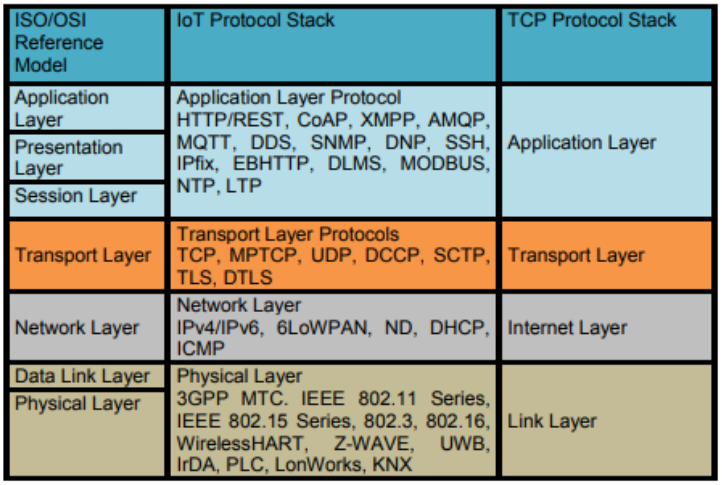


Рис. 2.8 Стек протоколів IoT

Protocol	Transport Protocol	Messaging	WAN (2G, 3G, 4G)	Power	Compute Resources	Security
HTTP/ REST	TCP	Rqst/Rspnse	Excellent	Fair	100Ks/RAM Flash	Low-Optimal
MQTT	TCP	Pub/Subsrb Rqst/Rspnse	Excellent	Good	10Ks/RAM Flash	Medium-Optimal
CoAP	UDP	Rqst/Rspnse	Excellent	Excellent	10Ks/RAM Flash	Medium-Optimal

Рис. 2.9 Протоколи прикладного рівня IoT

Протокол передачі гіпертексту (HTTP) є найбільш поширеним і популярним протоколом прикладного рівня у Всесвітній павутині. Стандартизація HTTP має було зроблено Інженерною робочою групою Інтернету (IETF) у співпраці з Консорціумом Всесвітньої павутини (W3C) (MIT). HTTP працює на основі технології обміну повідомленнями клієнт-сервер, де клієнт запитує сторінку мови гіпертекстової розмітки (HTML) із сервера, а сервер також відповідає сторінкою HTML. Як показано в таблиці 8, HTTP покладається на TCP як транспортний протокол, який використовує сокети для передачі даних. З'єднання між клієнтом і сервером починається з клієнта через сокет-з'єднання на порт 80, який є номером порту, призначеного для HTTP на сервері. Коли з'єднання встановлено, це означає, що сервер приймає запит клієнта, який знаходиться у формі сторінки HTML, та інші об'єкти.

Однак після встановлення з'єднання клієнтський браузер і веб-сервер обмінюються сторінками HTML і об'єктами. Після завершення запиту TCP припиняє з'єднання між клієнтом і сервером, а також очищає пам'ять, щоб попередні запити від клієнта були видалені.

У HTTP запити, такі як GET, PUT, POST і DELETE, є чотирма методами, які найчастіше використовуються. Запит GET відображає веб-сторінку та її об'єкти за запитом до користувача. Методи запиту PUT і POST використовуються для зміни ресурсів сервера, а запит DELETE видаляє непотрібні ресурси. Крім того, існує два типи HTTP-з'єднання, які можна встановити за допомогою TCP. Це непостійні (HTTP/1.0) і постійні (HTTP/1.1) підключення.

Постійний (HTTP/1.1) залежить від кількості TCP-з’єднань, необхідних для передачі уніфікованого покажчика ресурсу (URL) веб-сторінки та її об’єктів. На рис.2.10 показана схема HTTP-з'єднання між клієнтом і сервером.

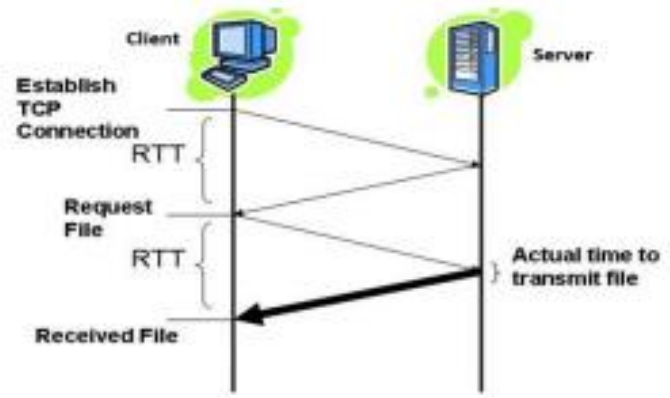


Рис.2.10 HTTP, що встановлює TCP-з’єднання між клієнтом і сервером

Round-Trip Time (RTT) — це час, витрачений на надсилання пакета від клієнта до сервера та отримання відповіді. Він також представляє час, необхідний для встановлення з’єднання TCP, надсилання запиту та отримання відповіді або отримання файлу з його час передачі. Математично загальне RTT від початку встановлення з’єднання TCP до отримання запитаного файлу можна виразити як 2 x RTT + час передавання файлу (FTT).

Передавання репрезентативного стану (REST)

REST — це архітектура програмного забезпечення, яка не залежить від мови й операційної системи, для розробки мережевих програм і розповсюдження системи HTTP для з’єднання машин. REST — це система «клієнт-сервер» без збереження стану, з можливістю кешування, «точка-точка» з уніфікованим інтерфейсом і розроблена як полегшена система. Режим зв’язку починається, коли клієнт надсилає повідомлення у формі запиту на сервер, а сервер відповідає клієнту у формі відповіді, вказуючи, чи був запит, надісланий клієнтом, успішним чи сталася помилка. За допомогою REST зв’язок між пристроями в хмарі

можливий через TCP/IP, де HTTP використовується для підключення до всесвітньої мережі (www).

2.4 Протокол CoAP

CoAP — це модельний протокол передавання документів типу «клієнт-сервер» на основі Інтернету, схожий на HTTP, і його стандартизовано в рамках IETF, обмежених середовищ RESTful (CoRE). Розроблений для обмежених пристроїв і обмежених мереж. Пристрої обмеження – це вбудовані пристрої з обмеженою потужністю, пам'яттю та ресурсами обробки, і очікується, що вони будуть підключені та функціонуватимуть подібно до основних процесів.HTTP є основним протоколом, тому що зв'язок між клієнтом і сервером занадто важкий для таких пристроїв.CoAP було розроблено для усунення обмежень, які HTTP накладає на обмежені пристрої, такі як датчики та пристрої з низьким енергоспоживанням, підключені через мережі з втратою енергії (LLN).

Модель дизайну CoAP еквівалентна моделі HTTP «клієнт-сервер», але більша частина її реалізації відбувається в рамках комунікацій M2M або D2D, і вони можуть виступати як клієнтом, так і сервером.CoAP не підтримує протокол керування передачею при транспортуванні, але працює через протокол UDP (User Datagram Protocol).Він використовує широкомовну та багатоадресну передачу UDP для адресації, а взаємодія між клієнтом і сервером є асинхронною через UDP.Однак, оскільки транспорт, орієнтований на дейтаграми, не має з'єднання, зв'язок клієнт-сервер CoAP також не з'єднується, і його можна використовувати на додаток до служб коротких повідомлень (SMS) та інших протоколів зв'язку на основі пакетів.

Крім того, пристрої, підключені за допомогою CoAP, мають можливість виявляти та досліджувати один одного, щоб узгодити способи обміну даними між собою. CoAP також підтримує методологію спостереження за змінами стану ресурсу. Це модель передачі стану, яка дозволяє клієнту безперервно отримувати відповіді від сервера. Це важливо, наприклад, у програмі охорони здоров'я IoT, де дані з датчика, прикріпленого до пацієнта, є життєво важливими та потребують

постійного моніторингу. Іншими словами, CoAP — це асинхронний обмінник повідомленнями, який відбувається через спостереження/повідомлення. Подібно до HTTP, клієнт використовує команду запиту GET у спостережуваному режимі, щоб висловити інтерес до будь-яких оновлень із сервера. Клієнт отримує сповіщення кожного разу, коли змінюється стан ресурсу на сервері.

CoAP також розроблено як умовний спостерігач або модель на основі подій. Це означає, що він дозволяє клієнту отримувати повідомлення лише тоді, коли виконуються певні дії на спостережуваних ресурсах. Оскільки повідомлення надходять не при кожній події, а лише при потрібних подіях, завдяки отриманим керуючим повідомленням можна заощадити енергію. Наприклад, у додатку IoT для моніторингу температури датчик може надсилати оновлення щосекунди, навіть якщо нічого суттєвого не змінилося від однієї передачі даних до іншої. За допомогою спостережуваних ресурсів CoAP повідомлятимуться лише про цікаві події, які відбуваються періодично, або спостережуване значення змінюється із заздалегідь заданим розміром кроку. Ще одна особливість CoAP полягає в тому, що він підтримує проксі, тобто клієнт може запитувати дані від сервера CoAP за допомогою HTTP-запитів.

Типи повідомлень CoAP

Під час обміну повідомленнями в мережі CoAP існує чотири типи повідомлень. Це підтверджувані, непідтверджувані, підтвердження та скидання.

Коли клієнт надсилає запит на сервер із підтвердженими повідомленнями (CON), це вимагає, щоб одержувач підтвердив (ACK) повідомлення з тим самим ідентифікатором повідомлення.Ця передача між клієнтом і сервером зазвичай використовується, коли потрібна надійна доставка даних. Повторна передача даних відбувається після закінчення часу очікування ACK, і воно повторюватиме коло, доки не буде отримано ACK з ідентифікатором повідомлення. На рис.2.11 показане надійне передавання повідомлень між клієнтом і сервером.

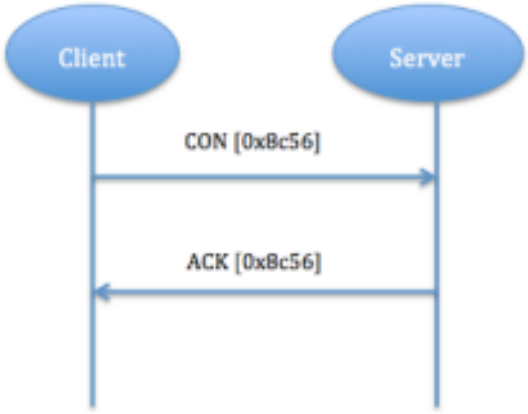


Рис.2.11 Передавання повідомлення-підтвердження CoAP

Техніка передачі даних без підтвердження не потребує ACK і є ненадійною. Цей тип техніки обміну даними використовує тип НЕ-повідомлення, який містить ідентифікатор повідомлення для контролю над передачею. Це найбільш поширено з потоками даних, де дані надсилаються, і існує ймовірність, що дані будуть втрачені або отримані не в порядку під час передачі. На рис.2.12 показане передавання повідомлення без підтвердження між клієнтом і сервером.

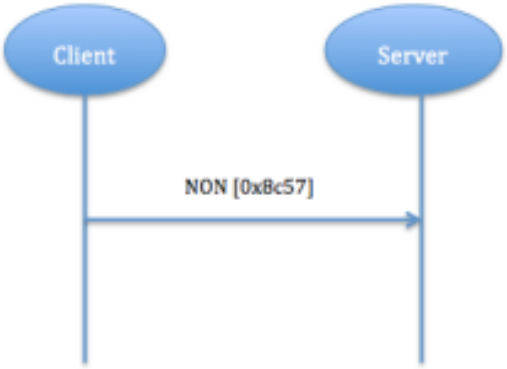


Рис.2.12 Передавання повідомлення без підтвердження CoAP

Повідомлення ACK з ідентифікатором повідомлення надсилається клієнту (відправнику) із сервера (одержувача), що надійшло конкретне підтверджуване повідомлення (CON).

CoAP також підтримує комбіновані повідомлення. Коли клієнт надсилає запит за допомогою повідомлень типу CON або NON, він негайно отримує повідомлення ACK, якщо це повідомлення підтверджується. ACK містить повідомлення-відповідь про успішну або невдалу доставку надісланого

повідомлення. Знову ж таки, коли сервер отримує запит на повідомлення CON і не може негайно відповісти на запит, він надсилає порожній ACK, щоб у випадку, якщо клієнт повторно надішле повідомлення через певний час. Однак новий CON надсилається клієнту щоразу, коли сервер готовий відповісти на повідомлення, і клієнт відповідає ACK, щоб підтвердити повідомлення CON від сервера. На рис.2.13 показано окремі відповіді, коли клієнт використовував запит GET, щоб запитати температуру від клієнта.

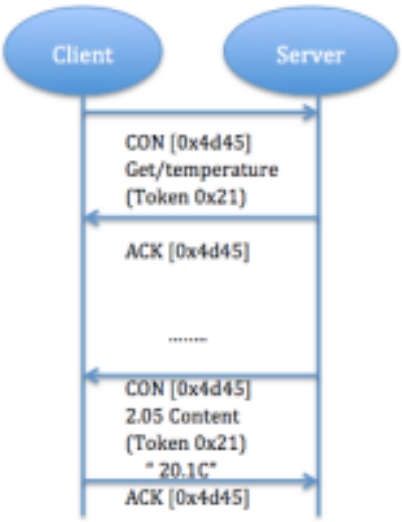


Рис.2.13 Повідомлення запиту CoAP CON з окремими відповідями

Таким чином, повідомлення ACK із повідомлень, що підтверджуються, не вказує на успішне чи невдале виконання будь-якого запиту, але повідомлення ACK може також містити відповідь із підключенням.

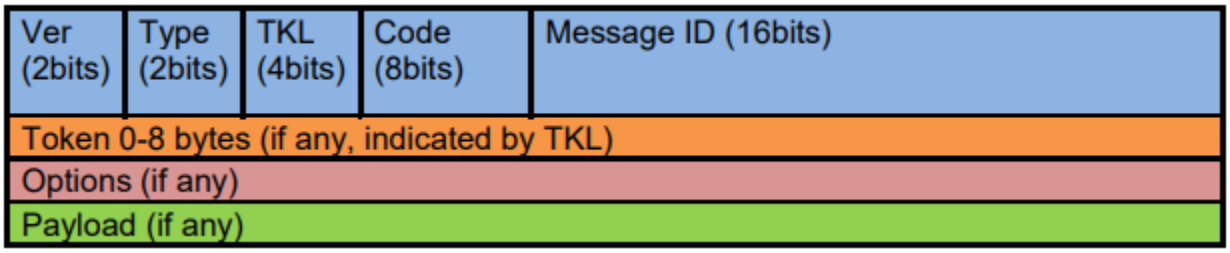
Скидання, також відоме як негативне підтвердження (NACK), — це повідомлення про помилку, надіслане з одержувача (сервера), щоб сповістити відправника (клієнта), що певне повідомлення втрачено або одержувач не зміг обробити повідомлення. Іншими словами, повідомлення про скидання надсилається для відхилення помилок і невідомих повідомлень, коли надходять певні повідомлення (підтверджувані чи непідтверджувані), але відсутній певний контекст для належної обробки.

Формат повідомлення CoAP

CoAP заснований на обміні компактними повідомленнями, які за замовчуванням передаються через UDP. Повідомлення CoAP кодуються в простому двійковому форматі, і це 4-байтовий заголовок фіксованого розміру, за яким слідують додаткові розширення, такі як значення маркера змінної довжини, послідовність нуля або більше параметрів CoAP у TypeLength-Value (TLV) формат і додаткове корисне навантаження, яке займає решту дейтаграми. Таблиця 9 показує структуру формату повідомлення CoAP. 4-байтовий заголовок складається з версії (Ver, 2-біт), типу (T, 2-біт), довжини маркера (TKL, 4-біт), коду (8-біт) та ідентифікатора повідомлення (8-біт).

Таблиця 2.1

Формат повідомлення CoAP



2-розрядний цілочисельний файл версії без знаку вказує номер версії CoAP, а для реалізації специфікації CoAP RFC-7252 йому встановлено значення 1 (двійковий 01). Це означає, що кожне повідомлення повинно мати цей номер версії. В іншому випадку повідомлення з невідомим номером версії мовчки ігноруються. Інші значення зарезервовано для майбутніх версій.

Тип (T) також є 2-розрядним беззнаковим цілим числом у заголовку, яке вказує, чи є повідомлення типу Confirmable, Non-Confirmable, Acknowledgement або Reset. Довжина маркера — це 4-розрядне ціле число без знаку в заголовку, яке вказує на довжину поля маркера змінної довжини, яка становить від 0 до 8 байтів. Якщо число встановлено на 0, це означає, що параметрів немає, і корисне навантаження (якщо воно є) слідує відразу за заголовком. Однак, якщо число

більше за 0, поле вказує на кількість варіантів, які слідують безпосередньо за заголовком.

Код також є 8-бітним беззнаковим цілим числом у заголовку, який розбивається на підполя; 3-бітовий клас (найбільш значущі біти) і 5-бітний деталь (молодший значущий біт). Клас може вказувати на запит, успішну відповідь, відповідь на помилку клієнта або відповідь на помилку сервера. Ідентифікатор повідомлення також є 16-бітним цілим числом без знаку в заголовку, і він використовується для виявлення дублювання повідомлення та зіставлення повідомлень типу Acknowledgement/REST з повідомленнями типу Confirmable/Non-Confirmable.

Значення маркера знаходиться поруч із заголовком і становить від 0 до 8 байтів, як зазначено в полі Token Length у заголовку. Він використовується для співвіднесення запитів і відповідей. Однак 8-байтовий заголовок допомагає захистити такі атаки, як спуфінг, і правилом є те, що всі повідомлення CoAP мають маркери, навіть якщо вони мають нульову довжину.

Як зазначалося раніше, параметри CoAP можуть бути присутніми, лише якщо значення поля маркера змінної довжини є ненульовим. Поле параметрів містить інформацію, яка впливає на продуктивність і функціональність CoAP. Крім того, CoAP визначає кількість опцій, які можна включити в повідомлення, і кожен екземпляр опції в повідомленні визначає номер опції, довжину опції та саме значення опції. Деталі та повний список опцій описано в документації RFC-7252. Корисне навантаження також є необов’язковим і може бути доступним лише тоді, коли воно має ненульову довжину та має префікс однобайтового маркера корисного навантаження (0xFF), який вказує кінець параметрів і початок корисного навантаження.

Дані корисного навантаження поширюються від маркера до кінця датаграми UDP. Відсутність маркера корисного навантаження означає корисне навантаження нульової довжини, а наявність маркера з іншого боку, за яким слідує корисне навантаження нульової довжини, має оброблюватися як помилка формату повідомлення. Крім того, повідомлення запиту та відповіді від клієнта та

сервера відповідно можуть містити корисні дані. Його також можна переносити разом із повідомленнями, що підтверджуються, і повідомленнями, що не підлягають підтвердженню. Це також може бути Piggybacked на повідомленнях підтвердження.

Передавання повідомлень між клієнтом і сервером

Обмін повідомленнями між кінцевими точками CoAP (клієнт-сервер) виконується асинхронно. CoAP використовує протокол UDP для транспортування повідомлень. Це неминуче буде ненадійним, а це означає, що повідомлення можуть надходити не в порядку, можуть виглядати дубльованими або непомітно зникати.

Однак CoAP реалізує легкий надійний механізм, подібний до протоколу TCP, який має такі функції:

- надійність повторної передачі з простою зупинкою та очікуванням із експоненційною відстрочкою для підтверджених повідомлень;
- виявлення дублікатів як для підтверджених, так і для непідтверджених повідомлень.

Повідомлення, що передаються в рамках CoAP, використовують методи передачі запитів і відповідей.

Запити (Requests)

Методи запиту CoAP подібні до методів запиту HTTP GET, POST, PUT і DELETE. Метод GET використовується для отримання стану інформаційного ресурсу, який надається в уніфікованому ідентифікаторі ресурсу (URI). Таку інформацію, як значення датчика, наприклад температура, назви пристроїв або стан пристрою, можна отримати за допомогою методу GET. Методи POST і PUT схожі за роботою, вони просто використовуються для створення нового ресурсу або під час оновлення цільового ресурсу. Запит методу DELETE використовується для видалення ресурсу, зазначеного URI.

CoAP підтримує метод спостереження за змінами ресурсу, і це інша форма методу запиту, який клієнт може використовувати для спостереження за ресурсом протягом певного періоду часу. Цей метод розроблено, оскільки методи GET, POST, PUT і DELETE не працюють належним чином, коли клієнт хоче спостерігати за ресурсом із сервера. Метод спостереження дозволяє вузлу сервера CoAP безперервно надсилати сповіщення після того, як він отримав реєстраційне повідомлення від клієнта. Мета сервера — оновлювати клієнта, повідомляючи спостерігача (клієнта) про останні значення ресурсу.

Коли спостерігач (клієнт) зацікавлений у спостереженні за ресурсом, він надсилає реєстраційне повідомлення на сервер. Надіслане повідомлення застосовує запит GET зі значенням параметра спостереження, встановленим на «0». Сервер додає клієнта в список спостерігачів ресурсу і починає розсилати повідомлення. Повідомлення сповіщень мають встановлене значення в полі спостережень і використовуються для перевірки оновленого вимірювання. У випадку, коли сервер не може додати нового спостерігача, він надсилає відповідь без значення параметра спостереження. На рис.2.14 показано, як клієнт реєструє свою зацікавленість у ресурсі та отримує сповіщення. У цьому прикладі клієнт зацікавлений у спостереженні за температурою на сервері та починає з надсилання реєстраційного повідомлення на сервер. Сервер додає клієнта (спостерігача) до своєї бази даних і починає надсилати сповіщення спостерігачу. Коли клієнт більше не зацікавлений у спостереженні за температурою, він надсилає повідомлення про скасування реєстрації зі значенням опції спостерігача «1».

Ще один спосіб для клієнта припинити спостереження - це відхилити сповіщення, надіславши повідомлення Reset. Крім того, повідомлення про передачу між сервером і клієнтом можуть бути підтвердженими або непідтвердженими. У разі повідомлення, що підтверджується, сервер очікує підтвердження від клієнта. Якщо після певного періоду часу з кількома повторними передачами він не отримав підтвердження, сервер вважає, що клієнт

більше не зацікавлений у спостереженні за ресурсом, а потім видаляє клієнта зі списку спостерігачів.

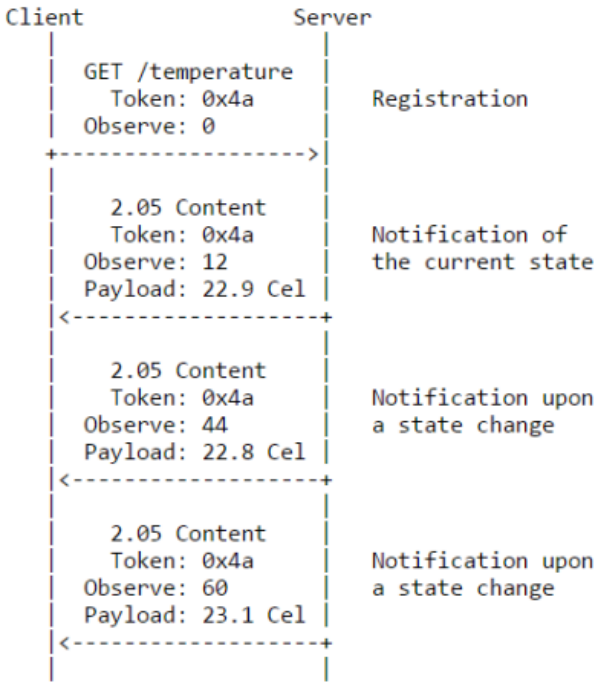


Рис.2.14 Спостереження клієнтом за ресурсом у CoAP

Відповідь (Response)

Коли запит надсилається від клієнта до сервера, сервер відповідає відповідним запитом за допомогою згенерованого клієнтом маркера. Відповідь ідентифікується за допомогою поля «Код» узаголовку CoAP, яке має значення «Код відповіді». Нижче наведено класи коду відповіді в CoAP.

Success 2.xx. Цей клас або код відповіді вказує на те, що запит клієнта було успішно отримано, зрозуміло та прийнято.

Client error 4.xx. Цей клас коду відповіді призначений для випадків, коли здається, що клієнт має помилку. Цей код відповіді застосовується до будь-якого методу запиту.

Server error 5.xx. Код відповіді класу помилок сервера вказує на випадки, коли сервер знає про помилку або нездатність виконати запит. Ці коди відповіді застосовуються до будь-якого методу запиту.

CoAP security

Як і в будь-якому спілкуванні між пристроями, безпека важлива, і вона не відрізняється від протоколу CoAP, який є стандартом (ISO/IEC 20922) для додатків IoT. Однак, розглядаючи безпеку будь-якої системи зв'язку, слід враховувати три елементи. Це цілісність системи, аутентифікація та конфіденційність. Безпека транспортного рівня датаграм (DTLS) RFC 6347 була розроблена як протокол безпеки для CoAP. По-перше, CoAP використовує транспорт дейтаграм, а DTLS може досягти вищезазначених елементів безпеки. Він добре підходить для захисту додатків і пристроїв, чутливих до затримки, має механізм переупорядкування повідомлень, які надходять з порушенням порядку, повторної передачі втрачених повідомлень під час рукостискання та розмірів повідомлень. Він толерантний до помилок під час дешифрування, але без повідомлень про помилки та припинення сеансу. Він також додає три механізми: 1 повторну передачу пакету, два призначення порядкового номера під час рукостискання та три виявлення повтору.

DTLS складається з двох рівнів. Нижній рівень, відомий як протокол запису DTLS, забезпечує безпеку підключення та має дві основні властивості:

- підключення приватне за допомогою симетричного шифрування;
- підключення є надійним завдяки перевірці цілісності повідомлень.

Ці властивості або параметри можна використовувати окремо, разом або не використовувати взагалі.

Верхній рівень складається з трьох протоколів, які включають сповіщення, рукостискання та дані програми.

Протокол рукостискання DTLS використовується для узгодження параметрів безпеки сеансу, який пізніше використовується для захищеного зв'язку.

Протокол сповіщень DTLS можна використовувати в будь-який час під час рукостискання й аж до закриття сеансу, повідомляючи про фатальні помилки або попередження.

Протокол даних програми DTLS складається з повідомлень даних програми, які виконуються на рівні запису та фрагментуються, стискаються та шифруються на основі поточного стану з'єднання.

Крім того, за деяких умов протокол Change Cipher Spec Protocol може замінити один із згаданих вище протоколів безпеки DTLS. Протокол повідомлення Change Cipher Spec використовується для сповіщення протоколу Record Protocol для захисту наступних записів за допомогою набору шифрів узгодження та ключів. Рис. 2.15 ілюструє процес протоколу рукостискання DTLS.

Рис.2.15 Процес рукостискання DTLS

2.5 Протокол MQTT

MQTT (ISO/IEC 20922) — це міжмашинний (M2M) транспортний протокол підключення IoT, який підходить для мереж із низьким енергоспоживанням і мережами з втратами. MQTT розроблено як клієнт-сервер і використовує парадигму протоколу обміну повідомленнями публікації/підписки. Його реалізація заснована на протоколі TCP/IP, який характеризується як надійний, впорядкований і перевірений на помилки протокол. Він надзвичайно легкий, відкритий, простий і простий у виконанні. Він призначений для забезпечення підключення до вбудованих пристроїв, щоб забезпечити зв'язок у обмежених середовищах, тобто зв'язок у пристроях і програмах M2M та IoT, де потрібен невеликий слід коду або пропускна здатність мережі обмежена.

MQTT використовує архітектуру публікації/підписки, яка складається з видавця (клієнта), передплатників, посередника (сервера), сеансів і тем. Парадигма публікації/підписки — це протокол зв'язку між клієнтом і сервером/абонентом, який потребує центрального посередника MQTT для керування та маршрутизації даних між вузлами або передплатниками мереж MQTT. Видавці — це легкі датчики, які підключаються до брокера для надсилання даних. Абоненти — це пристрої або додатки, які логічно підключені до клієнта, який зацікавлений у даних датчика, і вони підключені до Посередника, щоб отримувати інформацію, коли надходять нові дані. Посередник класифікує дані датчиків за темами та надсилає їх підписникам, зацікавленим у темах. Технічно теми — це черги повідомлень, які підтримують шаблон публікації/підписки для клієнтів, і логічно теми дозволяють клієнтам обмінюватися інформацією з визначеною семантикою.

Нарешті, сеанс ідентифікує приєднання клієнта до сервера. Уся комунікація між клієнтом і сервером відбувається як частина сеансу. Рис.2.16 ілюструє архітектуру передачі даних MQTT з посередником, який обслуговує сервер даних, направляючи всі дані до відповідних місць призначення.

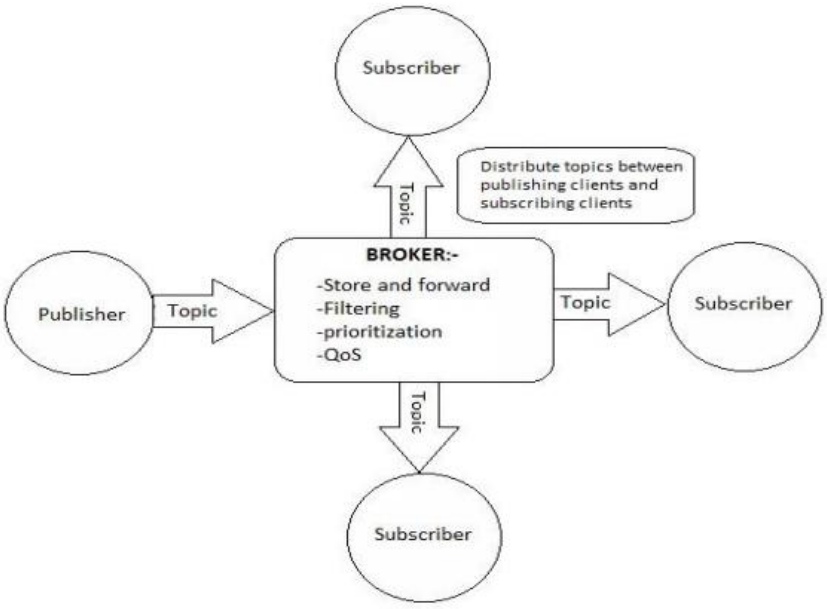


Рис.2.16 Архітектура передавання даних MQTT з посередником

Можна зробити висновок, що шаблон повідомлення публікації/підписки забезпечує обмін повідомленнями «один до багатьох» і що Посередник контролює розподіл інформації між клієнтом видавця (джерело даних) і клієнтом підписки (призначення даних). Посередник зберігає, пересилає, фільтрує та визначає пріоритетність опублікованих запитів від клієнта-видавця до клієнтів-передплатників. За допомогою системи MQTT Broker клієнти можуть перемикатися між ролями видавця та передплатника залежно від своїх цілей у конкретному випадку. Крім того, у Брокера є рівні якості обслуговування (QoS) MQTT. Рівні QoS становлять 0, 1 і 2, які описують зростаючі рівні гарантованої доставки повідомлень.

Обмін повідомленнями MQTT

MQTT визначає чотирнадцять (14) різних методів обміну повідомленнями. Основні типи повідомлень, які потрібно використовувати тільки кінцевим користувачам, це запит на підключення до сервера (підключити), від'єднати, підписатися, скасувати підписку та публікувати. Інші типи повідомлень використовуються для внутрішніх механізмів і потоків повідомлень. У табл.2.2 наведено список типів обміну повідомленнями.

Таблиця 2.2

Типи повідомлень MQTT

Enumeration	Mnemonic	Description
0	Reserved	Reserved
1 2	CONNECT CONNACK	Connection request to Server CONNECT Acknowledgement
3 4	PUBLISH PUBACK	PUBLISH message PUBLISH Acknowledgement
5	PUBREC	PUBLISH Received (assured delivery part 1)
6	PUBREL	PUBLISH Release (assured delivery part 2)
7	PUBCOMP	PUBLISH Complete (assured delivery part 3)
8 9	SUBSCRIBE SUBACK	SUBSCRIBE request SUBSCRIBE Acknowledgement
10 11	UNSUBSCRIBE UNSUBACK	UNSUBSCRIBE request UNSUBSCRIBE Acknowledgement
12 13	PINGREQ PINGRESP	Ping Request Ping Response
14	DSCONNECT	Client Disconnecting
15	Reserved	Reserved

Обмін повідомленнями підключення та підписки

На рис.2.17 показано налаштування сеансу підключення та підписки між клієнтом і сервером із встановленим прапором чистого сеансу 1(flag set 1) (flag =1).

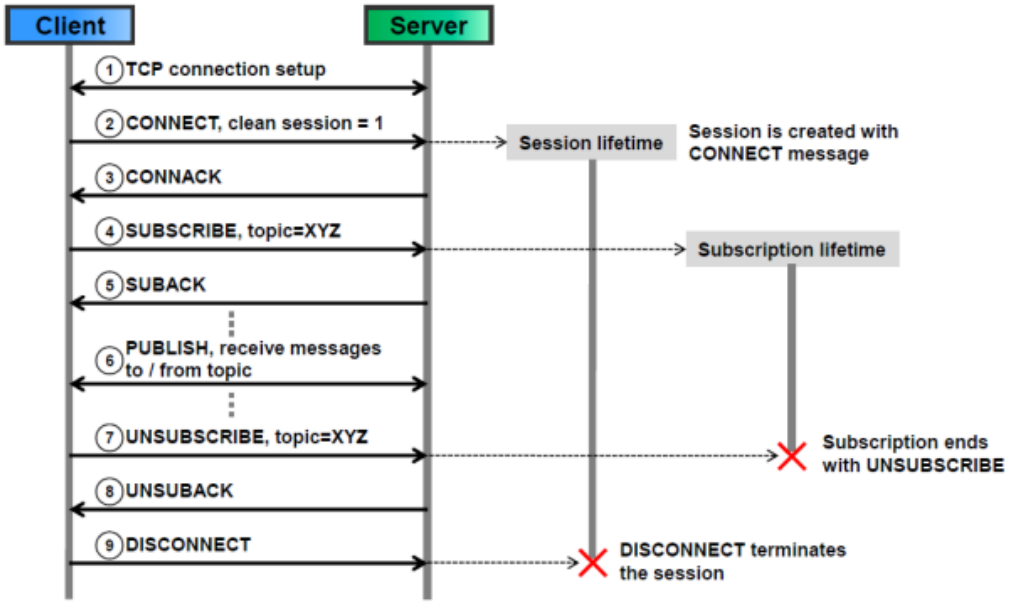


Рис.2.17 Повідомлення MQTT CONNECT і SUBSCRIBE

На рис.2.18 показано налаштування сеансу та підписки між клієнтом і сервером із установленим прапорцем чистого сеансу 0(flag set 0)(flag = 0).

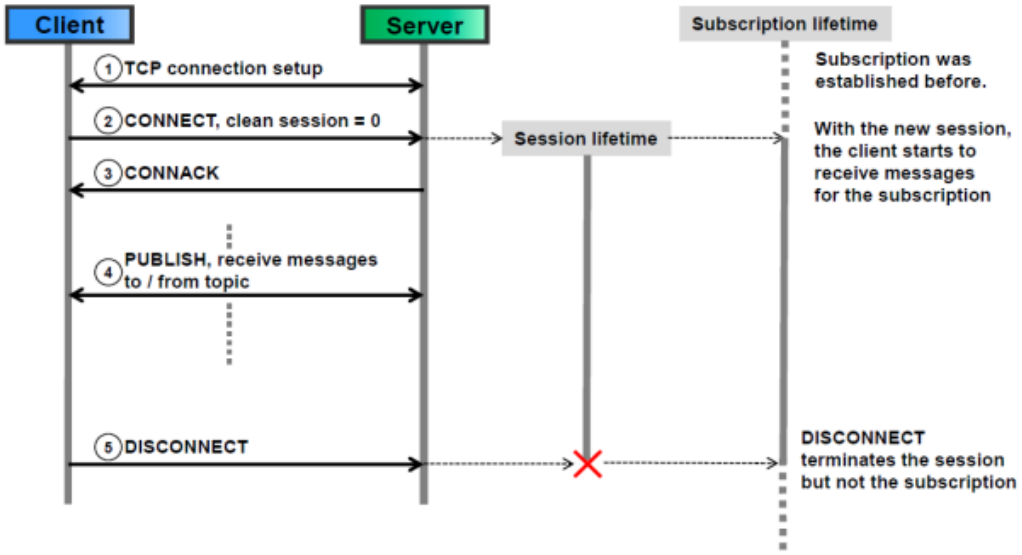


Рис.2.18 Повідомлення про підписку MQTT CONNECT (прапор = 0)

Формати обміну повідомленнями MQTT

Формат обміну повідомленнями протоколу MQTT, також відомий як контрольний пакет, складається з трьох частин; Фіксований заголовок, змінний заголовок і корисне навантаження. Кожен контрольний пакет MQTT містить фіксований заголовок. Він складається з двох байтів. Перший байт містить тип повідомлення та прапори, які мають такі поля, як прапор повторення (DUP), рівень QoS і RETAIN. Друге поле містить поле Remaining Length. Табл.2.3 ілюструє поля фіксованого заголовку.

Таблиця 2.3

Формат фіксованого заголовка MQTT

Field (bits)	Length	7	6	5	4	3	2	1	0
Byte 1		Message type				DUP flag	QoS level		RETAIN
Byte 2		Remaining Length (1-4 bytes)							

Як показано у табл.2.3, байт 1 складається з полів типу повідомлення та терміна «прапори» (DUP, рівень QoS і RETAIN). Другий байт (байт 2), поле Remaining Length має принаймні один байт. Подальший опис повідомлень MQTT у полях фіксованого заголовка пояснюється в табл. 2.4.

Таблиця 2.4

Пояснення поля фіксованого заголовку повідомлення MQTT

MQTT Message Fixed Header field	Description values	
Message Type	0: Reserved	8: SUNSCRIBE
	1: CONNECT	9: SUBACK
	2: CONNACK	10: UNSUBSCRIBE
	3: PUBLISH	11: UNSUBACK
	4: PUBACK	12: PINGREQ
	5: PUBREC	13: PINGRESP
	6: PUBREL	14: DISCONNECT
	7: PUBCOMP	15: Reserved
DUP (Duplicate) Flag	A Client or a Server (Broker) attempt to re-delivers a PUBLISH, SUBSCRIBE or UNSUBSCRIBE message. The Duplicate (DUP) bit is set as a message flag to indicates to the receiver a message may have already been received. This applies to messages with a QoS value greater that zero (0).	
QoS level	This indicates the level of delivery assurance of a PUBLISH message. Level 0: At most once delivery, no guarantee. Also, known as Fire and Forget Level 1: At least once delivery and with acknowledged delivery Level 2: Exactly once delivery with assurance of delivery Level 3: Reserved	
RETAIN	It instructs the Server (Broker) to RETAIN the last received PUBLISH message and deliver it as a first message to a new subscription after it has been delivered to the current subscribers. This is possible when the RETAIN flag is set to one (1).	
Remaining Length	It indicates the number of remaining bytes in the current message, including Data in the variable header and the payload.	

MQTT QoS

MQTT забезпечує типову доставку рівнів QoS проміжного програмного забезпечення, орієнтованого на повідомлення. Незважаючи на те, що протокол TCP/IP, на якому розташовується MQTT, забезпечує гарантовану доставку даних, проте втрата даних може статися під час передачі даних, якщо з'єднання TCP розривається. Тому MQTT додає три (3) рівні QoS поверх TCP.

QoS рівня 0

Максимально один раз (Fire and Forget). З цим рівнем QoS повідомлення доставляються відповідно до гарантій доставки основної мережі TCP/IP. У протоколі не очікується PUBACK і семантика повторних спроб не визначена. Повідомлення доставляється на сервер або не доставляється взагалі. Прикладом сценарію застосування може бути датчик температури. Дані датчика температури публікуються регулярно, і втрата будь-якого окремого значення не є критичною, оскільки користувачі, які підписалися на дані про температуру, інтегрують багато значень вибірки з часом, і, отже, окрема вибірка не має значення. Рис.2.19 ілюструє опублікований потік повідомлень із семантикою доставки рівня QoS 0.

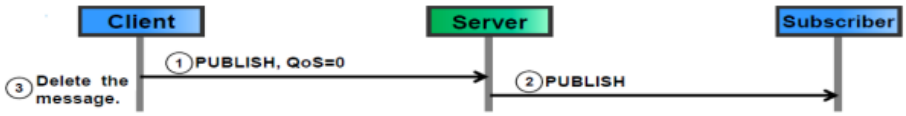


Рис. 2.19 Семантика доставки QoS рівня 0 не більше одного разу

QoS рівня 1

Хоча б один раз доставка. Завдяки цій семантиці доставки повідомлення гарантовано надходять на сервер і мають бути підтверджені (PUBACK). Однак може виникнути дублікат (DUP), який може виникнути через затримку в надходженні підтвердження (PUBACK) або виявлений (ID) збій або лінії зв'язку, або пристрою, що надсилає. Це означає, що коли відправник (клієнт) PUBLISH є даними, через деякий час, якщо PUBACK не отримано, він знову надсилає дані з бітом DUP, встановленим у заголовку повідомлення, що призводить до

дублювання повідомлень. Однак програма може відхилити повторюване повідомлення, оцінивши поле ідентифікатора повідомлення. Сценарієм застосування може бути датчик, який стежить за станом дверей. Це означає, що стан дверей відкрито/закрито або закрито/відкрито, і ці зміни станів публікуються передплатникам, наприклад, у формі сигналу тривоги або сигналу маячка. На рис.2.20 нижче показано семантику доставки QoS рівня 1.

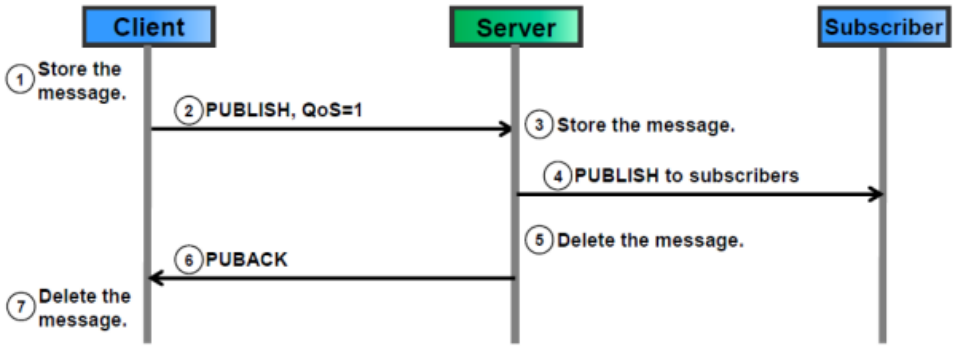


Рис.2.20 Семантика доставки QoS рівня 1 принаймні один раз

QoS рівня 2

Семантика одноразової доставки. Це найвищий, найбезпечніший і найповільніший рівень QoS, і він гарантує, що кожне повідомлення буде отримано абонентом лише один раз. Це також несе більшість накладних витрат з точки зору контрольних повідомлень і необхідності локального зберігання повідомлень. Це також поєднання семантики гарантії доставки принаймні один раз і більше не разу.

З рівнем QoS 2, коли отримувач отримав повідомлення PUBLISH, він обробляє це повідомлення та підтверджує його повідомленням PUBREC. Одержувач також зберігає повідомлення з посиланням на ідентифікатор повідомлення, доки він не надішле PUBCOMP. Це робиться для того, щоб уникнути дублювання обробки повідомлення двічі. Крім того, повідомлення PUBLISH магазину, що зберігається на клієнті, можна відкинути після отримання ним PUBREC. Повідомлення PUBREC зберігається після надходження, а клієнт відповідає PUBREL. Одержувач з іншого боку також видаляє всі збережені повідомлення після отримання PUBREL, і подібна подія відбувається на стороні

клієнта після отримання PUBCOMP від абонента. Рис.2.21 пояснює семантику QoS рівня 2.

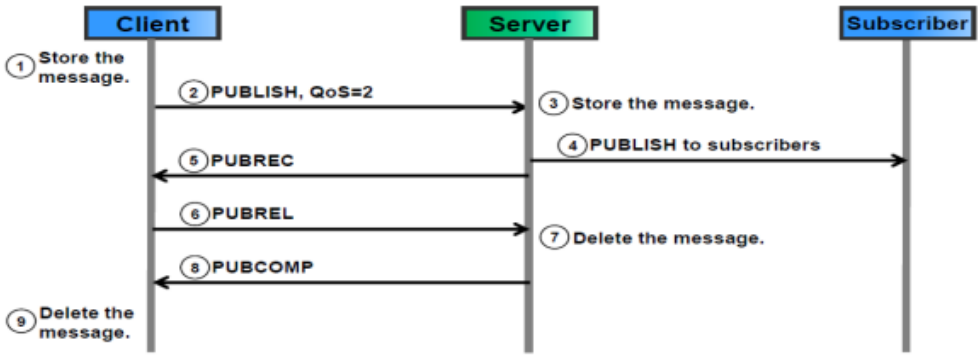


Рис.2.21 Семантика доставки QoS рівня 2 точно один раз

Заголовок змінної MQTT

Деякі типи повідомлень MQTT містять змінний компонент заголовка. Цей змінний компонент заголовка знаходиться між фіксованим заголовком і корисним навантаженням. Вміст заголовка змінної залежить від типу пакета. Поле ідентифікатора пакета заголовка змінної є загальним для кількох типів пакетів. Компонент багатьох типів контрольних пакетів складається з 2 байтів. Ідентифікатор пакета. Ці контрольні пакети: PUBLISH (де QoS >0), PUBACK, PUBREC, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE і UNSUBACK. Табл.2.5 ілюструє змінний формат заголовка, що знаходиться між фіксованим заголовком і корисним навантаженням з різними полями, включеними до нього.

Таблиця 2.5

Змінний заголовок, що знаходиться між фіксованим заголовком і корисним навантаженням

Field Length (bits)	7	6	5	4	3	2	1	0
Byte 1	Message Type				-	-		-
Byte 2	Remaining Length							
Byte 3	Protocol name UTF-8 encoded prefixed with 2 bytes string length (MSB)							
Byte 4	Protocol version (0x03 for MQTT version 3)							
Byte 4	Username Flag	Password Flag	Will RETAIN		Will QoS		Will Flag	Clean Session	Reserved
Byte 5	Keep Alive Timer MSB							
Byte 6	Keep Alive Timer LSB							
Byte 7	Client Identifier							
Byte 8	Will Topic							
Byte 9	Will Message							
Byte 10	Username							
Byte 11	Password							

Поля заголовка змінної описані в табл.2.6, де вони відображаються у заголовку.

Таблиця 2.6

Опис полів заголовка змінної MQTT

CONNECT Message Field	Опис / Значення (Description / Values)
Назва протоколу	Рядок назви протоколу в кодуванні UTF-8
Версія протоколу	Значення 3 для MQTT версії 3
Прапорець імені користувача	Якщо встановлено значення 1, це означає, що корисне навантаження містить ім’я користувача
Прапорець пароля	Якщо встановлено значення 1, це означає, що корисне навантаження містить пароль. Тобто, якщо встановлено прапорець імені користувача, прапорець пароля та пароль також повинні бути встановлені
Will RETAIN (Буде зберігати)	Якщо встановлено значення 1, це вказує або інформує сервер про повідомлення про волюмає бути збережено для Клієнта, який опубліковано у випадку Клієнт несподівано відключається
Will QoS	Він визначає рівень QoS для повідомлення Will
Чиста сесія	Якщо встановлено значення 1, сервер відкидає будь-яку попередню інформацію про клієнта, який повторно підключається (очистити новий сеанс). Якщо встановлено значення 0, Сервер зберігає підписки Клієнта, що від’єднується, включаючи збереження повідомлень рівня QoS 1 і 2 для цього Клієнта. Коли Клієнт повторно підключається, Сервер публікує Клієнту збережені повідомлення
Таймер Keep Alive	Використовується Сервером для виявлення порушених з’єднань з Клієнтом
Ідентифікатор клієнта	Ідентифікатор клієнта (від 1 до 23 символів) унікально ідентифікує клієнта для сервера. Ідентифікатор Клієнта має бути унікальним для всіх Клієнтів, які підключаються до Сервера
Will Topic	Тема Will, до якої публікується повідомлення Will, якщо встановлено позначку Will
Will Message (Буде повідомлення)	Повідомлення буде опубліковано, якщо встановлено позначку will
Ім'я користувача і Пароль	Ім’я користувача та пароль, якщо встановлено відповідні прапорці

Передплатників часто цікавить велика кількість тем. Однак підписка на кожну зазначену тему вимагає часу та ресурсів. Таким чином, MQTT Topic Wildcard використовується, коли клієнт хоче отримувати повідомлення різних тем зі схожою структурою одночасно. Теми можна впорядкувати за допомогою символів підстановки та символів підстановки; коса риска (/), знак числа (#) і знак плюс (+). Табл.2.7 описує символи підстановки та їх значення.

Таблиця 2.7

Символи підстановки MQTT та їх значення

Wildcard	Символ і приклад Symbol and example	Значення Meaning
Роздільник рівня теми	/ моя/теза/тема my/thesis/topic	Використовується для відокремлення кожного рівня в дереві тем і надання ієрархічної структури простору тем.
Однорівневий шаблон підстановки	+ моя/+/тема + my/+/topic	Відповідає одному повному рівню теми. Його можна використовувати кілька разів у підписці на тему.
Багаторівневий шаблон підстановки	# мій/# # my/#	Відповідає багатьом тематичним рівням. Це має бути останній символ підписки на тему.

Безпека MQTT (MQTT security)

Безпека є надзвичайно важливою незалежно від того, чи йдеться про банківські перекази, онлайн-магазини чи доступ до особистих документів через Інтернет. Крім того, основна ідея технології IoT полягає в тому, щоб з'єднати кожен об'єкт, наприклад, автомобілі, домашні та промислові машини, щоб можна було ефективно покращувати процеси, як бізнес, так і особисту діяльність. Однак підключення цих об'єктів до Інтернету означає розкриття життєво важливих і конфіденційних даних через Інтернет. Деяка життєво важлива та конфіденційна інформація може бути не призначена для загального споживання, і витік таких даних найчастіше завдає шкоди репутації постраждалої компанії чи особи. Отже, існує необхідність захисту таких даних від витоку.

Безпека в MQTT поділена на кілька рівнів, і кожен рівень запобігає різним видам атак. Рівні, на яких реалізуються деякі рівні безпеки, це рівень мережі, рівень транспорту та рівень додатків.

Впровадження зв'язку між Брокером і клієнтом через захищену мережу або віртуальну персональну мережу (VPN) є одним із надійних способів забезпечення безпеки з'єднань MQTT. Найкращою практикою безпеки на рівні мережі є реалізація шлюзу, де пристрої підключаються через шлюз, а брокер підключається через VPN. Основна роль шлюзу полягає в обробці та передачі інформації між пристроями та Інтернетом.

Безпека на транспортному рівні (TLS), наступник Secure Sockets Layer (SSL) — це криптографічний протокол, призначений для забезпечення безпечного зв'язку між клієнтом і сервером через незахищену мережу, а також у випадках, коли необхідно забезпечити конфіденційність системи. Він працює поверх TCP, забезпечуючи безпечний транспорт для протоколів верхнього рівня, таких як HTTP. TLS є дуже безпечним методом шифрування трафіку, але він також потребує ресурсів через потрібне рукостискання та збільшені накладні витрати на пакети. Однак, оскільки MQTT побудовано на основі TCP, він може використовувати TLS для захисту трафіку між клієнтом MQTT і сервером. Але оскільки TLS потребує ресурсів, а клієнти MQTT легкі, а енергія має високий пріоритет, достатньо шифрувати лише корисне навантаження замість шифрування всього пакета.

Відповідно до призначення портів і стандартів IANA.org, MQTT використовує або призначається порту 8883 на стороні посередника, коли використовується TLS. Це стандарт для підключень MQTT, коли він використовується поверх TLS. Крім того, коли MQTT використовується через з'єднання Plaintext TCP, він використовує порт 1883. Підключення до портів TLS і TCP можна змішувати, і не всі клієнти повинні бути підключені до брокера однаково. Це означає, що клієнт може підключитися до брокера за допомогою TLS і до іншого за допомогою відкритого тексту через TCP. TLS є складним протоколом. Він ресурсомісткий і обчислювальний, тому деякі цільові платформи

можуть не підтримувати його. Але за допомогою MQTT можна реалізувати безпеку та безпечні пакети на прикладному рівні.

Коли безпека застосована на прикладному рівні, вона реалізується в корисному навантаженні даних, де зберігаються дані програми. Зв'язок між клієнтом і сервером забезпечується таким чином, що він зашифрований, а ідентифікаційна інформація проходить автентифікацію. Ідентифікатор клієнта, облікові дані користувача та пароля також можна використовувати для захисту та автентифікації пристроїв на прикладному рівні. Вони можуть забезпечити передачу інформації за допомогою повністю реалізованого транспортного шифрування.

2.6 Порівняння протоколів MQTT та CoAP

Мережа Інтернету речей є складною через велику кількість фізичних взаємопов'язаних пристроїв IoT та обмежений характер пристроїв, середовища та типу мережі з втратами даних, якою вони керують. Одним із ключових завдань впровадження проекту IoT є ефективна підтримка міжмашинного зв'язку (M2M) у обмежених умовах. Поки що в цій роботі було розроблено, що MQTT і CoAP є найбільш перспективними протоколами, які можуть бути реалізовані в цих обмежених умовах.

Незважаючи на те, що MQTT і CoAP є абсолютно різними протоколами, у них є певна схожість, наприклад те, що вони призначені для використання в легких пристроях і в обмежених середовищах. Це означає, що обидва вони добре працюють із мережевими пристроями з низьким енергоспоживанням і обмеженими мережами. Через їхню схожість вибір відповідного протоколу для розробки програми IoT може бути складним залежно від програми. Однак під час планування правильного протоколу слід враховувати багато факторів. У цьому розділі буде розглянуто порівняння протоколу MQTT і протоколу CoAP на основі оцінки продуктивності за різними сценаріями, зробленими в інших місцях.

Основна відмінність між CoAP і MQTT полягає в тому, що CoAP працює поверх UDP, тоді як MQTT працює поверх TCP. Табл.2.8 ілюструє порівняння.

Таблиця 2.8

Порівняльна таблиця між MQTT і CoAP

MQTT	CoAP
Спосіб зв'язку в MQTT – це публікація та підписка, які сильно відокремлені один від одного	CoAP орієнтований на запит і відповідь і має модель асинхронного зв'язку
MQTT зазвичай має більший розмір пакета. Менші пакети розміром менше 127 байт мають пошук довжини пакета 1 байт, а максимальний розмір пакета становить 256 МБ.	CoAP має менший розмір пакета: MTU 1280 байт для IPv6, 127 байт для 6LOWPAN і 127 байт для IEEE 802.15.4
Поле заголовка MQTT має 2 байти	Поле заголовка CoAP становить 4 байти
MQTT дозволяє 16 різних типів обміну повідомленнями	CoAP дозволяє 4 типи повідомлень
MQTT підтримує асинхронний обмін повідомленнями	CoAP підтримує як синхронний, так і асинхронний обмін повідомленнями
MQTT має 3 рівні надійності програми, які є рівнями QoS	CoAP має 2 рівні надійності додатків у формі Confirmable (CON) і Non-Confirmable (NON).
Цикл передачі в межах MQTT набагато повільніший	CoAP має швидший цикл передачі
MQTT не є протоколом RESTful	CoAP — це протокол RESTful
MQTT працює за гнучкою тематичною підпискою	CoAP має стабільний механізм виявлення ресурсів
З міркувань безпеки MQTT не шифрується, але використовує шифрування TLS/SSL TCP	CoAP використовує безпеку DTLS UDP

2.7 Дослідження взаємодії в IoT

Загалом, сумісність — це ступінь, у якому дві або більше реалізованих систем від різних виробників можуть з'єднуватися, спілкуватися, обмінюватися, впроваджувати інновації, працювати та використовувати дані одна одної, покладаючись на послуги одна одної, як визначено загальним протоколом і стандартом. Основна ідея IoT полягає в тому, щоб підключити будь-який пристрій до Інтернету, який міг би підключитися до будь-якого іншого пристрою (пристроїв) або системи для обміну даними та інформацією. Однак в інфраструктурі різних доменів додатків IoT відсутні методи взаємозв'язку, які могли б забезпечити взаємодію між, наприклад, мережевими рівнями та рівнями додатків.

Подібно до традиційного Інтернету, взаємозв’язок або сумісність пристроїв і систем IoT відбувається в різному ступені та на різних рівнях у стеку протоколів зв’язку. Рівні, на яких відбувається взаємодія, це мережевий рівень, прикладний рівень і рівень анотації даних. На рис.2.22 показана архітектура сумісності протоколів мережевого рівня IoT з різними мережевими протоколами з низьким енергоспоживанням, такими як ZigBee, Z-Wave, Bluetooth, NFC, а також традиційними мережевими протоколами, такими як Ethernet, WiFi та апаратними з’єднаннями.

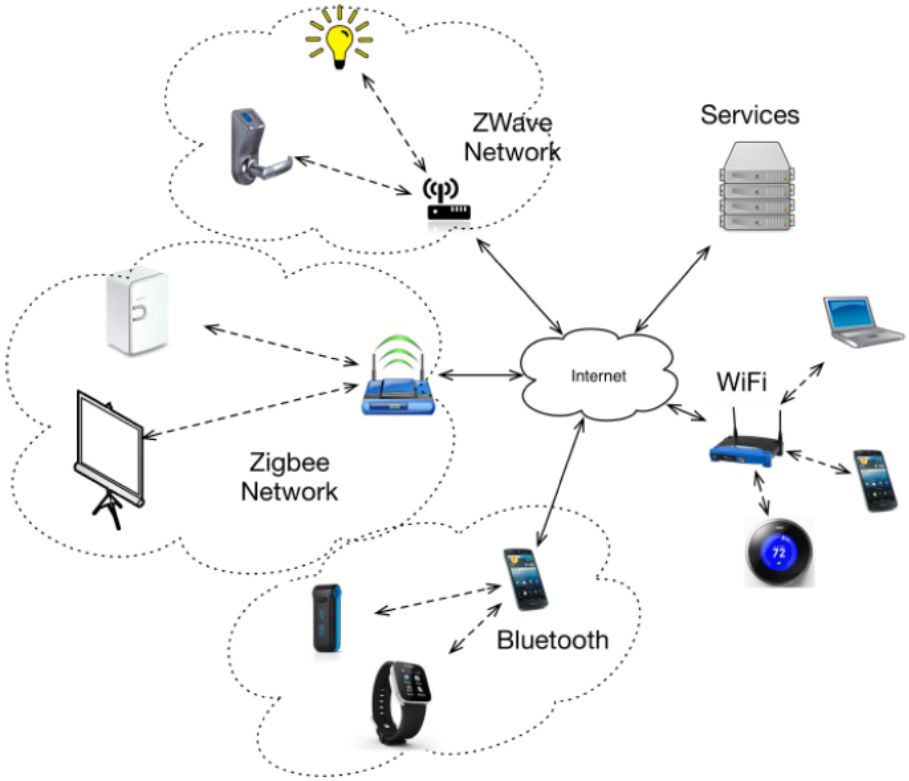


Рис.2.22 Архітектура сумісності мережного рівня IoT

Протоколи мережної сумісності розроблені для певного домену та програм для деяких стандартизованих апаратних компонентів, розроблених для підтримки кількох мережних протоколів.

Взаємодія між протоколами прикладного рівня

Як описано в цій дисертації, найбільш конкуруючими та пропонованими протоколами Інтернету речей прикладного рівня є CoAP та MQTT. Кожен протокол має унікальні характеристики та архітектуру масування для додатків

IoT. Однак сумісність між пристроями, реалізованими за допомогою цих протоколів прикладного рівня, та інших запропонованих протоколів IoT залишається проблемою.

Більше того, з'являються різноманітні ініціативи взаємодії, які працюють над вирішенням проблеми сумісності в протоколах прикладного рівня IoT. Більшість ініціатив мають відкритий вихідний код і побудовані на основі протоколів IoT Application Layer. Вони зосереджені на структурі даних, моделі зв'язку та семантиці даних IoT.

Відомими ініціативними консорціумами є AllSeen Alliance і AllJoyn, які є універсальними, безпечними і розробленими платформами зв'язку з відкритим кодом, які мають на меті підтримувати та забезпечувати взаємодію між пристроями IoT. Платформа AllJoyn підтримує виявлення пристроїв для взаємодії та взаємодії. Це означає, що продукти, програми та послуги, реалізовані за допомогою фреймворку AllJoyn, можуть підключатися навіть без доступу до Інтернету до різних мережевих рівнів, незалежно від виробника чи операційної системи.

Інша структура розробки, розроблена Open Interconnect Consortium (OIC) під назвою (IoTivity, n.d.) IoTivity, також реалізована для покращення сумісності між пристроями IoT. IoTivity — це фреймворк із відкритим вихідним кодом, який містить відкриття механізмів пристроїв, передачу даних у формі повідомлень і потокової моделі, обмін інформацією та механізм контролю, керування даними, зберігання, аналіз даних з інших джерел і керування пристроями. Він також забезпечує діагностику пристрою.

Взаємодія семантики

Взаємодія не може бути досягнута лише шляхом передачі даних із загальним форматом даних у межах протоколу IoT Application Layer. Взаємодія семантики забезпечує інший вимір взаємодії даних на прикладному рівні на вищому рівні, ніж необроблені дані, що передаються за допомогою AllJoyn або IoTivity. Семантична сумісність означає, що дві окремі системи автоматично

інтерпретують значення даних, переданих двома системами, і отримують однакові значення. Що стосується платформ IoT, Sensor Semantic Network (SSN) надає набір онтологій, а SenML (Sensor Markup Language) також надає метамоделі, які розроблені для забезпечення взаємодії з додатками мов, таких як JavaScript Object Notation (JSON), Eclipse Vorto або Eclipse Ponte та Eclipse Franca (Ієрархія проєктів з відкритим вихідним кодом Eclipse Foundation, n.d.). Метамоделі та онтології пов’язані, але метамодель називають суворим набором правил, тоді як онтології є словниками.

РОЗДІЛ 3. ВПРОВАДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПРОТОКОЛУ MQTT

3.1 Впровадження та використання протоколу

Протокол MQTT забезпечує легкий спосіб використання моделі публікації/підписки для обміну повідомленнями. У цьому підрозділі описано і представлено деякі з широко використовуваних брокерів MQTT і клієнтських бібліотек.

Mosquitto є ліцензованим EPL/EDL брокером повідомлень з відкритим кодом. Він розроблений Eclipse Foundation і підтримує версії протоколу MQTT 3.1, 3.1.1.1 і 5.0. Це однопотокова немасштабована реалізація. Він написаний на C. Він також пропонує бібліотеку C для реалізації клієнтів MQTT.

Bevywise MQTT Route — комерційно ліцензований брокер повідомлень із закритим кодом на основі MQTT, розроблений Bevywise Networks. Він побудований на специфікаціях MQTT 3.1 і 3.1.1.1 і реалізує всі рівні QoS MQTT. Це немасштабована реалізація з фіксованим потоком (2 потоки). Він написаний на C і Python. Він також забезпечує реалізацію клієнта MQTT.

EMQ X розроблено EMQ Inc. Це посередник/клієнт повідомлень з відкритим вихідним кодом на основі Erlang, побудований на специфікаціях MQTT 5.0. Він має високу масштабованість і підтримує кластеризацію. Він може працювати практично скрізь, починаючи від краю до хмари. Він використовує ліцензію Apache версії 2.0.

HiveMQ CE (Community Edition) — це масштабований брокер MQTT з відкритим вихідним кодом на основі Java. Він підтримує MQTT 3.x і MQTT 5.0. Він розроблений компанією HiveMQ GmbH. Він поширюється з ліцензією Apache версії 2.0.

HiveMQ — це масштабована, комерційно ліцензована платформа обміну повідомленнями на основі MQTT, побудована на специфікаціях 3.x та 5.0 версій протоколу MQTT. Він написаний на Java і розроблений HiveMQ GmbH. HiveMQ також має реалізацію клієнта MQTT.

IBM Watson IoT Platform Message Gateway — це комерційно ліцензований масштабований сервіс обміну повідомленнями на основі MQTT 3.x та 5.0. Він пропонує понад 40 клієнтських бібліотек MQTT.

JoramMQ є відкритим вихідним кодом, масштабованим брокером повідомлень на основі специфікацій MQTT 3.x. Він розроблений ScalAgent. Він має комерційні та безкоштовні розповсюдження, які поширюються відповідно до комерційної ліцензії та GNU Lesser General Public License.

flespi — комерційно ліцензована, масштабована платформа IoT, розроблена Gurtam. Він написаний на C і реалізує версії специфікації MQTT 3.1, 3.1.1 і 5.0.

PubSub+ — це брокер подій, який реалізує специфікацію MQTT 3.1.1. Він розроблений Solace і доступний у безкоштовній і комерційній версіях.

Thingstream — це масштабована платформа IoT-комунікацій як сервіс, яку пропонує u-blox. Він реалізує специфікацію MQTT 5.0 і пропонує як брокера MQTT, так і клієнтську бібліотеку. Він комерційно поширюється.

VernemQ — це широкомасштабований брокер MQTT з відкритим вихідним кодом, розроблений Octavo Labs AG. Він написаний на Erlang і реалізує версії MQTT 3.x і 5.0. Він використовує ліцензію Apache версії 2.0.

RabbitMQ — це брокер повідомлень на основі erlang, який підтримує версію MQTT 3.1.1. Він розроблений Pivotal Software. Він використовує ліцензію MPL 1.1.

Apache ActiveMQ — це відкритий, масштабований, багатопротокольний сервер повідомлень, написаний на Java і розроблений Apache Software Foundation. Він поширюється у двох смаках – «Класика» та «Артеміда». Він використовує ліцензію Apache версії 2.0. Він реалізує специфікацію MQTT 3.1.1.

Adafruit IO розроблений Adafruit. Він надає клієнтські бібліотеки MQTT з відкритим кодом для Python, Ruby та Arduino. Він реалізує протокол MQTT 3.1.1 і використовує ліцензію MIT для розповсюдження програмного забезпечення. net-mqtt — це реалізація MQTT з відкритим кодом для мови Haskell, розроблена Дастіном Саллінгсом. Він підтримує версії протоколу 3.x, 5.0. Він використовує ліцензію BSD 3.

Eclipse Paho MQTT пропонує клієнтські бібліотеки з відкритим кодом на мовах Java, Python, JacaScript, GoLang, C, C++, Rust і .Net(C#). Бібліотека Paho C реалізує всі версії MQTT, а бібліотеки лише на всіх інших мовах, які реалізують версії 3.1 і 3.1.1 протоколу MQTT. Він використовує Eclipse Public License 1.0 і Eclipse Distribution License 1.0 (BSD).

wolfMQT — це клієнтська бібліотека C MQTT з відкритим вихідним кодом для вбудованого використання. Він розроблений WolfSSL і доступний у GPL v2.0 і комерційних ліцензіях.

Eclipse M2Mqtt — це клієнтська бібліотека C# з відкритим кодом для платформ .Net і WinRT. Він використовував Eclipse Public License 1.0.

Machine Head — це бібліотека MQTT з відкритим вихідним кодом на основі clojure, яка поширюється з ліцензією Creative Commons Attribution 3.0 Unported. Він розроблений ClojureWerkz.

MQTT-C — це відкрита бібліотека мови C на основі MQTT, розроблена Ліамом Біндлом. Він поширюється з ліцензією MIT.

Таблиця 3.1

Порівняння можливостей брокерів MQTT

Features	**Source code availability model**		**Design and Implementation**			
MQTT Brokers	Open/Close	Software License	Written in	Supported OS	Latest stable release, release date	Developed by
Mosquitto	Open	Eclipse Public License 1.0, Eclipse Distribution License 1.0 (BSD)	C	Linux, Mac, Windows	1.6.9, 2020-02-27	Eclipse Foundation
Bevywise MQTT Route	Close	Commercial License	C, Python	Linux, Unix, MacOS, Windows, Raspbian	2.0, 2019-12-03	Bevywise Networks
EMQ X	Open	Apache License version 2.0	Erlang	Linux, Mac, Windows, BSD	3.0, 2019-04-03	EMQ Enterprise Inc.
HiveMQ CE	Open	Apache License version 2.0	Java	Linux, Mac, Windows	2020.3, 2020-07-06	HiveMQ GmbH
HiveMQ	Close	Apache License version 2.0	Java	Linux, Mac, Windows	4.3.5, 2020-07-17	HiveMQ GmbH
IBM WIoTP Message Gateway	Close	Commercial License	C	Linux	5.0.0.1, 2019-02-29	IBM
JoramMQ	Close	Commercial License, LGPL	Java	Linux, Unix, MacOS, Windows, Raspbian	1.13, 2019-04-29	ScalAgent
flespi	Close	Commercial License	C	-	-, 2018-04-05	Gurtam
PubSub+	Close	Commercial License, Free Version	C, C++	Linux, Mac, Windows	8.13, 2018-09-28	Solace
Thingstream	Close	Commercial License	C, C++, Java, JavaScript, Python, Go	-	3.3.0, 2019-03-14	u-box
VerneMQ	Open	Apache License version 2.0	Erlang	Linux, Mac	1.9.1, 2019-08-12	Octavo Labs AG
RabbitMQ	Open	MPL 1.1	Erlang	Linux, Unix, Mac, Windows, BSD	3.8.1, 2019-10-31	Pivotal Software
ActiveMQ	Open	Apache License version 2.0	Java	Windows, Unix, Linux, Cygwin	5.15.10, 2018-09-01	Apache Software Foundation
ActiveMQ Artemis	Open	Apache License version 2.0	Java	Windows, Unix, Linux, Cygwin	2.10.1, 2019-09-26	Apache Software Foundation

3.2 Таксономія особливостей реалізації MQTT

Щоб порівняти та проаналізувати особливості реалізацій протоколу MQTT, ми визначаємо сім категорій таксономії.

Модель доступності вихідного коду. Існує два типи моделі доступності вихідного коду: із закритим кодом і з відкритим кодом. Під моделлю з відкритим вихідним кодом вихідний код випущеного програмного продукту можна переглядати та змінювати. Відповідно до моделі із закритим вихідним кодом вихідний код не надається у відкритому доступі. Доступні різні ліцензії на комерційне та некомерційне розповсюдження та перерозповсюдження програмного забезпечення. Ліцензія на програмне забезпечення є юридичним документом, який надає постійні вказівки щодо використання та розповсюдження програмного забезпечення.

Показники вихідного коду: У цьому елементі таксономії вибрано показники вихідного коду, щоб представити та порівняти різні характеристики якості вихідного коду перевірених брокерів MQTT. Погана якість програмного забезпечення ускладнює читання та розуміння коду, важче вгадати функціональність або перевірити наявність невизначених символів, а також повторно використовувати код. Це негативно впливає на загальну продуктивність працівників і прибутковість компанії.

У цій роботі я використав аналізатор вихідного коду під назвою CLOC для обчислення певних показників якості. CLOC — це інструмент командного рядка, який приймає імена файлів, каталогів та/або архівів як вхідні дані та розпізнає прикладні мови програмування для розробки програмного забезпечення, підраховує кількість файлів, порожніх рядків, рядків коментарів та рядків коду. CLOC працює лише з програмами з відкритим кодом. Тому в цьому порівнянні перерахувано лише брокери MQTT з відкритим кодом.

Розробка та впровадження: ця категорія зосереджена на артефактах, пов’язаних з розробкою програмного забезпечення, таких як мова програмування, яка використовується для розробки програми, підтримувані операційні системи

або платформи, останній стабільний випуск і дата випуску, можливість крос-компіляції, а також організація, компанія чи особа що розробляє.

Функції протоколу: у цій категорії ми порівнюємо різні реалізації MQTT відповідно до підтримуваних функцій MQTT, таких як QoS, збереження, постійний сеанс, спільні підписки, журнал помилок, вбудований шлюз, версія MQTT(3. x/5.0), а також доступність підтримки MQTT-SN тощо. Якщо параметр «Зберегти» увімкнено (встановлено значення «true»), брокер зберігає останнє збережене повідомлення та відповідне QoS для цієї теми. «Постійний сеанс» представляє поточне з'єднання з брокером повідомлень MQTT). «Спільні підписки» — це функція MQTT V5, яка дозволяє клієнтам MQTT спільно використовувати одну підписку на брокера. MQTT-SN відноситься до MQTT для сенсорних мереж. «Вбудований шлюз» відноситься до шлюзу MQTT, який діє як посередник між датчиками/пристроями та будь-якою платформою IoT.

Безпека: у цій категорії перераховано різні функції, пов'язані з безпекою, реалізовані різними брокерами MQTT та клієнтськими бібліотеками, такими як автентифікація, MQTT через TSL/SSL, TCP, WS/WSS, безпека потоків тощо. Для зв'язку MQTT через TSL/SSL порт 8883 зарезервовано. MQTT через Websockets дозволяє отримувати дані MQTT безпосередньо через веб-браузер.

Хмарні пропозиції: у цій категорії різні реалізації MQTT порівнюються з точки зору доступності Bare Metal, доступності хмарного хостингу та підтримки Docker.

Підтримка візуалізації даних: такі функції утворюють цю категорію: користувальницький інтерфейс та інформаційна панель, користувацький інтерфейс користувача, інтеграція ElasticSearch, підтримка Tableau та інтеграція ElasticSearch. «Tableau» — це сервіс візуалізації даних та інтеграція ElasticSearch». Це розподілена пошукова система RESTful, створена для хмари.

Таблиця 3.2

Порівняння брокерів MQTT

MQTT Broker	Prime programming language	No. of files	No of blank lines	No. of comment lines	Lines of code
VerneMQ	Erlang	286	6701	8649	46014
RabbitMQ	Erlang	340	14436	13361	95078
Mosquitto	C	540	12027	7628	58519
HiveMQ CE	Java	1349	34768	36414	127000
EMQX	Erlang	181	5005	4664	22370
ActiveMQ	Java	5058	113811	162830	474415

3.3 Порівняння брокерів

Щодо порівнюваних брокерів MQTT, майже 50% досліджених рішень мають відкритий код, а 50% інструментів мають закритий код. Що стосується впровадження переглянутих брокерів, майже 57,2 % рішень написані на C та Java, де внесок кожної мови становить 28,60 %, 14,30 % рішень написані на Erlang, C++ та Python: кожне мова складає 9,50% рішень, майже 4,8% розглянутих рішень написані на JavaScript, а частка рішень на основі Go також становить ті ж 4,8% рішень. Ми бачимо, що ActiveMQ має найбільшу кількість рядків коду з великою кількістю коментарів. За кількістю рядків коментарів і рядків коду за ActiveMQ йдуть HiveMQ CE, EMQX, VerneMQ, RabbitMQ і Mosquitto відповідно. Майже всі рішення підтримують більшість категорій QoS, LWT та функції постійного підключення. RabbitMQ підтримує лише QoS 0 і 1. З усіх розглянутих брокерських рішень до 61,53% підтримують функцію спільної підписки, 57,14% рішень вже використовують MQTT 5.0, а 69,23% рішень не реалізують MQTT-SN. Всі брокери ввімкнули функції безпеки через аутентифікацію, MQTT через TSL/SSL і WS/WSS. З усіх розглянутих рішень близько 64,28% інструментів підтримують хмарний хостинг, 85,71% рішень мають доступність контейнера Docker, 92,3% мають інтерфейс користувача та інформаційну панель. Лише невелика кількість рішень підтримують інтеграцію сервісів Bare Metal, ElasticSearch і Tableau.

Таблиця 3.3

Порівняння можливостей брокерів MQTT з властивостями протоколу

Implemented protocol specific features									
MQTT Brokers	QoS Support	Retain Flag	Persistent Session	Shared Subscriptions	Last Will and Testament	Error Log	Built-in Gateway	MQTT Version	MQTT-SN Support
Mosquitto	0, 1, 2	Yes	Yes	No	Yes	No	No	3.1.1, 5.0	No
Bevywise MQTT Route	0, 1, 2	Yes	Yes	Yes	Yes	Yes, user can view in UI	Yes	3.x, 5.0	Yes
EMQ X	0, 1, 2	Yes	Yes	Yes	Yes	No	No	3.1.1	Yes
HiveMQ CE	0, 1, 2	Yes	Yes	Yes	Yes	No	No	3.x, 5.0	No
HiveMQ	0, 1, 2	Yes	Yes	Yes	Yes	No	No	3.x, 5.0	No
IBM WIoTP Message Gateway	0, 1, 2	Yes	Yes	-	-	Yes	Yes	3.x, 5.0	-
JoramMQ	0, 1, 2	Yes	Yes	Yes	-	Yes	No	3.x	Yes
flespi	0, 1, 2	Yes	Yes	Yes	Yes	Yes	No	3.1,3.1.1,5.0	No
PubSub+	0, 1, 2	Yes	Yes	Yes	Yes	Yes	Yes	3.1.1	No
Thingstream	0, 1, 2	-	-	No	-	-	No	5.0	Yes
VerneMQ	0, 1, 2	Yes	Yes	Yes	Yes	Yes	No	3.x, 5.0	No
RabbitMQ	0, 1	Partial	Yes	No	Yes	Yes	Yes	3.1.1	No
ActiveMQ	0, 1, 2	Yes	Yes	No	Yes	-	No	3.1.1	No
ActiveMQ Artemis	0, 1, 2	Yes	Yes	No	Yes	-	No	3.1.1	No

Майже 83,33% рішень мають відкритий код. Щодо реалізації розглянутих брокерів, майже 22,9% рішень написані на C, 17,1% рішень на Java. За рішеннями на основі C і Java йдуть C++ (11,4%), Python (11,4%), Erlang (8,6%), C# (5,7%), PHP (5,7%), Perl (5,7%), Ruby (5,7%) Рішення на основі , Go (2,9%), JavaScript (2,9%). Paho MQTT і Thingstream підтримують найбільшу кількість мов програмування, наприклад C, C++, Java, JavaScript, Python і Go. Серед усіх розглянутих бібліотечних рішень клієнти HiveMQ MQTT, net-mqtt, MQTT-C, Mosquitto і EMQ X підтримують функцію потокової безпеки. Клієнтські бібліотеки MQTT, такі як net-mqtt, Paho-MQTT, wolfMQTT, MQTT-C, Mosquitto і EMQ X, підтримують перехресну компіляцію. Майже всі бібліотечні рішення підтримують більшість категорій QoS.

Security features			
MQTT Brokers	**Authentication**	**MQTT over TSL/SSL**	**WS/WSS**
Mosquitto	Yes	Yes	Yes
Bevywise MQTT Route	Yes	Yes	Yes
EMQ X	Yes	Yes	Yes
HiveMQ CE	Yes	Yes	Yes
HiveMQ	Yes	Yes	Yes
IBM WIoTP Message Gateway	Yes	Yes	Yes
JoramMQ	Yes	Yes	Yes
flespi	Yes	Yes	Yes
PubSub+	Yes	Yes	Yes
Thingstream	Yes	Yes	Yes
VerneMQ	Yes	Yes	Yes
RabbitMQ	Yes	Yes	Yes
ActiveMQ	Yes	Yes	Yes
ActiveMQ Artemis	Yes	Yes	Yes

З цього порівняння ми виявили, що «контейнер Docker», «підтримка MQTT через WS/WSS», «автентифікація», «інтеграція REST API», «MQTT через TSL/SSL», «підтримка TCP», «зберігати позначку», « постійний сеанс», «Інтерфейс користувача та інформаційна панель», «Остання воля та заповіт» і «QoS» є найбільш підтримуваними функціями всіма брокерами. Також виявлено, що RabbitMQ не підтримує підписки QoS2. RabbitMQ автоматично знижує QoS 2 до QoS1. Деякі з найменш підтримуваних функцій: «Підтримка MQTT-SN», «вбудований шлюз», «підтримка локалізації», «голий метал», «підтримка таблиць», «підтримка правил», «журнал помилок» тощо.

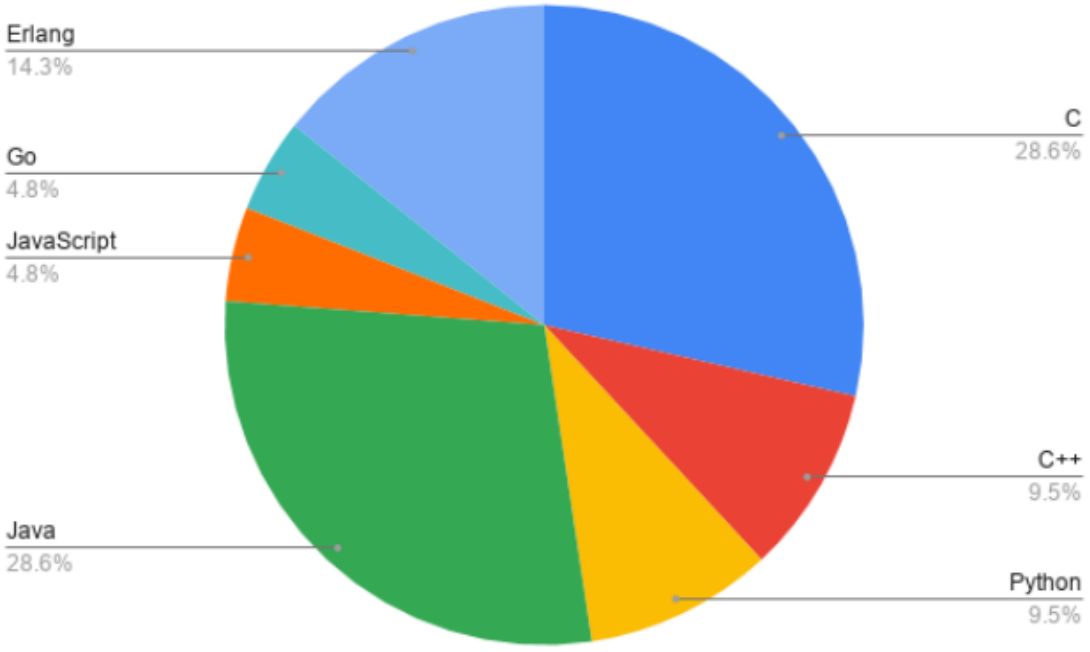


Рис. 3.1 Співвідношення мов програмування, що використовуються для реалізації брокерів MQTT

Таблиця 3.4

Порівняння хмарних і візуальних властивостей брокерів MQTT

Features	**Cloud offerings**			**Data visualization support**		
MQTT Brokers	Bare Metal	Cloud Hosting	Docker Container Availability	UI and dashboard	ElasticSearch Integration	Tableau Support
Mosquitto	-	Yes	Yes	No	No	No
Bevywise MQTT Route	Yes	Yes	Yes	Yes	Yes	Yes
EMQ X	Yes	Yes	Yes	Yes	No	No
HiveMQ CE	-	Yes	Yes	Yes	No	No
HiveMQ	-	Yes	Yes	Yes	No	No
IBM WIoTP Message Gateway	Yes	Yes	Yes	Yes	No	No
JoramMQ	-	No	No	-	Yes	No
flespi	-	Yes	Yes	Yes	No	Yes
PubSub+	Yes	Yes	Yes	Yes	No	No
Thingstream	-	Yes	No	Yes	No	No
VerneMQ	-	No	Yes	Yes	No	No
RabbitMQ	-	No	Yes	Yes	No	No
ActiveMQ	-	No	Yes	Yes	No	No
ActiveMQ Artemis	-	No	Yes	Yes	No	No

Таблиця 3.5

Порівняння клієнтських бібліотек MQTT

Features	**Source code availability model**		**Design**			**Implemented Protocol features**			**Security**		
MQTT Client Libraries	Open/Close	Software License	Written in	Supported OS	Ability to cross compile	MQTT Version	QoS	MQTT-SN	TSL/SSL	WS/WSS	Thread Safety
Adafruit IO	Open	MIT License	Ruby on Rails, Node.js, Python	CentOS, Debian, Docker, Mac OS X, Linux, Raspbian	-	-	0, 1	-	-	-	-
HiveMQ MQTT Client	Open	Apache License Version 2.0	Java	Linux, Mac, Windows	-	3.x, 5.0	0, 1, 2	-	Yes	Yes	Yes
net-mqtt	Open	BSD 3	Haskell	-	Yes	3.x, 5.0	0, 1, 2	Yes	Yes	-	Yes
Paho MQTT	Open	Eclipse Public License 1.0, Eclipse Distribution License	C, C++, Java, JavaScript, Python, Go	Varies upon language. Details Here: https://www.eclipse.org/paho/downloads.php	Yes (For C, C++ clients)	3.x,5.0 (only C and Java client library)	0, 1, 2	Yes	Yes	Yes	-
Thingstream	Close	Commercial License	C, C++, Java, JavaScript, Python, Go	-	-	5.0	0, 1, 2	Yes	Yes	-	-
wolfMQTT	Open	GNU Public License Version 2, Commercial License	C	Win32/64, Linux, Mac OS X, FreeRTOS Microchip, Harmony, Nucleus	-	3.1,1, 5.0	0, 1, 2	Yes	Yes	-	-
M2Mqtt	Open	Eclipse Public License 1.0	C#	Windows		3.x	0, 1, 2	-	Yes	-	-
Machine Head	Open	Creative Commons Attribution 3.0 Unported License	Clojure	Mac OS X, Linux	-	3.x	-	-	-	-	-
MQTT-C	Open	MIT License	C	-	Yes	3.x	0, 1, 2	-	Yes	-	Yes
Mosquitto	Open	Eclipse Public License 1.0, Eclipse Distribution License 1.0 (BSD)	C	Windows, BSD, Linux macOS, QNX	Yes	3.x, 5.0	0, 1, 2	No	Yes	Yes	Yes
EMQ X	Open	Apache License version 2.0	Erlang	Linux, Unix, MacOS, Windows,Raspberry P	Yes	3.x, 5.0	0, 1, 2	Yes	Yes	Yes	Yes
Bevywise MQTT Route	Close	Commercial license	C, Python	Linux, Unix, MacOS, Windows, Raspberry P	-	3.x, 5.0	0, 1, 2	Yes	Yes	Yes	-

3.4 Тенденції та проблеми

MQTT є одним із найбільш широко використовуваних рішень протоколу IoT. Легкий принцип проектування MQTT дозволяє обмінюватися даними у форматі звичайного тексту, що представляє загрозу безпеці. Отже, кілька реалізацій брокерів MQTT дозволяють шифрувати окрему функцію на верхній частині TLS. Це призводить до деяких накладних витрат на продуктивність. В даний час багато брокерів MQTT використовують пакет повідомлень типу CONNECT, щоб увімкнути аутентифікацію. Брокери вимагають, щоб клієнти надсилали імена користувачів та паролі з повідомленням CONNECT для перевірки з'єднання, якщо з'єднання не вдається. Підвищення безпеки для MQTT є постійними зусиллями. Різні методи авторизації постійно розробляються, експериментуються та випробовуються, щоб покращити аспекти безпеки MQTT.

Конфіденційність має першорядне значення для захисту системи, і цього можна досягти шляхом шифрування повідомлень, які будуть опубліковані на прикладному рівні. Шифрування може бути реалізовано як модель від клієнта до брокера, або як модель end-to-end. У моделі шифрування типу «клієнт-брокер» брокери розшифровують повідомлення, що передаються в заголовку теми, шифрують їх і надсилають клієнтам, які підписані на цю конкретну тему. Цей процес вимагає більшої обчислювальної потужності та енергії. Тоді як у моделі наскрізного шифрування брокеру не потрібно розшифровувати інформацію, що передається на різних темах. Брокер просто функціонує як поштове відділення, щоб відправити повідомлення відповідним одержувачам. Цей процес передбачає споживання меншої обчислювальної потужності та енергії. Комерційний і технічний інтерес до бездротових сенсорних мереж (WSN) зростає. Типова бездротова сенсорна мережа складається з великої кількості датчиків і виконавчих механізмів, що працюють від батарей, і мають обмежені обчислювальні ресурси. Ці пристрої повинні взаємодіяти один з одним.

Виявлено такі відкриті питання:

• Очевидно, що MQTT-SN, який є важливою функцією MQTT версії 5, має бути значною мірою прийнятий більшою кількістю реалізацій брокерів;

• тим не менш, оскільки MQTT-SN є наступником MQTT, він також успадковує попередні проблеми безпеки аутентифікації та шифрування, тому невеликий компроміс у зв'язку MQTT-SN може вплинути на всю інфраструктуру IoT з точки зору конфіденційності та доступності. Вирішення проблеми безпеки в MQTT дало б велику перевагу перед іншими доступними протоколами.

• Оскільки система IoT генерує велику кількість даних, різні брокери можуть розширити підтримку різних програм баз даних, пошуку з підтримкою AI та аналізу даних MQTT.

• Стандартна архітектура MQTT визначає лише одного брокера в системі, отже, вона не підходить для граничних додатків IoT і не може використовувати найкраще з багатоядерного середовища. Оскільки світ швидко рухається вперед у напрямку розподілених і граничних обчислень, необхідні додаткові дослідження щодо багатопоточних, масштабованих реалізацій MQTT, прототипів MQTT на основі периферій та безпеки MQTT-SN, щоб подолати цю прогалину.

• Проблема конфіденційності потоків даних, що відбуваються в системі IoT. Хоча існують деякі роботи, пов’язані з конфіденційністю в Інтернеті речей загалом, все ще необхідна велика кількість досліджень щодо рішень конфіденційності для окремих протоколів.

3.5 Реалізація інформаційної системи

Одна із областей застосування MQTT – це обмін між пристроями та програмами, що підключені до Інтернет. Для роботи з MQTT ми будемо використовувати тестовий клієнт та брокер. Для тестового приладу візьмемо http://test.mosquitto.org/gauge/, а тестовим клієнтом буде http://www.hivemq.com/demos/websocket-client/

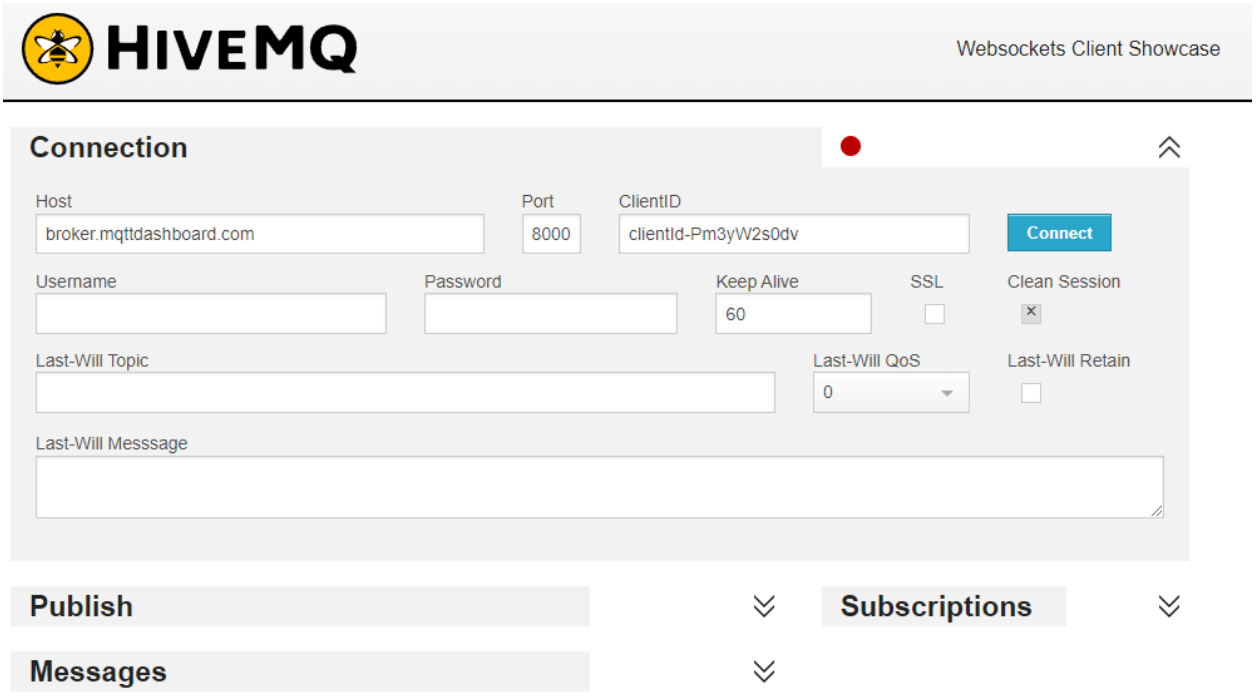


Рис. 3.2 Головна сторінка брокеру

Рис. 3.3 Тестовий прилад

На сторінці клієнта в полі Host введемо test.mosquitto.org, в полі Port 8080 і натискаємо Connect. З’являється напис Connect.

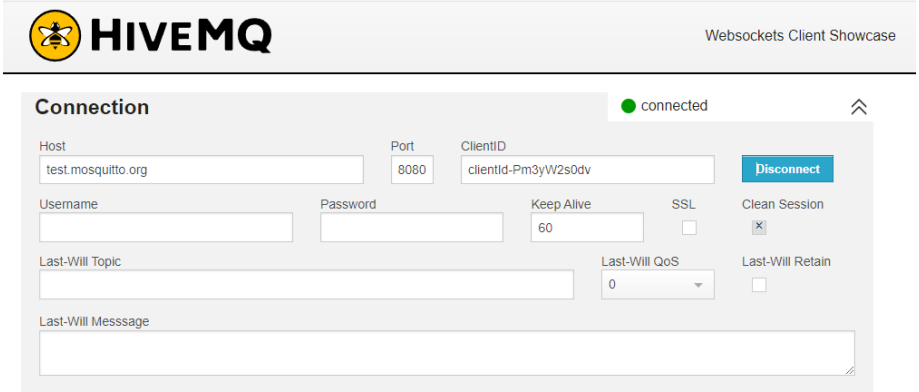


Рис. 3.4 Підключення

На панелі Publish Веб-сокет клієнта в полі Topic вводимо “temp/random”, QoS = 1 і в полі Massage будь-яке значення від 0 до 50, я задам 25. Натискаємо Publish та переходимо до тестового приладу. Відображається задане значення.

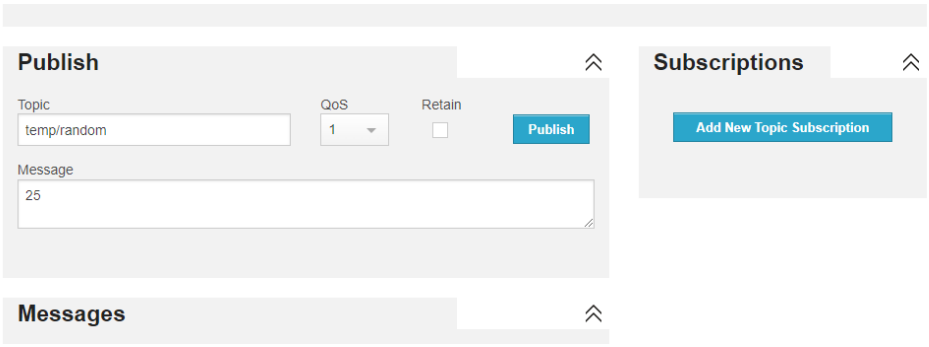


Рис. 3.5 Приклад заданих параметрів

Рис.3.6 Результат який відображається на приладі

Розглянемо графічний тестовий сервер і як змінюється графік при різних значеннях. Підєднуємось до веб-сокету клієнта. В полі Massage вносимо значення з періодичністю в 5 секунд(від 10 до 100).

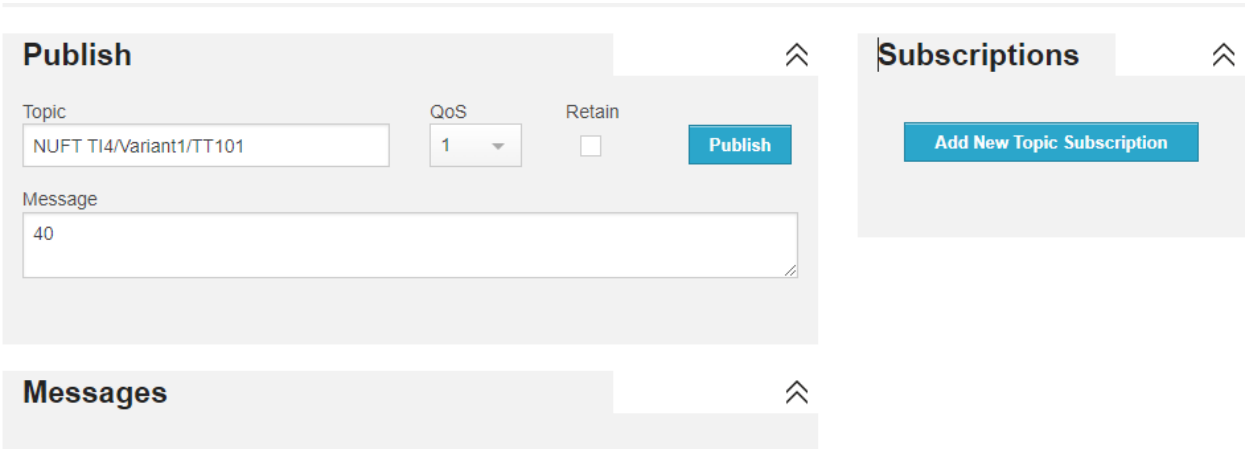


Рис. 3.7 Задані параметри

Оранжевим кольором показано як змінювались показники протягом часу.

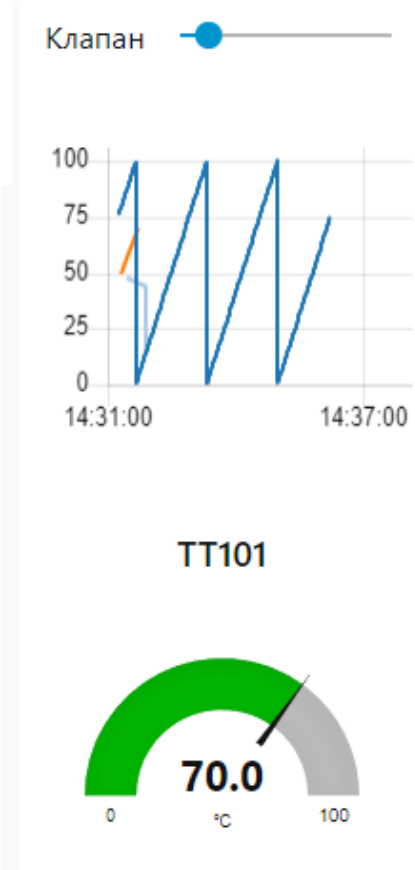


Рис.3.8 Візуалізація заданих параметрів

Натиснемо Add New Topic Subscribtion, підключимось до нашого веб-сокету. На тестовому прикладі декілька разів перемістимо верхній повзунок, після чого в меню Massage з’являється звіт про кожну зміну значень.

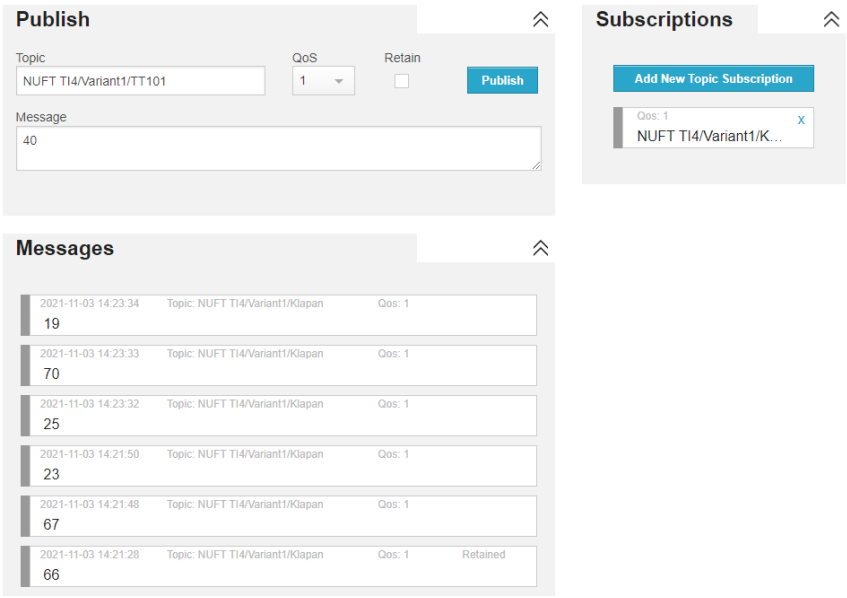


Рис. 3.9 Звіт про кожну зміну показників

Підпишемось на всі повідомлення з нашої тестової гілки додав іще одне значення в Subscription. Тепер всі значення тестового вебсокет-клієнта будуть відображатись в полі Massages

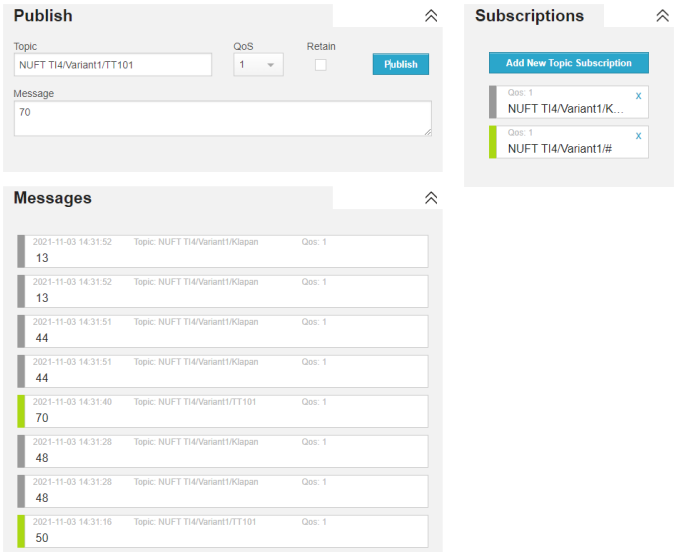


Рис. 3.10 Звіт про кожну зміну показників з декількох приладів

Перейдемо на http://test.mosquitto.org/sys/. Тут відображено дерево спеціальних систем які доступні на брокері test.mosquitto.org

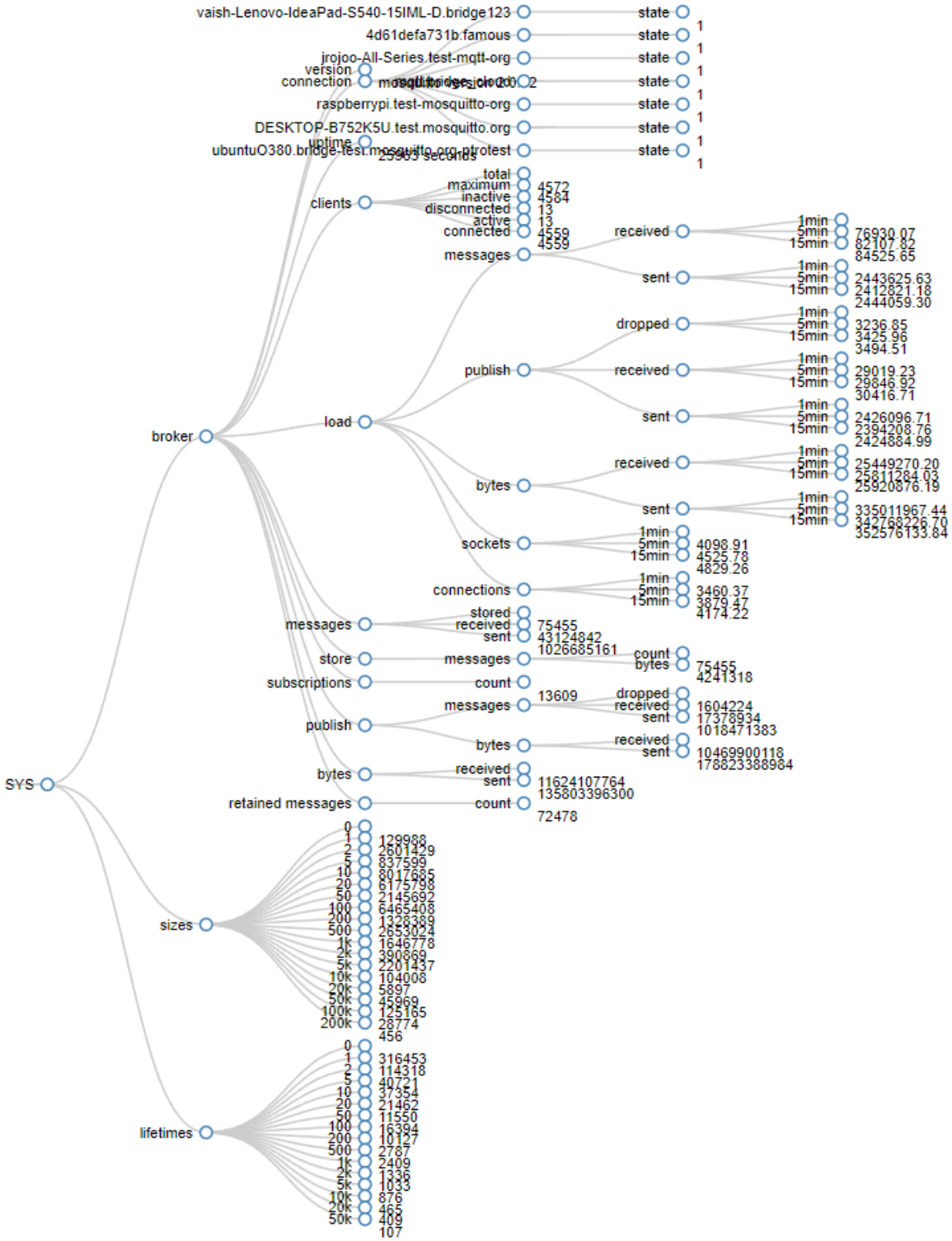


Рис. 3.11 Доступні можливі системи на брокері

Перейдемо до середи розробки Node-RED. Створимо новий вузол MQTT In. В налаштуваннях серверу додамо новий брокер з назвою mosquito і адресою серверу http://test.mosquitto.org

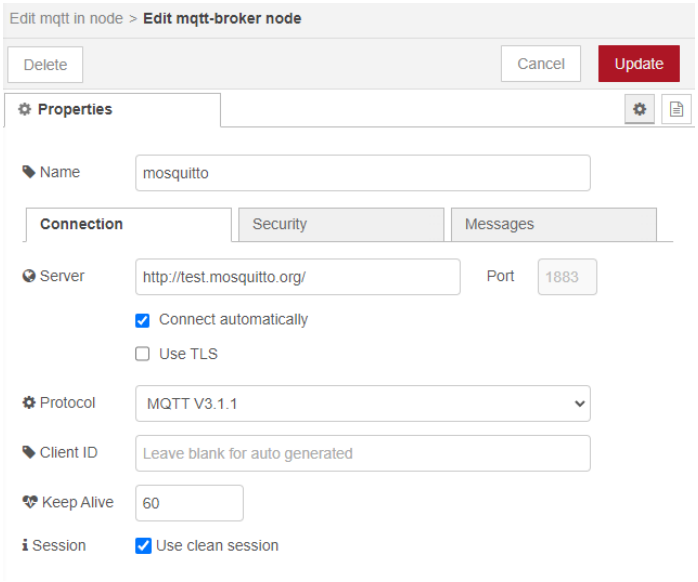


Рис. 3.12 Створення нового MQTT вузла за допомогою Node-RED

Додамо вузол Debug і з'єднаємо їх між собою. Зробимо розгортання і бачимо, що вузол MQTT in показує статус Connect, і справа в вікні Debug відображаються усі значення які змінюються в веб-сокеті.

Рис. 3.13 Звіт зміни показників в Node-RED

Використовуючи вузли Slider та Gauge реалізуємо зв'язок локального графічного інтерфейсу з віртуальним приладом на тестовому сервері.

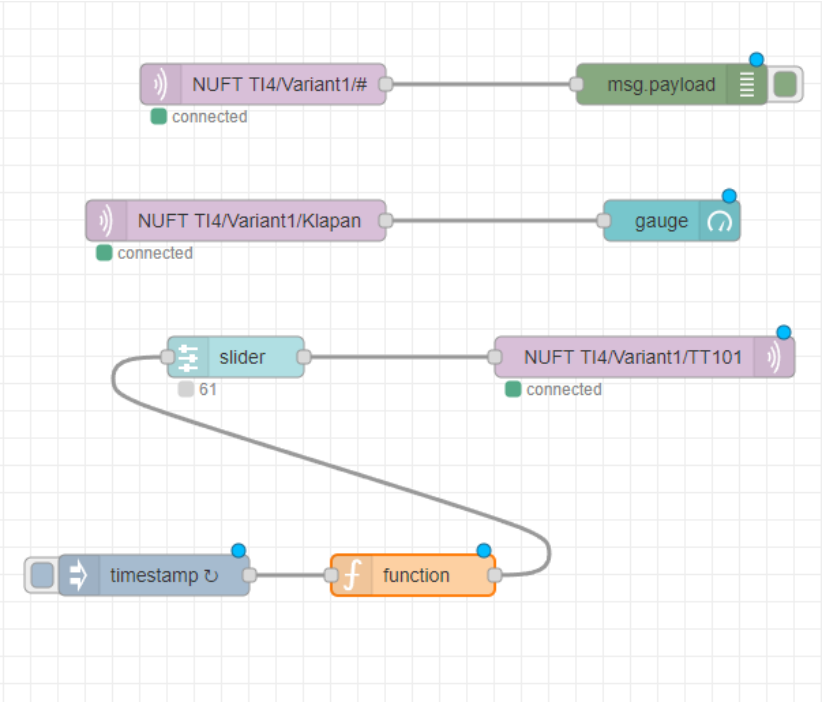


Рис. 3.14 Реалізована схема за допомогою Node-RED

До коду вузла Function внесемо наступні дані.

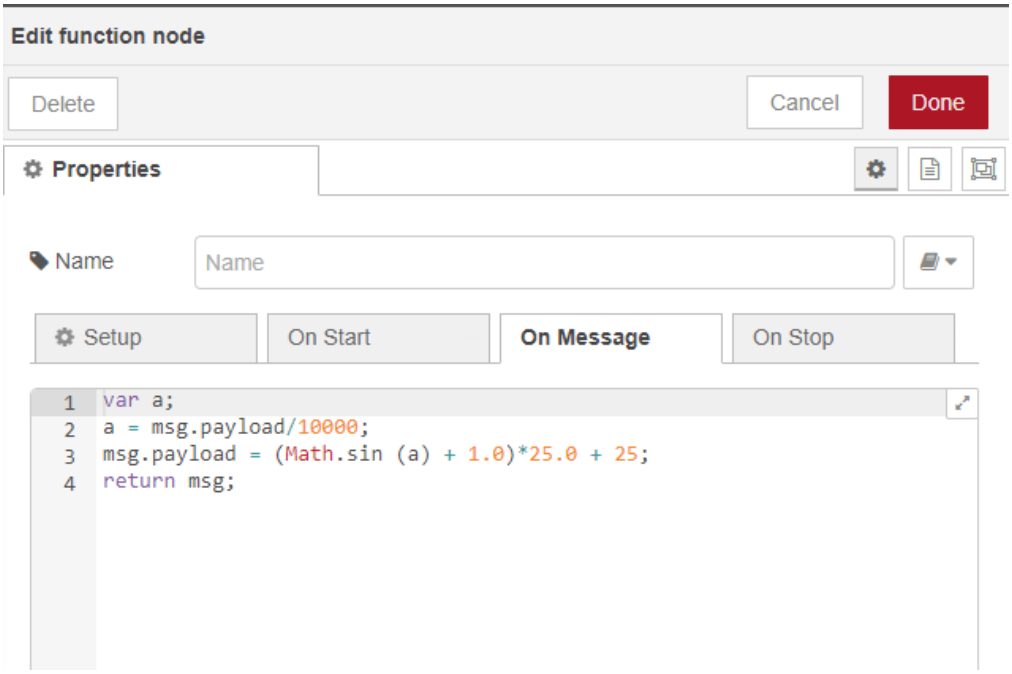


Рис. 3.15 Дані, що внесено у вузол Function

В результаті ми бачимо, як значення самостійно змінюються і показники відображаються на графіку.

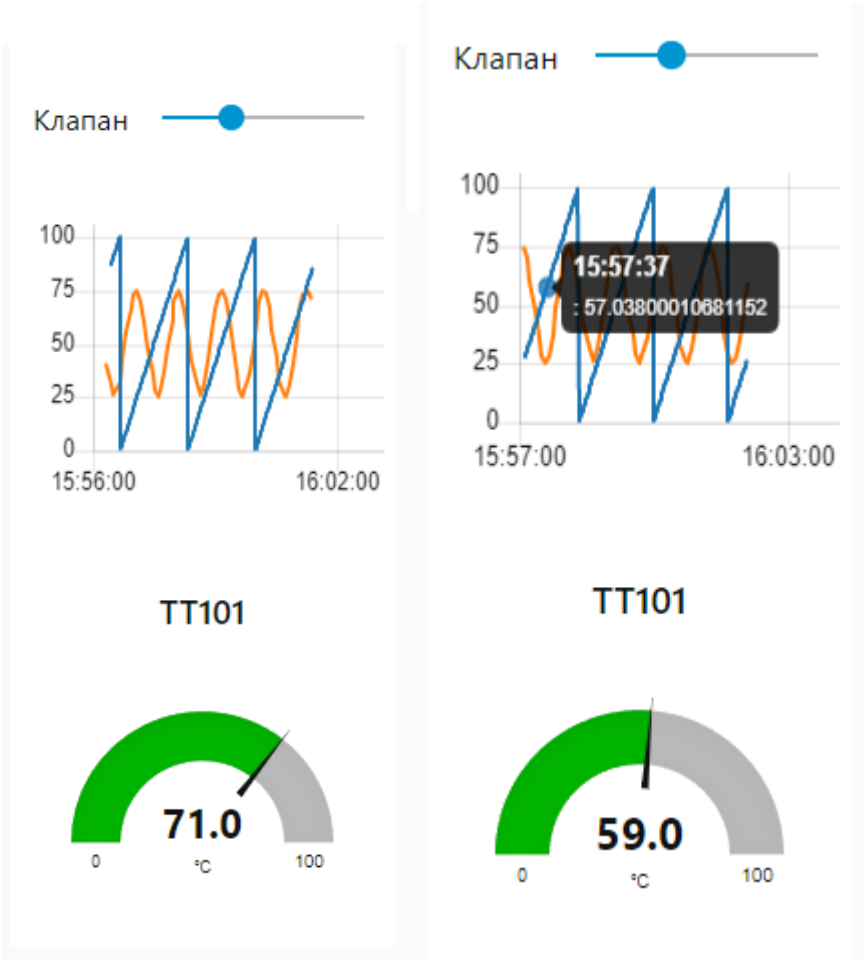


Рис. 3.16 – Візуалізація зміни показників

ВИСНОВКИ

Стандартизація Інтернету речей є важливим елементом для забезпечення ефективної та безпечної взаємодії між різними пристроями та системами, які входять до складу IoT. Стандартизація дозволяє різним виробникам створювати пристрої та рішення, які можуть ефективно взаємодіяти один з одним. Це важливо для розширення екосистеми IoT і для того, щоб користувачі могли вибирати пристрої від різних виробників, будучи впевненими в їхній сумісності. Стандартизація сприяє вдосконаленню безпеки в Інтернеті речей. Застосування єдиної системи безпеки допомагає уникати вразливостей та забезпечує стійкість систем. За наявності стандартів розробка нових пристроїв та рішень стає більш ефективною, оскільки виробники можуть використовувати загальні норми та специфікації. Це прискорює інновації та сприяє швидшому впровадженню нових технологій.

Стандартизація дозволяє зменшити ризики та витрати, пов'язані з розробкою та впровадженням нових рішень. Це робить ринок IoT більш привабливим для виробників та споживачів. Допомагає зробити Інтернет речей більш доступним та привабливим для глобального співтовариства. Вона сприяє розвитку міжнародних стандартів, які дозволяють пристроям та системам взаємодіяти без обмежень. Є ключовою для створення дієвої, безпечної та гнучкої інфраструктури, яка сприяє розвитку Інтернету речей та впровадженню його в різноманітних сферах життя.

Протоколи IoT є набором стандартів і правил, які визначають, як пристрої та системи в Інтернеті речей повинні обмінюватися інформацією. Вони відіграють ключову роль у забезпеченні ефективної та надійної комунікації між різними пристроями в масштабах IoT. Узагальнимо у висновках, чому протоколи IoT є важливим елементом цієї технологічної області. Протоколи дозволяють різним пристроям і системам взаємодіяти та обмінюватися інформацією. Вони визначають формати даних, правила комунікації та інші параметри, що забезпечують сумісність між різними пристроями в Інтернеті речей. Спрощують

передавання даних між пристроями, роблячи його ефективним та оптимізованим. Це особливо важливо в умовах обмежених ресурсів, які часто є характерними для пристроїв IoT.

Протоколи можуть включати механізми шифрування та автентифікації, що допомагають забезпечити безпеку обміну даними між пристроями. Є частиною стандартизованого підходу до розвитку IoT, що дозволяє виробникам розробляти сумісні пристрої та рішення. Це сприяє інтероперабельності між різними пристроями та системами. Багато пристроїв IoT працюють на обмежених джерелах енергії, тому протоколи можуть включати механізми для ефективного управління енергоспоживанням, такі як сплячий режим та оптимізацію передавання даних.

Протоколи враховують вимоги масштабованості, тобто їх можна використовувати у різних масштабах — від невеликих домашніх систем до великих міських мереж. Взаємодія пристроїв в Інтернеті речей вимагає встановлення загальних правил та стандартів, які дозволяють їм працювати разом ефективно та надійно. Протоколи IoT вирішують це завдання, створюючи основу для розвитку Інтернет-екосистеми речей.

ПЕРЕЛІК ПОСИЛАНЬ

1. https://eclipse.org/community/eclipse_newsletter/2014/february/article2.php

2. http://channel4it.com/publications/Internet-veshchey-25146.html

3. http://igate.com.ua/news/15786-chto-takoe-internet-veshhej-infografika

4. http://wiki.kspu.kr.ua/index.php/Технологія_Bluetooth

5. ISO/IEC 30141:2018.

6. ISO/IEC WD 30162 Internet of Things (IoT) — Compatibility requirements and model for devices within industrial IoT systems.

7. ISO/IEC 20924:2018.

8. HiveMQ Community Edition. Accessed: Aug. 23, 2020. [Online]. Available: https://github.com/680hivemq/hivemq-community-edition.

9. Recommendation Y.2060 – [Електронний ресурс] – електронні текстові дані – Режим доступу: https://www.itu.int/rec/T-REC-Y.2060-201206-I.

10. AMQP Advanced Message Queuing Protocol. Protocol Specification– [Електронний ресурс] – електронні текстові дані – Режим доступу:https://www.rabbitmq.com/resources/specs/amqp.

11. M. Houimli, L. Kahloul, and S. Benaoun. Formal specification, verification and evaluation of the MQTT protocol in the Internet of Things. In Mathematics and Information Technology, pages 214–221. IEEE, 2017.

12. K. Mladenov. Formal verification of the implementation of the MQTT protocol in IoT devices. Master thesis, University of Amsterdam, 2017.

13. R. Wang, L. Kristensen, H. Meling, and V. Stolz. Application of Model-based Testing on a Quorum-based Distributed Storage. In Proc. of PNSE'17, volume 1846 of CEUR Workshop Proceedings, pages 177–196, 2017.

14. J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, ''A survey of communication protocols for Internet of Things and related challenges of fog and cloud computing integration,'' ACM Comput. Surv., vol. 51, no. 6, pp. 1–29, Feb. 2019, doi: 10.1145/3292674.

15. MQTT Protocol. Accessed: Aug. 24, 2020. [Online]. Available: http://www.scalagent.com/en/jorammq-33/technology-36/mqtt-protocol.

16. Apache ActiveMQ. Accessed: Aug. 24, 2020. [Online]. Available: http://activemq.apache.org/

17. MQTT For Sensor Networks (MQTT-SN). Accessed: Aug. 24, 2020. [Online]. Available: https://www.mqtt.org/new/wp-content/uploads/ 2009/06/MQTT-SN_spec_v1.2.pdf.

18. Internet of Things for Industry and Human Application. In Volumes 1-3. Volume 1. Fundamentals and Technologies / V. S. Kharchenko (ed.) - Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. - 605p.

19. Internet of Things for Industry and Human Application. In Volumes 1-3. Volume 2. Modelling and Development /V . S . Kharchenko (ed.) - Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. - 547p.

20. Internet of Things for Industry and Human Application. In Volumes 1-3. Volume 3. Assessment and Implementation /V. S . Kharchenko (ed.) – Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. – 918 p.

21. A.P. Plakhtyeyev. E.V. Babeshko, V.A. Tkachenko, J.V. Zdorovets. Architectures and Embedded Platform Based development of Internet / Web of Things systems: Laboratory works / V.S. Kharchenko (edit.) - Ministry of Education and Science of Ukraine, National Aerospace University “KhAI”, 2019. – 147 p.

22. Internet of Things for Industry and Human Application. Fundamentals of Internet of Things / V.S. Kharchenko (ed.). – Ministry of Education and Science of Ukraine, National Aerospace University KhAI, 2019. – 95 p.

23. Butenko V.O., Odarushchenko O.N., Strjuk A.Y., Odarushchenko E.B., Mobile and hybrid Internet of Things based computing: Practicum / Kharchenko V.S. (Ed.) – Ministry of Education and Science of Ukraine, National Aerospace University “KhAI”, 2019. – 124 p.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)