Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повне найменування інституту, факультету)

Кафедра телекомунікацій
(повна назва кафедри)

«На правах рукопису»
УДК __004.89 _

До захисту допущено

В.о. завідувача кафедри

_____ Валерій ЯВІСЯ
(підпис) (ініціали, прізвище)

“___” _____20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації) 172 Телекомунікації та радіотехніка
(код і назва спеціальності)

За освітньо-професійною програмою Інженерія та програмування інфокомунікацій.

на тему: Узагальнена архітектура для системи Інтернету речей

Виконала: студентка __2__ курсу, групи ____ТЗ-81мн_____
(шифр групи)

_____Гончаренко Оксана Миколаївна_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник проф. каф. ТК, д.т.н. Кравчук С.О._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант__Розділ 3__ __доцент, к.т.н. Міночкін Д.А. __ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____доцент, к.т.н Осипчук С.О._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 рік

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут телекомунікаційних систем
(повна назва)

Кафедра телекомунікацій
(повна назва)

Спеціальність 172 Телекомунікації та радіотехніка
(код і назва)

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою Інженерія та програмування інфокомунікацій.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Валерій ЯВІСЯ
(підпис) (ініціали, прізвище)

«21» січня 2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Гончаренко Оксані Миколаївні
(прізвище, ім'я, по батькові)

1. Тема дисертації Узагальнена архітектура для системи Інтернету речей

науковий керівник дисертації Кравчук С.О., д.т.н., професор ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «30» березня 2020 р. № 923-с

2. Строк подання студентом дисертації 12.05.2020 р.

3. Об'єкт дослідження Системи Інтернету речей.

4. Предмет дослідження Архітектура для системи Інтернету речей.

5. Перелік завдань, які потрібно розробити:

1) Проаналізувати технології IoT.
2) Дослідити архітектуру та алгоритми роботи IoT.
3) Оцінити продуктивність архітектури AWS IoT.

6. Перелік графічного (ілюстративного) матеріал: Сервісно-орієнтована архітектура (SOA) для IoT; довідкова архітектура IoT; підсумок порівняння

платформ IoT; архітектура AWS IoT; основні блоки AWS IoT; Схематичне відображення бізнес-процесу.

7. Орієнтовний перелік публікацій

Міночкін Д. А. МЕТОДИ ВИКОРИСТАННЯ BLOCKCHAIN ТЕХНОЛОГІЙ В ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ / Д. А. Міночкін, Г.Ю. Бурлака, О.М.Гончаренко. // Збірник матеріалів Міжнародної науково-технічної конференції «ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ ». – 2020. – С. 321–323.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
3	Міночкін Д.А., доцент каф. ТК	21.01.2019	13.03.2020 р.

9. Дата видачі завдання 15.01.2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Вивчення та аналіз технологій IoT	22.01.2019 – 10.03.2019	
2	Огляд існуючих архітектур та алгоритмів роботи IoT	10.03.2019 – 20.05.2019	
3	Порівняння платформ IoT	20.05.2019 – 15.09. 2019	
4	Оцінка продуктивності архітектури AWS IoT	15.09.2019 – 25.01.2020	
5	Підготовка пояснювальної записки магістерської дисертації та ілюстративного матеріалу для її захисту	25.01.2020 – 10.05.2020	

Студент _____ О.М.Гончаренко
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ С.О.Кравчук
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Дипломна робота містить: 86 сторінок, 25 рисунків, 9 таблиць, 14 посилань.

Метою магістерської дисертації є аналіз технологій IoT, дослідження архітектурних рішень та розробка уніфікованої архітектури для систем Інтернету речей та оцінка її продуктивності.

Інтернет речей (IoT) — це унікальна платформа, яка стає дуже популярною з кожним днем. Самою причиною цього є прогрес в технологіях і їх здатність зв'язуватися з усім. Ця особливість отримання зв'язку сама по собі надала безліч можливостей і велику сферу розвитку. Той факт, що технології в різних областях розвивалися протягом багатьох років, є причиною того, що ми спостерігаємо швидку зміну форми, розміру і продуктивності різних інструментів, компонентів і продуктів, що використовуються в повсякденному житті. І це перевага спрощеної технології в поєднанні з такою платформою, як IoT, полегшує роботу, а також приносить користь як виробнику, так і кінцевому користувачеві. Інтернет речей дає нам можливість створювати ефективне управління, додатки для виробництва, рішення для порятунку життя, правильне вдосконалення і багато іншого. Технології IoT застосовується у порятунку життя, розумних містах, сільському господарстві, промисловості і т. д.

Ключові слова: Інтернет речей (IoT), узагальнена архітектура , AWS IoT, MQTT, HTTP, продуктивність мережі, пропускна здатність, середній час відгуку.

ABSTRACT

The diploma work consists of 86 pages, 25 figures, 9 tables, 14 references.

The purpose of graduation work is to analyze IoT technologies, study architectural solutions and develop a unified architecture for Internet of Things systems and evaluate its performance.

The Internet of Things (IoT) is a unique platform that is becoming very popular every day. The very reason for this is advances in technology and their ability to connect with everything. This feature of communication in itself has provided many opportunities and a large area of development. The fact that technology in various fields has evolved over the years is the reason that we are seeing a rapid change in the shape, size and performance of various tools, components and products used in everyday life. And this is the advantage of simplified technology in combination with a platform such as IoT, facilitates the work, and also benefits both the manufacturer and the end user. The Internet of Things gives us the ability to create effective management, applications for production, life-saving solutions, proper improvement and more. IoT technology is used in life-saving, smart cities, agriculture, industry, etc.

Keywords: Internet of Things (IoT), generalized architecture, AWS IoT, MQTT, HTTP, network performance, bandwidth, average response time.

Пояснювальна записка
до магістерської дисертації

на тему: Узагальнена архітектура для системи Інтернету речей

Київ – 2020 рік

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 9
ВСТУП 10
1. АНАЛІЗ ТЕХНОЛОГІЙ IOT 12
1.1 Що таке інтернет речей 12
1.2 Перспективи розвитку інтернету речей 13
1.3 Історія розвитку IoT 18
1.4 Технології і протоколи в IoT 23
1.4.1 Технологія Bluetooth 23
1.4.2 Технологія Zigbee 25
1.4.3 Технологія Z-Wave 26
1.4.4 Технологія Wi-Fi 26
1.4.5 Стільниковий зв'язок 27
1.4.6 Технологія NFC 27
1.4.7 LoRaWAN 28
1.4.8 Протоколи обміну повідомленнями IoT 28
1.4.9 Ідентифікаційні і відстежуючі технології 31
1.5 Інформаційна безпека та захист конфіденційності 32
2. ДОСЛІДЖЕННЯ АРХІТЕКТУР ТА АЛГОРИТМІВ РОБОТИ IoT 35
2.1 Стандартизація IoT 35
2.2 Детальний огляд сервіс-орієнтованої архітектури IoT 37
2.3 Узагальнена архітектура IoT 42
2.4 Порівняння платформ IoT 47
2.4.1 Архітектура FIWARE 48
2.4.2 Архітектура OpenMTC 49
2.4.3 Архітектура SiteWhere 51
2.4.4 Архітектура Webinos 52
2.4.5 Архітектура AWS IoT 55
2.4.6 Платформа IBM Watson IoT 56
2.4.7 Microsoft Azure IoT Hub 57
2.4.8 SmartThings 59

2.4.9 Підсумок порівняння 61
3. ОЦІНКА ПРОДУКТИВНОСТІ АРХІТЕКТУРИ AWS IoT 64
3.1 Основні компоненти AWS IoT 64
3.1.1 Принцип роботи AWS IoT 67
3.1.2 Принцип роботи сервісів 68
3.2 Фактори, що впливають на продуктивність Web-служб 71
3.3 Опис системи та розрахунок пропускної здатності 72
3.4 Середній час відгуку 80
ВИСНОВКИ 84
ПЕРЕЛІК ПОСИЛАНЬ 85

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AWS (Amazon Web Services) — Amazon веб-сервер.

CoAP (Constrained Application Protocol) — протокол обмеженого застосування.

HTTP (HyperText Transfer Protocol) — протокол передачі даних.

IoT (Internet of Things) — інтернет речей.

IoTIM (IoT integration middleware) — програмне забезпечення для інтеграції IoT.

JSON (JavaScript Object Notation) — текстовий формат обміну даними, заснований на JavaScript.

M2M (Machine-to-Machine) — технологія міжмашинної взаємодії.

MQTT (message queuing telemetry transport) — спрощений мережевий протокол .

PZH (Personal Zone Hub) центр персональної зони.

PZP (Personal Zone Proxy) — проксі-сервер персональної зони.

QoS (Quality of service) — якість обслуговування.

RFID (Radio Frequency IDentification) — радіочастотна ідентифікація.

SOA (Service Oriented Architecture) — сервісно-орієнтована архітектура.

TCP/IP (Transmission Control Protocol / Internet Protocol) — набір протоколів мережі Інтернет.

UDP (User Datagram Protocol) — один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання.

UUID — універсальний унікальний ідентифікатор.

WSDL (Web Services Description Language) — мова опису веб-сервісів і доступу до них, заснована на мові XML.

WSN (Wireless sensor network) — бездротова сенсорна мережа.

ВСТУП

Актуальність роботи. Інтернет речей (IoT) — це унікальна платформа, яка стає дуже популярною з кожним днем. Самою причиною цього є прогрес в технологіях і їх здатність зв'язуватися з усім. Ця особливість отримання зв'язку сама по собі надала безліч можливостей і велику сферу розвитку. Той факт, що технології в різних областях розвивалися протягом багатьох років, є причиною того, що ми спостерігаємо швидку зміну форми, розміру і продуктивності різних інструментів, компонентів і продуктів, що використовуються в повсякденному житті. І це перевага спрощеної технології в поєднанні з такою платформою, як IoT, полегшує роботу, а також приносить користь як виробнику, так і кінцевому користувачеві. Інтернет речей дає нам можливість створювати ефективне управління, додатки для виробництва, рішення для порятунку життя, правильне вдосконалення і багато іншого. Технології IoT застосовується у порятунку життя, розумних містах, сільському господарстві, промисловості і т. д. [1].

IoT привертає все більше уваги. Загальною метою є взаємозв'язок фізичного з цифровим світом. Тому фізичний світ вимірюється датчиками і переводиться в оброблювані дані, а дані повинні бути переведені в команди, які виконуються датчиками. Через зростаючий інтерес до IoT, кількість платформ, призначених для підтримки IoT, значно зросла. В результаті різних підходів, стандартів та випадків використання існує велика різноманітність та неоднорідність платформ IoT. Це призводить до труднощів у розумінні, виборі та використанні відповідних платформ. У даній роботі вирішуються ці питання, через детальний аналіз декількох найновіших платформ IoT з метою сприяння розумінню базових понять, подібності та відмінностей між ними . Буде показано, що різні компоненти різних платформ можна віднести до абстрактної довідкової архітектури та проаналізувати ефективність цього відображення.

Таким чином, **актуальність** розробки уніфікованої архітектури для системи Інтернету речей полягає у тому, що це полегшить розробку нових систем та мереж інтернету речей.

Метою магістерської дисертації є розробка уніфікованої архітектури для систем Інтернету речей та оцінка її продуктивності.

Для досягнення поставленої мети, необхідно виконати наступні **задачі:**

1) Проаналізувати технології IoT.
2) Дослідити архітектури та алгоритми роботи IoT.
3) Оцінити продуктивність архітектури AWS IoT.

Об'єктом дослідження є системи Інтернету речей.

Предметом дослідження є архітектура для системи Інтернету речей.

Методи дослідження, що використані в роботі — це загальнонаукові методи, такі як: аналіз, порівняння та моделювання.

Наукова новизна одержаних результатів: проведено узагальнення архітектурного рішення, що дозволяє його використання для побудови проектів різного призначення та інтеграції з існуючими.

Практична цінність роботи. Узагальнена система спрощує створення систем інтернету речей для розробників та дозволяє просту інтеграцію з існуючими такими системами.

Дана робота має важливе значення для подальшого використання в навчальному курсі «Інтернет речей» для підвищення рівня знань студентів Інституту телекомунікаційних систем, що навчаються за спеціальністю «172 Телекомунікації та радіотехніка».

Апробація результатів роботи. Робота була апробована на наступній конференції: XIV Міжнародна Науково-технічна конференція «Перспективи телекомунікацій 2020» (м.Київ, НТУУ «КПІ ім. І. Сікорського»), яка проходила 13-17 квітня 2020 року, у доповіді на тему «МЕТОДИ ВИКОРИСТАННЯ BLOCKCHAIN ТЕХНОЛОГІЙ В ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМАХ».

1. АНАЛІЗ ТЕХНОЛОГІЙ IOT

1.1 Що таке інтернет речей

Інтернет з розвитком технологій став проникати в звичні нам речі — це невід'ємна частина нашого життя, де в майбутньому вже ніхто не зможе уявити, як це жити без таких зручностей, і одним з таких відгалужень глобальної мережі став інтернет речей. Концепція інтернету речей з'явилася в кінці минулого століття, основною метою якої є об'єднання і взаємодія один з одним різних, звичних у повсякденному житті предметів за допомогою різних каналів зв'язку, наприклад, з використанням інтернет протоколу IP.

Багато людей кажуть, що інтернет речей це такий "новий тренд", але мало хто може пояснити що це таке насправді. Інтернет речей, за даними Gartne (дослідницька компанія з США), і справді є дуже "трендовим", він знаходиться на самій вершині "хайп-циклу" поряд з іншими технологіями, такими як перетворення "мова-мова", машинне навчання, крипто валюта , автономні засоби пересування, тому не дивно, що про нього чув кожен.

Інтернет речей є новою стадією розвитку інтернету, в даний момент відбувається перехід саме до нового поданням інтернету. Важливими відмінностями від простого інтернету є: фокус на речах, набагато більшу кількість підключених предметів, менші розміри підключаються об'єктів, фокус на зчитуванні інформації, необхідність створення інфраструктури і стоїть питання розробки нових стандартів. Інтернет речей (IoT, Internet of Things) — це мережа фізичних об'єктів, які мають вбудовані засоби, які дозволяють взаємодіяти один з одним за допомогою різних протоколів зв'язку. Деякі визначення інтернету речей говорять про те, що це стадія розвитку інтернету, коли до неї включено більше саме речей, а не людей. За цим визначенням виходить, що людство вже перейшло до нього приблизно в 2009 році і з кожним роком кількість пристроїв росте з величезною швидкістю.

Як правило в мережу можна об'єднати різні девайси: датчики, прилади, пристрої і т.д. Основна мережа для взаємодії, як можна здогадатися, є

інтернет і його протокол IP. Існують і інші різні протоколи бездротового зв'язку між "речами" - Wi-Fi і LAN, Bluetooth, RFID, Bluetooth Low Energy (з низьким енергоспоживанням), Wi-Fi HaLow, LPWAN, Z-Wave, ZigBee, LoWPAN і безліч інших [2].

1.2 Перспективи розвитку інтернету речей

Інтернет речей можна застосовувати в багатьох областях, навіть там, де не кожен міг би це уявити. Як правило IoT підрозділяється на кілька сегментів: для споживача і для бізнесу або промисловості. Зараз елементи інтернету речей можна зустріти всюди, але ж не кожен це знає, що він уже "підключений" до цієї мережі. Наприклад, безліч людей використовують різні пристрої-носії, на зразок розумних годиннників, фітнестрекерів, у когось є "розумний телевізор" (зазвичай називається Smart TV і т.п.), хтось використовує девайси для відстеження діяльності та місцезнаходження свого вихованця, в деяких будинках застосовується "Розумний будинок", коли кожен пристрій об'єднано в одну загальну мережу - освітлення, система водопостачання, датчики, вхідні двері, телефон і т. д. Деякі компанії вже зараз виробляють "розумний одяг", який може відстежувати стан людини і навколишнього його середовища і видавати якісь рекомендації або відразу адаптуватися, наприклад, за допомогою нагрівальних елементів створити комфортну температуру для людини, а також можна за допомогою супутника і мобільних мереж відстежувати шлях, по якому відбувалося переміщення. Зараз деякі військові відомства намагаються створювати костюми для піхоти, екзоскелети, які допоможуть солдатам підтримувати зв'язок, виводити різну інформацію про противника або власне тіло на вбудований дисплей і т. п.

У бізнес сфері вже зараз використовується інтернет речей і проходять тестування нові технології. Зараз в деяких країнах світу проводяться запуски безпілотних автомобілів (зазвичай водій в машині є, але не для водіння, а для відстеження правильності пересування і для позаштатних ситуацій), які можуть перевозити пасажирів, орієнтуючись в просторі за допомогою стандартних технологій позиціонування (супутникова навігація), датчиків

зору, різних стабілізаторів і далекомірів, камер, сенсорів, як правило тут же застосовується машинне навчання, нейромережі — машина сама себе навчає і навчається на своїх же помилках. Інтернет речей можна використовувати для створення розумного міста, де дана технологія значно полегшить управління майном міста, підвищить його ефективність, оптимізує витрати, поліпшить якість і продуктивність різних міських служб, що негайно підвищить якість життя міських жителів і загальний добробут. Крім міської сфери і транспорту, інтернет речей знайшов відмінне застосування в сфері геолокації по торговим центрам, так звані "маячки" бікони (beacon), які при вході в торговий центр зв'язуються з мобільним девайсом і можуть повідомляти якусь інформацію про магазин, рахувати відвідувачів певного магазину, повідомляти про знижки. Також можна використовувати їх для навігації людей з обмеженими можливостями, пошуку різних речей — ключів і багажу, навігації по величезних торгових центрах, де система навігації за допомогою супутника не працює належним чином, безконтактної реєстрації на заходах, для контролю відвідуваності навчального закладу (актуально для шкіл і університетів) і т. д. На основі інтернету речей можуть бути реалізовані також: "розумна планета", "розумний будинок", "розумна енергетика", "розумна медицина", "розумне виробництво" та інші [2].

IoT надає нові можливості для поліпшення охорони здоров'я. Всі об'єкти в системах охорони здоров'я (люди, обладнання, медикаменти і т. д.), що знаходяться на повсюдному виявленні, сприйнятті і комунікації IoT, можуть постійно відслідковуватися і контролюватися. Завдяки глобальному з'єднанню вся інформація, пов'язана з охороною здоров'я (логістика, діагностика, лікування, одужання, лікування, управління, фінанси і навіть повсякденна діяльність), може ефективно збиратися, управлятися і спільно використовуватися. Наприклад, частота серцевих скорочень пацієнта може час від часу вимірюватися датчиками, а потім відправлятися в кабінет лікаря. Використовуючи персональні обчислювальні пристрої (ноутбук, мобільний телефон, планшет і т. д.) і мобільний доступ в Інтернет (WiFi, 3G, LTE і т. д.), медичні послуги на основі IoT можуть бути мобільними і персоналізованими.

Широке поширення послуг мобільного інтернету прискорило розвиток послуг IoT на дому (IHH). Проблеми безпеки та конфіденційності є двома основними проблемами.

Сьогоднішній FSC (Forest Stewardship council) надзвичайно поширений і складний. Він має великий географічний і часовий масштаб, складні операційні процеси і велика кількість зацікавлених сторін. Складність викликала безліч проблем в галузі управління якістю, операційної ефективності та суспільної безпеки харчових продуктів. Технології IoT пропонують багатообіцяючі можливості для вирішення завдань відстеження, видимості і керованості. Він може охоплювати FSC так званим способом від ферми до тарілки, від точного землеробства до виробництва, обробки, зберігання, розподілу та споживання продуктів харчування. В майбутньому очікуються більш безпечні, більш ефективні і стійкі FSC. Типове рішення IoT для FSC (так зване Food-IoT) складається з трьох частин: а) польові пристрої, такі як вузли WSN, зчитувачі / мітки RFID, термінали призначеного для користувача інтерфейсу і т. д.; б) магістральна система, така як бази даних, сервери і багато видів терміналів, з'єднаних розподіленими комп'ютерними мережами і т. д.; і в) інфраструктур зв'язку, таким як WLAN, стільниковий зв'язок, супутниковий зв'язок, лінія електропередачі, Ethernet і т. д. Астатична система забезпечує пропускну здатність єдиної мережі, всі ці елементи можуть бути розподілені по всьому FSC. Крім того, він також пропонує ефективні сенсорні функції для відстеження та моніторингу процесу виробництва продуктів харчування. Величезний обсяг необроблених даних може бути додатково проаналізовано і проаналізовано для поліпшення бізнес-процесу і підтримки прийняття рішень. Аналітика великих даних може використовуватися для вирішення проблеми аналізу величезної кількості даних, зібраних з FSC.

Безпека шахт є серйозною проблемою для багатьох країн через умови роботи в підземних шахтах. Щоб запобігти і зменшити кількість нещасних випадків на рудниках, необхідно використовувати технології IoT для виявлення сигналів про бідування в шахтах, щоб зробити можливим раннє

попередження, прогнозування лих і підвищення безпеки підземного виробництва. Використовуючи RFID, WiFi і інші технології і пристрої бездротового зв'язку для забезпечення ефективного зв'язку між наземним і підземним обладнанням, гірничодобувні компанії можуть відстежувати місцезнаходження підземних шахтарів і аналізувати критичні дані безпеки, зібрані з датчиків, для посилення заходів безпеки. Ще одним корисним додатком є використання хімічних і біологічних датчиків для раннього виявлення захворювань та діагностики підземних шахтарів, так як вони працюють в небезпечному середовищі. Ці хімічні та біологічні датчики можуть використовуватися для отримання біологічної інформації від людського тіла і органів і для виявлення небезпечного пилу, шкідливих газів і інших небезпечних факторів навколишнього середовища, які можуть стати причиною нещасних випадків. Проблема полягає в тому, що бездротові пристрої мають потребу в живленні і можуть привести до детонації газу в шахті. Необхідні додаткові дослідження щодо характеристик безпеки пристроїв IoT, які використовуються в гірничому виробництві.

IoT буде грати все більш важливу роль у транспортній та логістичної галузях. Оскільки все більше і більше фізичних об'єктів оснащуються штрих-кодами, RFID-мітками або датчиками, транспортні та логістичні компанії можуть проводити моніторинг в реальному часі переміщення фізичних об'єктів з пункту відправлення в пункт призначення по всьому ланцюжку поставок, включаючи виробництво, доставку, розподіл, і так далі. Крім того, IoT, як очікується, запропонує перспективні рішення для трансформації транспортних систем і автосервісів. Оскільки транспортні засоби мають все більш потужні можливості розпізнавання, створення мереж, зв'язку та обробки даних, технології IoT можуть використовуватися для розширення цих можливостей і розподілу недостатньо використовуваних ресурсів між транспортними засобами на парковці або в дорозі. Наприклад, технології IoT дозволяють відслідковувати існуюче розташування кожного транспортного засобу, відстежувати його рух і прогнозувати його майбутнє місце розташування. Нещодавно інтелектуальна інформаційна система (система

iDrive), розроблена BMW, використовувала різні датчики і мітки для моніторингу навколишнього середовища, наприклад, відстежування місцеположення транспортного засобу і стану дороги для визначення напрямку руху. Також розроблено інтелектуальну систему моніторингу для контролю температури / вологості в рефрижераторах з використанням RFID-міток, датчиків і технології бездротового зв'язку. У найближчому майбутньому ми побачимо розробку автомобільного автопілота, який може автоматично виявляти пішоходів або інші транспортні засоби і приймати необхіднне рульове управління, щоб уникнути зіткнень в міру необхідності. Безпека і захист конфіденційності важливі для широкого використання IoT в сфері транспорту і логістики, оскільки багато водіїв транспортних засобів стурбовані витоком інформації і порушенням конфіденційності. Необхідні розумні зусилля в області технологій, законодавства і регулювання для запобігання несанкціонованому доступу або розкриття даних про конфіденційність.

IoT використовується в галузі пожежної безпеки для виявлення потенційної пожежі та раннього попередження про можливі пожежі. У Китаї до продуктів пожежогасіння прикріплюються RFID-мітки і/або штрих-коди для створення загальнонаціональних баз даних і систем управління інформацією про продукти пожежогасіння. Використовуючи RFID-мітки, мобільні зчитувачі RFID, інтелектуальні відеокамери, сенсорні мережі, і мережі бездротового зв'язку, протипожежний орган або пов'язані з ним організації можуть виконувати автоматичну діагностику для реалізації моніторингу навколишнього середовища в реальному часі, раннього оповіщення про пожежу та аварійного порятунку в міру необхідності. Дослідники в Китаї також використовують технології IoT для створення систем автоматичної пожежної сигналізації, щоб підняти управління пожежогасінням та управління надзвичайними ситуаціями на новий рівень. Інфраструктура додатків IoT, використовуваних для управління в надзвичайних ситуаціях в Китаї містить рівень сприйняття, рівень передачі, рівень підтримки, рівень платформи і рівень додатків та була розроблена для

інтеграції як локальних, так і галузевих аварійних систем. Встановлення стандартів для реалізації Fire IoT є зараз нагальною проблемою [3].

Можна оцінити кількість сфер застосування інтернету речей і сказати прямо — перспективи є, причому дуже хороші. Інтернет речей зараз переживає період дуже бурхливого зростання. Аналітики оцінюють цей ринок також вкрай перспективним і дуже вдалим для капіталовкладень, які ростуть з кожним роком і до 2020 - 2030 років стануть більш від 1 до 2 трлн. доларів США по всьому світу. Наприклад, мобільні телефони відносяться до інтернету речей і вже зараз це найпопулярніша і найбільш численна категорія пристроїв в даній концепції. Саме інтернет речей допоможе людству робити нові відкриття, проводити дослідження, аналізувати безліч даних — все для подальшого процвітання протягом багатьох століть. Інтернет речей в деяких країнах включений в список проривних і інноваційних технологій — це ще раз доводить той факт, що перспектив розвитку безліч [2].

1.3 Історія розвитку IoT

Інтернет речей (IoT) лише нещодавно увійшов у наше повсякденне життя. Він оточує нас скрізь, куди ми їздимо: підключені автомобілі, що рухаються на вулиці, пристрої домашньої автоматики, розташовані в будинку, смарт-офісні датчики, вбудовані в робоче місце, та фітнес-трекери, що носяться на наших кузовах. Як повідомляє Statista (онлайн-портал статистики), вони створюють масову екосистему з 26,66 мільярдів взаємопов'язаних речей, які мають неабиякий вплив на суспільства та економіку у всьому світі.

Але світ не завжди був таким. До 1999 року термін "Інтернет речей" ще не існував. Отже, як саме Інтернет речей розвивався так швидко і став такою звичайною модницею, і які віхи ознаменували Інтернет речей у глобальному масштабі? Щоб відповісти на ці запитання, давайте зануримось у коріння цієї неймовірної технології.

Сама концепція підключених пристроїв датується 1832 р., коли був спроектований перший електромагнітний телеграф. Телеграф забезпечив

прямий зв'язок між двома машинами через передачу електричних сигналів. Однак справжня історія IoT почалася з винайдення Інтернету - дуже важливого компонента - в кінці 1960-х, який потім швидко розвивався протягом наступних десятиліть.

Можливо у це важко повірити, але першим підключеним пристроєм був торговий автомат Coca-Cola, розташований в Університеті Карнегі-Дині, яким керували місцеві програмісти (1980-ті роки). Вони інтегрували мікроперемикачі в машину і використовували ранню форму Інтернету, щоб перевірити, чи охолоджуючий прилад підтримує напої досить холодними та чи є в наявності банки з колою. Цей винахід сприяв подальшим дослідженням у цій галузі та розвитку взаємопов'язаних машин у всьому світі.

У 1990 році Джон Ромкі вперше підключив тостер до Інтернету за допомогою протоколу TCP / IP. Через рік вчені Кембриджського університету виступили з ідеєю використати перший прототип веб-камери для контролю кількості кави, наявного в кавовій локальній лабораторії комп'ютерної лабораторії. Вони запрограмували веб-камеру три рази в хвилину фотографувати кавовий горщик, а потім надсилати зображення на місцеві комп’ютери, таким чином дозволяючи всім бачити, чи є кава.

1999 рік був легко одним з найбільш значущих для історії IoT, оскільки Кевін Ештон ввів термін "Інтернет речей". Екстон-технолог (visionary technologist), Ештон, виступав з презентацією для Procter & Gamble, де описав IoT як технологію, яка з'єднала кілька пристроїв за допомогою тегів RFID для управління ланцюгами поставок. Він спеціально використовував слово "Інтернет" у назві своєї презентації, щоб привернути увагу аудиторії, оскільки Інтернет на той час просто став великою справою. Незважаючи на те, що його уявлення про підключення пристроїв на основі RFID відрізняється від сьогоднішнього IT-інтернету, прорив Ештона відіграв важливу роль в історії історії речей та технологічного розвитку в цілому.

На початку 21 століття термін "Інтернет речей" набув широкого вжитку в засобах масової інформації, коли про нього згадували представники

"Гардіан", "Форбс" та "Бостонський глобус". Інтерес до технології IoT постійно збільшувався, що призвело до 1-ї Міжнародної конференції з Інтернету речей, що відбулася у Швейцарії у 2008 році, де учасники 23 країн обговорили RFID, бездротовий зв'язок короткого діапазону та сенсорні мережі.

Крім того, кілька основних розробок сприяли еволюції IoT. Один був холодильником, підключеним до Інтернету, який був представлений LG Electronics у 2000 році, що дозволяє його користувачам робити покупки в Інтернеті та здійснювати відеодзвінки. Ще однією істотною розробкою став маленький робот у формі кролика на ім’я Набазтаг, створений у 2005 році, який міг розповісти останні новини, прогноз погоди та зміни на фондовому ринку.

За даними Cisco, тоді ще кількість взаємопов'язаних пристроїв перевершувало кількість людей на Землі (рис.1.1).

Cisco IBSG, April 2011

World population	6.3 billion	6.8 billion	7.2 billion	7.6 billion
Connected devices	500 million	12.5 billion	25 billion	50 billion
Devices per person	0.08	1.84	3.47	6.58
	2003	2010	2015	2020

More connected devices than people

Рис.1.1 Інтернет речей . Як наступна еволюція інтернету змінює все

Бум IoT був підтриманий його доповненням до циклу хайпу (ажіотажу) Hartner Hype для нових технологій у 2011 році. У тому ж році IPv6 — протокол мережевого рівня, який є центральним для IoT — був запущений публічно.

З того часу взаємопов'язані пристрої стали широко поширеними та звичними в нашому повсякденному житті. Глобальні технологічні гіганти, такі як Apple, Samsung, Google, Cisco та General Motors, зосереджують свої зусилля на виробництві датчиків і пристроїв IoT — від взаємопов’язаних

термостатів та розумних окулярів до автошкол, що керують авто. IoT знайшов свій шлях майже до кожної галузі: виробництво, охорона здоров'я, транспорт, нафта та енергетика, сільське господарство, роздрібна торгівля та багато іншого. Цей драматичний зсув переконав нас, що революція IoT зараз саме тут.

На сьогоднішній день платформи IoT зберігають міцну позицію серед провідних тенденцій циклу хайпу разом з віртуальними помічниками, підключеними будинками та власними автомобілями 4-го рівня. Технологія досягне свого піку продуктивності за 5–10 років (рис.1.2).

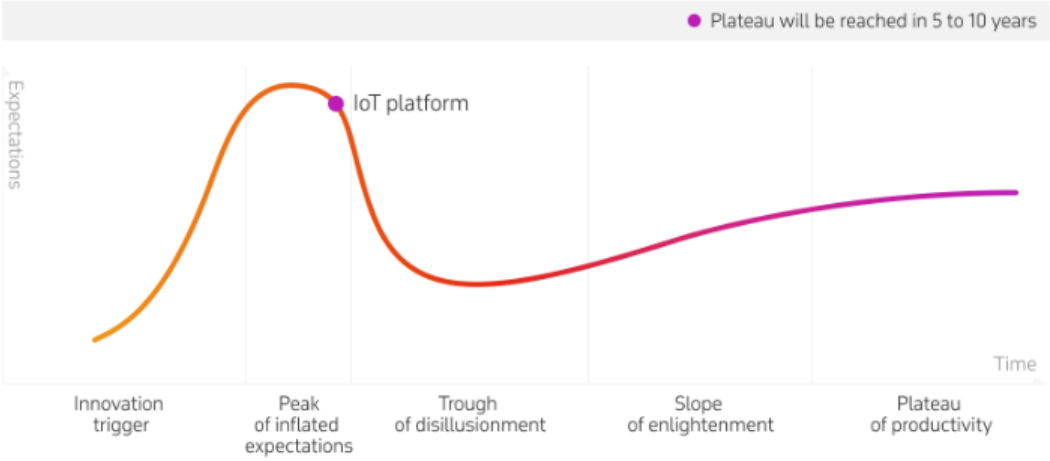

Рис. 1.2 Цикл хайпу нових технологій станом на 2018 рік

Враховуючи цей стрімкий темп розвитку, IoT незабаром домінує у світі. У 2019 році дослідницька компанія Gartner передбачила, що корпоративний і автомобільний IoT ринок зросте до 5,8 мільярда кінцевих точок у 2020 році, що означатиме збільшення на 21% порівняно з 2019. Все, що можна підключити, буде підключено, тим самим утворюючи комплексну цифрову систему, в якій всі пристрої спілкуються з людьми і один одного.

Ось кілька найважливіших факторів, що сприяють швидкому розширенню IoT:

- Падіння вартості сенсора
- Зниження витрат на збір і зберігання даних завдяки хмарним рішенням
- Широко розширюється інтернет-зв'язок
- Збільшення обчислювальної потужності

- Збільшення проникнення смартфонів і планшетів

Безсумнівно, швидке зростання IoT в корені змінить світ, в якому ми живемо. Уявіть, як підключений автомобіль отримає доступ до вашого графіку роботи, і повідомить колег про те, що ви запізнилися на збори, якщо він потрапить у затор по дорозі на роботу.

Наше неминуче взаємозалежне майбутнє, без сумніву, принесе багато користі і цікавих можливостей для людей. Однак і у нього будуть свої проблеми.

Незважаючи на те, що потужний IoT сам по собі, він надає набагато більше можливостей в поєднанні з іншими технологіями, такими як блокчейн, штучний інтелект, машинне навчання, великі дані, AR / VR і хмарні і прикордонні обчислення. В майбутньому буде набагато більше змішаних рішень.

Наприклад, застосування блокчейна в IoT допоможе децентралізувати мережі і забезпечити більш надійну передачу даних між взаємопов'язаними пристроями. Блокчейн вже є провідним трендом IoT, і від об'єднання цих двох технологій напевно буде більше користі.

Майбутнє IoT тісно пов'язане з штучним інтелектом і машинним навчанням. Приклади додатків включають в себе інтелектуальне обслуговування взаємопов'язаних пристроїв, оптимізацію виробничих процесів і пристрої «розумний будинок», які вивчають ваші уподобання. У найближчому майбутньому пристрої IoT будуть не тільки повідомляти інформацію, а й приймати автономні рішення і самі стануть розумнішими завдяки використанню методів машинного навчання.

Хмарні і периферійні обчислення будуть як і раніше невід'ємною частиною сховища даних IoT в 2019 році і в наступні роки, оскільки експерти прогнозують, що периферійні обчислення скоро отримають ще більшу популярність.

Незважаючи на зусилля численних урядів щодо посилення правил безпеки IoT та вдосконалення механізмів захисту взаємозв'язку, безпека даних та проблеми конфіденційності ніколи не зменшаться. Кіберзлочинці

використовують все більш складні тактики, щоб знайти вразливості на підключених пристроях, тим самим отримуючи доступ до приватної інформації. Як результат, споживачі та організації все більше стурбовані безпекою IoT і розглядають це як провідну перешкоду на шляху широкого впровадження IoT [4].

1.4 Технології і протоколи в IoT

Існує кілька комунікаційних протоколів і технологій, що використовуються в інтернеті речей. Деякими з основних технологій і протоколів IoT (протоколи зв'язку IoT) є Bluetooth, Wifi, радіопротоколи, LTE-A і WiFi-Direct. Ці протоколи зв'язку IoT задовольняють і відповідають конкретним функціональним вимогам системи IoT.

1.4.1 Технологія Bluetooth

Одним з найбільш перспективних і важливих стандартів для передачі даних в IoT є технологія Bluetooth . Розробка технології почалася в 1989 році компанією Ericsson. Пізніше був створений консорціум, в яких входять безліч IT-гігантів: Intel, IBM, Nokia, Toshiba та інші. В результаті об'єднання була утворена некомерційна організація Bluetooth Special Interest Group (SIG), яка в 1998 році і випустила найпершу специфікацію стандарту.

Зараз мало хто може уявити собі, що у нього в телефоні, ноутбуці або в різних переносних пристроях немає "синього зуба". Цей стандарт має свою виділену частоту, його швидкість одна з найвищих серед інших стандартів, є підтримка всіх рівнів протоколів і шифрування. Bluetooth широко поширений, оскільки вже зараз технологія вбудована практично в усі мобільні девайси, найпопулярніші - це всім відомі смартфони та планшети, ноутбуки. Даний стандарт в ранніх версіях дозволяє, як правило, пристроям спілкуватися на близькій відстані один з одним, коли вони знаходяться в радіусі приблизно 10 метрів, при цьому дальність ще залежить від інших факторів - перепон і перешкод, з новими специфікаціями радіус був значно збільшений.

З розвитком Bluetooth версії 4.0, технологія в себе включає кілька протоколів: стандартний Bluetooth високошвидкісний на основі Wi-Fi і найбільш перспективний - Bluetooth з низьким енергоспоживанням (Bluetooth Low Energy, BLE). Для інтернету речей найбільш підходящою і перспективною є технологія з низьким енергоспоживанням. BLE призначений в першу чергу для пристроїв, які мають невеликі розміри, тобто для пристроїв, в яких важлива компактність і куди не можна встановити акумулятор або батарею великого обсягу. Bluetooth LE споживає в 10-20 разів менше енергії і цілком здатний передавати дані в 50 і більше разів швидше на відстані більше 100 метрів, ніж класичні Bluetooth-рішення. Наприклад, розумний годинник, в якому встановлена мініатюрна батарейка "таблетка" або "монетка". Крім перерахованих вище переваг, BLE має високу безпеку, надійність, низьку затримку при підключенні і низьку споживану потужність. Є ще одна важлива особливість даного стандарту, вона полягає в адаптивності перебудови частоти, тобто відбувається корекція помилок при передачі сигналу, BLE швидко змінює свою робочу частоту, вибираючи найбільш оптимальну для усунення перешкод, проблем переповнення і для зниження інтерференції (накладення хвиль).

Специфікація Bluetooth 5.0 була створена, з орієнтацією на інтернет речей. Це остаточно показало, що стандарт прагне "захопити" ринок пристроїв. У порівнянні з попередньою "цілою" версією 4.0, була підвищена швидкість передачі даних майже до швидкостей HSPA + і LTE ранніх версій, при цьому енергоспоживання залишилося в колишніх показниках. Важливим показником для побудови мереж інтернету речей якраз є енергоефективність. В даний момент дана специфікація мало поширена через відносно недавню версію свого "релізу". Bluetooth 5 як і всі попередні версії має зворотну сумісність. Цілком можливо, через кілька років всі мобільні пристрої будуть підтримувати 5 версію стандарту, що є найважливішою перевагою цієї технології над іншими.

Інтернет речей стає реальністю завдяки Bluetooth, головним чином через мобільні пристрої. Як правило, стандарт застосовується в розумних

годинниках, крокомірах, термометрах, розумних чайниках, в лампочках з контролем кольору освітлення, в смартфонах і планшетах, в ноутбуках і деяких материнських платах для настільних ПК, в "маячках-біконах", медичних приладах, є також в різній побутовій техніці, є досвід успішного застосування стандарту в "розумному будинку". Маячки можуть використовуватися для надання інформації про який-небудь експонат на виставці, для реклами, для заміни GPS на внутрішню систему позиціонування – і це далеко не всі області застосування "біконів".

Bluetooth в мобільних пристроях може використовуватися для управління різними системами: освітлення, різної домашньої автоматикою, охоронною системою і багатьма іншими. Досить цікавим варіантом застосування є система з двох технологій: Bluetooth + NFC, де одна з технологій забезпечує більший радіус дії, а друга служать для встановлення швидкого і безпечного з'єднання між двома пристроями за рахунок меншого радіуса дії (наприклад, бездротові навушники з NFC -чіпом) [2].

Приклад застосування Bluetooth технології в Інтернеті речей наведено на рис.1.3.

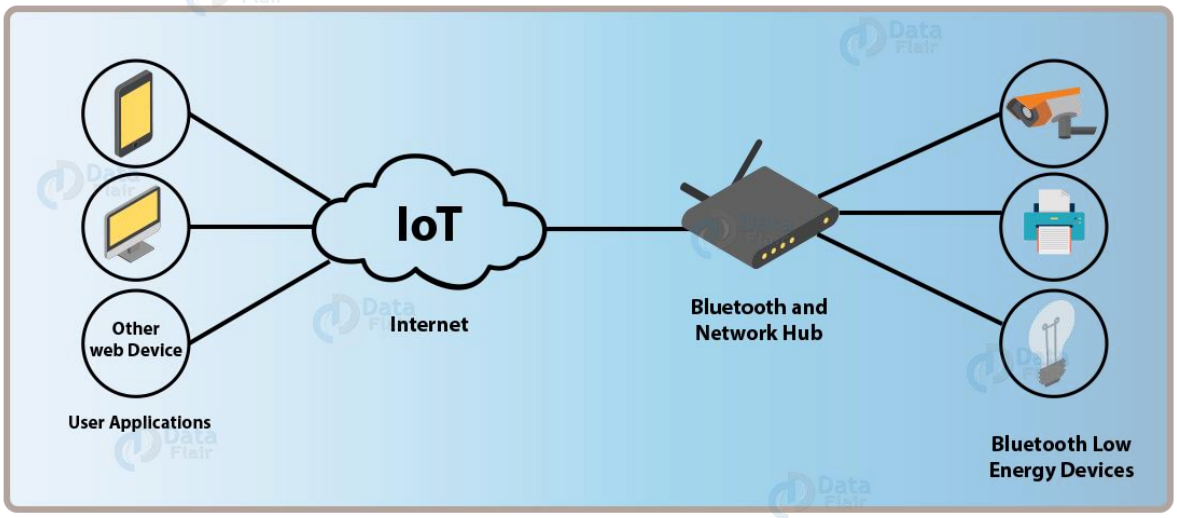

Рис.1.3 Роль Bluetooth у майбутньому IoT [5]

1.4.2 Технологія Zigbee

ZigBee схожий на Bluetooth і в основному використовується в промислових умовах. Він має ряд істотних переваг в складних системах, що

пропонують низьке енергоспоживання, високу безпеку, надійність і високу продуктивність, і має хороші можливості для використання переваг бездротового управління і сенсорних мереж в додатках IoT. Остання версія ZigBee - це недавно випущений 3.0, який являє собою об'єднання різних бездротових стандартів ZigBee в єдиний стандарт.

Zigbee застосовують для недорогих, малопотужних бездротових мереж. Технологію широко використовують для контролю і спостереження, для роботи додатків, які вимагають безпечної передачі даних.

Управлінням і запуском мережі ZigBee займається координатор. Маршрутизатор забезпечують передачу даних. Заключне ланка в ланцюжку мережі - кінцеві пристрої, які приймають і відправляють інформацію. При необхідності вони можуть переходити в сплячий режим, щоб економити заряд акумулятора [6].

1.4.3 Технологія Z-Wave

Z-Wave — це технологія IoT для радіозв'язку з низьким енергоспоживанням, яка в першу чергу призначена для домашньої автоматизації таких продуктів, як контролери ламп і датчики, серед багатьох інших пристроїв. Z-Wave використовує більш простий протокол, ніж деякі інші, що може забезпечити більш швидку і просту розробку, але єдиним виробником мікросхем є Sigma Designs в порівнянні з декількома джерелами для інших бездротових технологій, таких як ZigBee і інші.

1.4.4 Технологія Wi-Fi

Підключення до WiFi є одним з найпопулярніших протоколів зв'язку IoT, часто очевидним вибором для багатьох розробників, особливо з урахуванням доступності WiFi в домашньому середовищі всередині локальних мереж. Існує широка існуюча інфраструктура, а також пропонує швидку передачу даних і можливість обробляти великі обсяги даних. В даний час найбільш поширеним стандартом WiFi, використовуваним в домашніх умовах і на багатьох підприємствах, є 802.11n, який пропонує діапазон сотень мегабіт в секунду, що добре для передачі файлів, але може бути занадто енерговитратним для багатьох додатків IoT.

1.4.5 Стільниковий зв'язок

Будь-який додаток IoT для роботи на великих відстанях може використовувати наявні можливості стільникового зв'язку GSM / 3G / 4G. Хоча стільниковий зв'язок здатен відправляти великі обсяги даних, особливо для 4G, вартість і споживання енергії будуть занадто високими для багатьох додатків. Але він може бути ідеальним для проектів на основі датчиків з низькою пропускною здатністю, які будуть відправляти дуже мало даних через Інтернет (рис.1.4) [5].

Рис. 1.4 Стільниковий зв’язок для IoT [5]

1.4.6 Технологія NFC

NFC (Near Field Communication) — це технологія IoT (рис.1.5). Він забезпечує простий і безпечний зв'язок між електронними пристроями і, зокрема, для смартфонів, дозволяє споживачам виконувати транзакції без фізичної присутності. Це допомагає користувачеві отримати доступ до цифрового контенту і підключити електронні пристрої. По суті, це розширює можливості технології безконтактних карт і дозволяє пристроям обмінюватися інформацією на відстані менше 4 см.

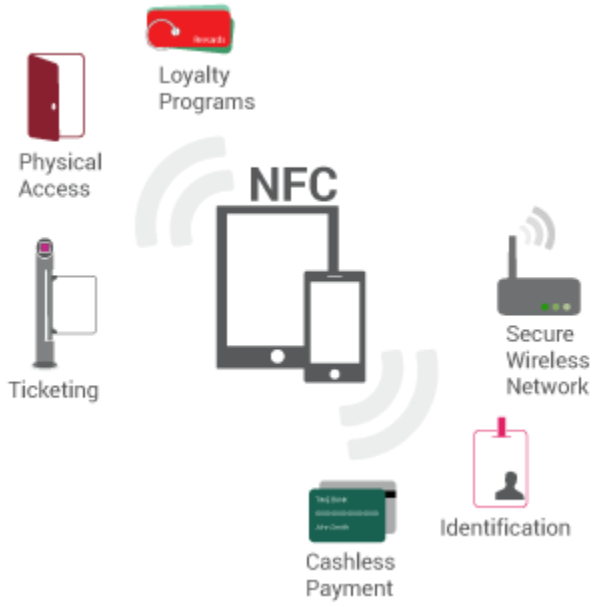

Рис. 1.5 Протокол зв'язку IoT - NFC

1.4.7 LoRaWAN

LoRaWAN — це одна з популярних технологій IoT, призначена для додатків глобальної мережі (WAN). Конструкція LoRaWAN для надання WAN з низьким енергоспоживанням функціями, спеціально необхідними для підтримки недорогий мобільний захищеного зв'язку в IoT, інтелектуальному місті і промислових додатках. Спеціально відповідає вимогам для низького енергоспоживання і підтримує великі мережі з мільйонами і мільйонами пристроїв, швидкість передачі даних варіюється від 0,3 кбіт / с до 50 Кбіт / с.

1.4.8 Протоколи обміну повідомленнями IoT

А) Протокол обмеженого застосування CoAP (Constrained Application Protocol) — це службовий інтернет-протокол для обмежених гаджетів. Він призначений для того, щоб дозволити простим пристроям з обмеженим доступом підключатися до IoT через мережі з обмеженою пропускною спроможністю. Цей протокол в основному використовується для зв'язку між комп'ютерами (M2M) і спеціально розроблений для систем IoT, заснованих на протоколах HTTP (рис.1.6).

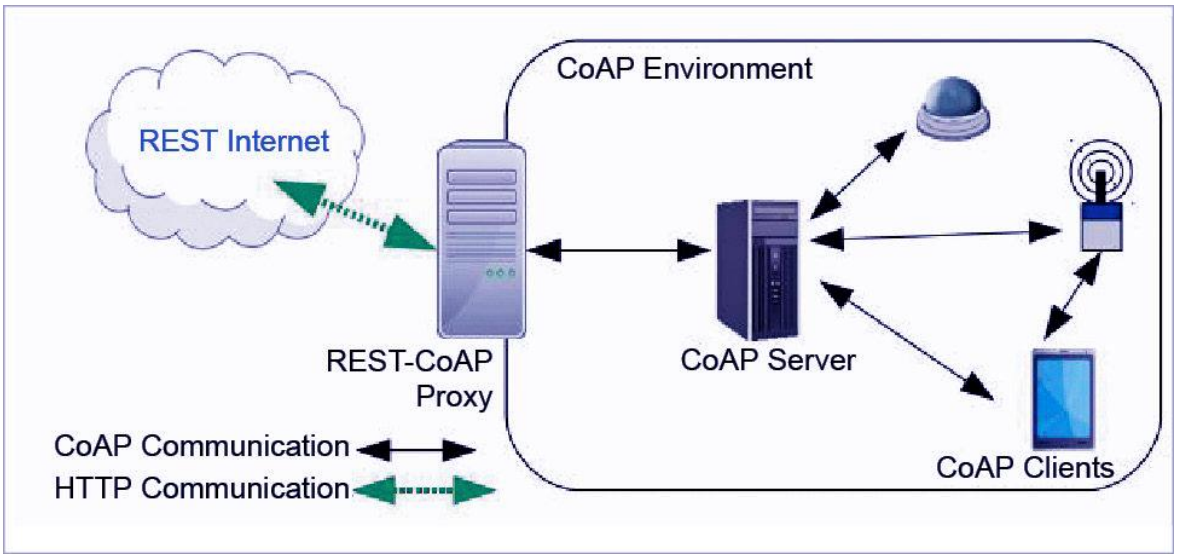

Рис. 1.6 Процес обміну повідомленнями в IoT мережі

CoAP використовує протокол UDP для полегшеної реалізації. Він також використовує архітектуру restful, яка схожа на протокол HTTP. Він використовує dtls для зручного перемикання статистики в ковзаючому шарі.

Б) MQTT (Message Queue Telemetry Transport) — транспортний протокол телеметрії в черзі повідомлень. Це протокол обміну повідомленнями, розроблений за допомогою Енді Стенфорд-Кларка з IBM і Арлена Ніппер з Arcom в 1999 році і призначений для зв'язку в системах M2M (міжмашинної взаємодії). Зазвичай він використовується для віддаленого відстеження в IoT. Його основне завдання — зібрати статистику по багатьом гаджетам і надати інфраструктуру. MQTT пов'язує гаджети і мережі з пакетами і проміжним програмним забезпеченням. Всі пристрої підключаються до серверів концентратора фактів, таким як новий пристрій IBM для перегляду повідомлень. Протокол MQTT працює на прикладному рівні поверх TCP /IP для надання простих і надійних потоків інформації.

Обмін повідомленнями в протоколі MQTT здійснюється між клієнтом (client), який може бути видавцем або підписником (publisher / subscriber) повідомлень, і брокером (broker) повідомлень (рис.1.7).

Видавець відправляє дані на MQTT брокер, вказуючи в повідомленні певну тему, топік (topic). Підписники можуть отримувати різні дані від багатьох видавців залежно від підписки на відповідні топіки.

Пристрої MQTT використовують певні типи повідомлень для взаємодії з брокером, нижче представлені основні:

- Connect - встановити з'єднання з брокером
- Disconnect - розірвати з'єднання з брокером
- Publish - опублікувати дані в топік на брокера
- Subscribe - підписатися на топік на брокера
- Unsubscribe - відписатися від топіка

Рис. 1.7 Схема простої взаємодії між підписником, видавцем і брокером

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

Архітектуру протоколу розробив John O'Hara з банку JP Morgan Chase & Co. «Повідомлення» (message) відправляються в «точку обміну». Точка обміну розподіляє повідомлення в одну або кілька «черг». При цьому в точці обміну повідомлення не зберігаються. Точки обміну бувають трьох типів:

- розгалуження (англ. fanout) — повідомлення передається в усі причеплені до неї черги ;
- прямі (direct) — повідомлення передається в чергу з ім'ям, що збігається з ключем маршрутизації (routing key), який вказується при відправці повідомлення ;
- теми (topic) — щось середнє між розгалуженням і прямим типом, повідомлення передається в черзі, для якої збігається маска на ключ маршрутизації.

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

Г) DDS (Data Distribution Service) — служба поширення даних для систем реального часу є стандартом міжмашинної взаємодії Object Managment Group, метою якої є забезпечення масштабованих, оперативних, надійних, високопродуктивних і сумісних обмінів даними з використанням шаблону Видавець-підписник. DDS задовольняє потреби додатків, пов'язаних з управлінням повітряним рухом, розумних мереж енергопостачання, автономних засобів пересування, робототехніки, логістики, енергопостачання, медичного обладнання, симуляції і тестування, космонавтики і оборони, а також інших програм, що вимагають обміну даних в реальному часі [5].

Д) HTTP (Hyper Text Transfer Protocol) — це протокол прикладного рівня, що дозволяє отримувати різні ресурси, наприклад HTML-документи. Протокол HTTP лежить в основі обміну даними в Інтернеті. HTTP є протоколом клієнт-серверногї взаємодії, що означає ініціювання запитів до сервера самим одержувачем, зазвичай веб-браузером (web-browser). Отриманий підсумковий документ буде складатися з різних піддокументів, які є частиною підсумкового документа: наприклад, з окремо отриманого тексту, опису структури документа, зображень, відео-файлів, скриптів і багато чого іншого.

Клієнти і сервери взаємодіють, обмінюючись поодинокими повідомленнями (а не потоком даних). Повідомлення, відправлені клієнтом, зазвичай веб-бруезером, називаються запитами, а повідомлення, відправлені сервером, називаються відповідями.

1.4.9 Ідентифікаційні і відстежуючі технології

Технології ідентифікації і відстеження, що використовуються в IoT, включають системи RFID, штрих-код і інтелектуальні датчики. Проста система RFID складається зі зчитувача RFID і мітки RFID. Завдяки своїй здатності ідентифікувати, записувати і відстежувати пристрої і фізичні

об'єкти, система RFID все частіше використовується в таких галузях, як логістика, управління ланцюгами постачання й моніторинг медичних послуг. Інші переваги системи RFID включають надання точної інформації про підключені пристрояї в режимі реального часу, зниження трудовитрат, спрощення бізнес-процесів, підвищення точності інвентарної інформації та підвищення ефективності бізнесу. До теперішнього часу система RFID успішно використовувалася багатьма виробниками, дистриб'юторами та роздрібними торговцями в багатьох галузях промисловості.

Нещодавній розвиток технології RFID фокусується на таких аспектах:

1) активні системи RFID з передачею з розширеним спектром;

2) технологія управління RFID-додатками.

Існує ще багато можливостей для зростання додатків на основі RFID. Для подальшого просування технології RFID, RFID може бути інтегрований з WSN для кращого відстеження та відстеження подій в режимі реального часу. Зокрема, з'являються технології бездротових інтелектуальних датчиків, такі як електромагнітні датчики, біосенсори, зовнішні датчики, мітки датчиків, незалежні мітки і сенсорні пристрої, які ще більше полегшують впровадження і розгортання промислових служб і додатків. Завдяки інтеграції даних, отриманих інтелектуальними датчиками, з даними RFID, можна створювати більш потужні програми IoT, які підходять для промислових середовищ.

1.5 Інформаційна безпека та захист конфіденційності

Прийняття і широке поширення нових технологій і послуг IoT багато в чому буде залежати від інформаційної безпеки і захисту конфіденційності даних, які є двома складними проблемами в IoT через його розгортання, мобільності і складності. Багато існуючих технологій доступні для використання споживачами, але не підходять для промислових застосувань, які мають строгі вимоги безпеки. Для захисту інформації необхідно ретельно вивчити існуючу технологію шифрування, запозичену у WSN або інших мереж, коли вони використовуються для створення IoT. Оскільки Інтернет

речей дозволяє щодня відстежувати, відстежувати і підключати багато речей, а також можна збирати багато особистої та приватної інформації автоматично. Захист конфіденційності в середовищі IoT стає більш серйозною, ніж в традиційному середовищі ІКТ, оскільки число векторів атак на об'єкти IoT, мабуть, значно більше. Наприклад, медичний монітор буде збирати інформацію про пацієнта, таку як частота серцевих скорочень і рівень цукру в крові, а потім відправляти інформацію безпосередньо в кабінет лікаря по мережі. Коли інформація передається по мережі, дані пацієнта можуть бути вкрадені або зламані. Іншим прикладом є те, що біосенсор, який використовується в харчовій промисловості, може використовуватися для контролю температури і бактеріального складу їжі, що зберігається в холодильнику. Коли деякі продукти харчування псуються, дані можуть бути відправлені в продовольчу компанію через мережу. Однак такі дані повинні зберігатися строго конфіденційно, щоб захистити репутацію харчової компанії. Слід зазначити, що деякі питання, такі як визначення недоторканності приватного життя і юридичне тлумачення, як і раніше розпливчасті і не є чітко визначені в IoT. Хоча існуючі технології мережевої безпеки забезпечують основу для конфіденційності та безпеки в IoT, ще належить виконати велику роботу. Надійний механізм захисту безпеки для IoT повинен бути досліджений в таких аспектах:

1) визначення безпеки і конфіденційності з точки зору соціальної, правової та культурної точок зору;

2) механізм довіри і репутації;

3) безпеку зв'язку, така як наскрізне шифрування;

4) конфіденційність спілкування і призначених для користувача даних;

5) безпеку послуг і додатків [3].

Висновки:

У якості складної кіберфізичної системи IoT включає в себе різні пристрої, оснащені можливостями виявлення, ідентифікації, обробки, зв'язку та роботи в мережі. Зокрема, датчики і виконавчі механізми стають все більш потужними, менш дорогими і меншими, що робить їх використання

повсюдним. Галузі зацікавлені в розгортанні пристроїв IoT для розробки промислових додатків, таких як автоматизований моніторинг, контроль, управління і обслуговування. У зв'язку з швидким розвитком технологій і промислової інфраструктури, IoT, як очікується, буде широко застосовуватися в промисловості. Наприклад, харчова промисловість інтегрує WSN і RFID для створення автоматизованих систем відстеження, моніторингу та відстеження якості продуктів харчування по всьому ланцюжку поставок продуктів харчування з метою поліпшення якості продуктів харчування.

2. ДОСЛІДЖЕННЯ АРХІТЕКТУР ТА АЛГОРИТМІВ РОБОТИ IoT

Інтернет речей (IoT) надав багатообіцяючу можливість побудувати потужні промислові системи та додатки, використовуючи зростаючу популярність радіочастотної ідентифікації (RFID) та бездротових, мобільних та сенсорних пристроїв. В останні роки було розроблено та впроваджено широкий спектр промислових застосувань IoT.

2.1 Стандартизація IoT

IoT можна розглядати як глобальну мережеву інфраструктуру, що складається з безлічі підключених пристроїв, що використовують сенсорні, комунікаційні, мережеві технології та технології обробки інформації.

На сьогоднішній день IoT завойовує популярність в таких галузях, як логістика, виробництво, роздрібна торгівля і фармацевтика. З розвитком технологій бездротового зв'язку, смартфонів і сенсорних мереж в IoT включається все більше мережних речей або інтелектуальних об'єктів. В результаті ці технології, пов'язані з IoT, також мали великий вплив на нові інформаційні та комунікаційні технології (ІКТ) і технології корпоративних систем (рис. 2.1).

Швидке зростання IoT ускладнює стандартизацію. Однак стандартизація відіграє важливу роль для подальшого розвитку і поширення IoT. Стандартизація в IoT спрямована на зниження вхідних бар'єрів для нових постачальників послуг і користувачів, на поліпшення взаємодії різних додатків/ систем і на забезпечення більш ефективної роботи продуктів або послуг на більш високому рівні. Необхідний ретельний процес стандартизації і багато зусиль з координації, щоб пристрої і додатки з різних країн могли обмінюватися інформацією. Різні стандарти, використовувані в IoT (наприклад, стандарти безпеки, стандарти зв'язку і стандарти ідентифікації), можуть бути ключовими факторами, що сприяють поширенню технологій IoT, і повинні розроблятися з урахуванням нових технологій. Конкретні проблеми в стандартизації IoT включають проблему сумісності, проблеми рівня радіодоступу, семантичну сумісність, а також проблеми безпеки і

конфіденційності. Крім того, для спрощення інтеграції різних послуг також рекомендуються галузеві керівні принципи або стандарти для впровадження IoT в промислових середовищах.

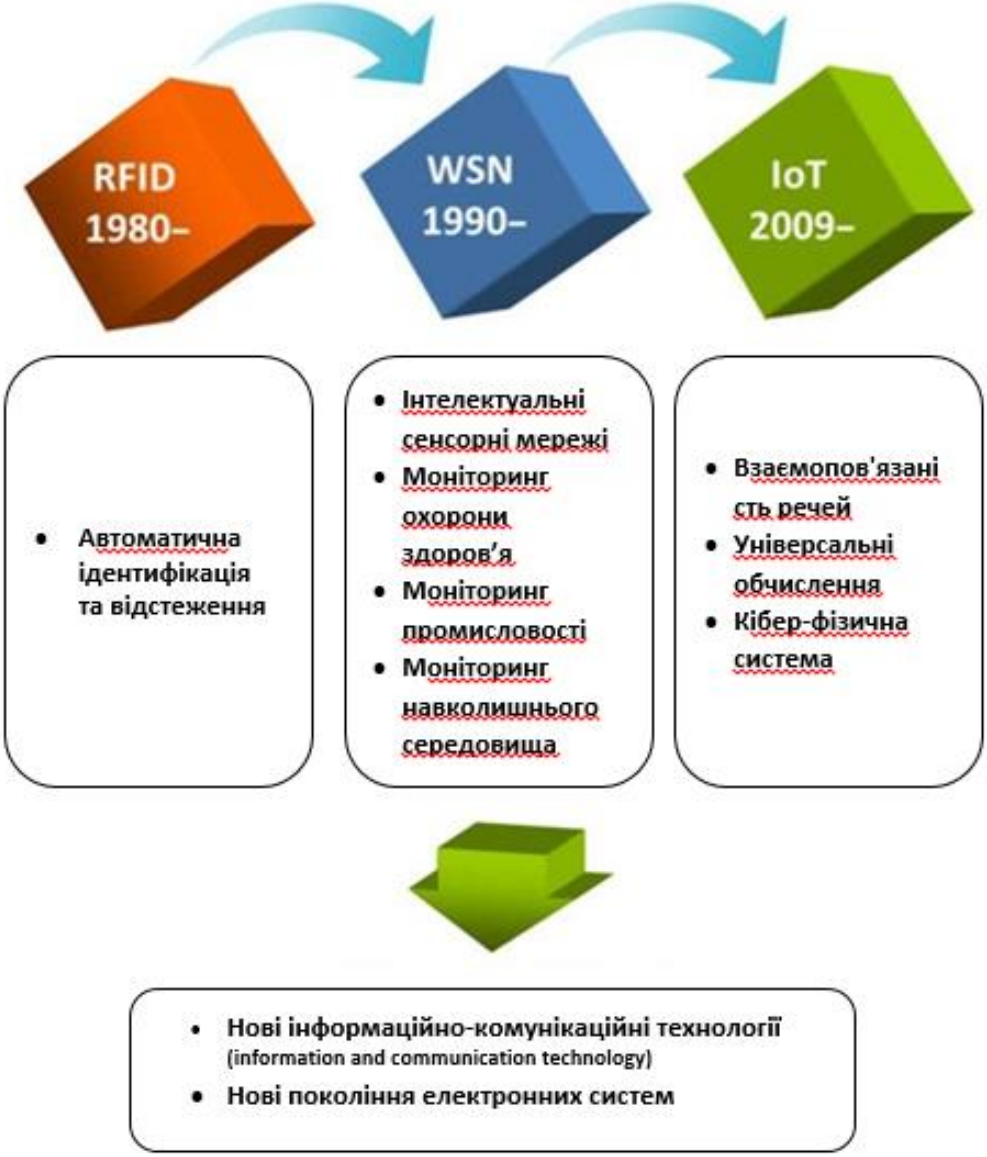

Рис. 2.1 IoT-технології та їх вплив на нові ІКТ та корпоративні системи.

Щоб надавати високоякісні послуги кінцевим користувачам, необхідно розробити технічні стандарти IoT, щоб визначити специфікацію для обміну інформацією, її обробки та обміну даними. Успіх IoT залежить від стандартизації, яка забезпечує функціональну сумісність, сумісність, надійність і ефективні операції в глобальному масштабі. Багато країн і організації зацікавлені в розробці стандартів IoT, тому що це може принести величезні економічні вигоди в майбутньому. В даний час над створенням різних стандартів IoT працюють численні організації, такі як Міжнародний

союз електрозв'язку, Міжнародна електротехнічна комісія, Міжнародна організація по стандартизації, IEEE, Європейський комітет з електротехнічної стандартизації, Китайський інститут стандартизації електроніки і Американський національний інститут стандартів. Оскільки в розробці стандартів IoT бере участь дуже багато організацій, необхідна тісна координація між різними організаціями по стандартизації для координації і регулювання відносин між міжнародними організаціями зі стандартизації та національними / регіональними організаціями зі стандартизації. Встановлюючи загальноприйняті стандарти, розробники і користувачі можуть впроваджувати програми та послуги IoT, які будуть розгорнуті і використані в великих масштабах, при цьому зберігаючи витрати на розробку і обслуговування в довгостроковій перспективі. Стандартизація технологій в IoT також прискорить широке поширення технологій і інновацій IoT [3].

2.2 Детальний огляд сервіс-орієнтованої архітектури IoT

IoT прагне поєднувати різні речі в мережі. В якості ключової технології інтеграції різнорідних систем або пристроїв сервіс-орієнтованої архітектури (SOA—Service Oriented Architecture) може застосовуватися для підтримки IoT. SOA успішно використовується в таких областях досліджень, як хмарні обчислення, WSN і автомобільна мережа. Було запропоновано досить багато ідей для створення багаторівневої архітектури SOA для IoT на основі обраної технології, бізнес-потреб і технічних вимог. Наприклад, Міжнародний союз електрозв'язку рекомендує, щоб архітектура IoT складалася з п'яти різних рівнів: визначення, доступ, мережева взаємодія, проміжне програмне забезпечення, і прикладні рівні. З точки зору функціональних можливостей в таблиці 1 показаний чотирьохшаровий SOA IoT. У таблиці II наведені конструктивні міркування для промислових додатків IoT. На рис. 2.2 показана SOA, де чотири шари взаємодіють один з одним.

Архітектурний дизайн IoT пов'язаний зі стилями архітектури, мережами та комунікаціями, інтелектуальними об'єктами, веб-сервісами та додатками, бізнес-моделями і відповідними процесами, спільної обробкою

даних, безпекою і т. д. З технологічної точки зору дизайн архітектури IoT повинен розглянути розширюваність, масштабованість , модульність і сумісність між гетерогенними пристроями. Оскільки речі можуть переміщатися або вимагати взаємодії в реальному часі зі своїм середовищем, необхідна адаптивна архітектура, щоб допомогти пристроїв динамічно взаємодіяти з іншими речами. Децентралізована та гетерогенна природа IoT вимагає, щоб архітектура забезпечувала IoT ефективну керовану подіями можливість. Таким чином, SOA вважається хорошим підходом для забезпечення взаємодії між гетерогенними пристроями безліччю способів.

Таблиця 2.1 Чотирьох-рівнева архітектура для IoT (Internet of Things)

Рівень	Опис
Рівень зчитування	Інтегрований з існуючим обладнанням (RFID, датчики, виконавчі пристрої тощо) для визначення / контролю фізичного світу та отримання даних
Мережевий рівень	Забезпечує основну мережеву підтримку та передачу даних по бездротовій або дротовій мережі
Службовий рівень	Створює та керує службами. Він надає послуги для задоволення потреб користувачів
Інтерфейсний рівень	Забезпечує методи взаємодії з користувачами та іншими програмами

Таблиця 2.2 Висновки проектування для промислової IoT заявки

Дизайнерські цілі	Опис
Енергія	Як довго може пристрій IoT працювати з обмеженим джерелом живлення?
Затримка	Скільки часу потрібно для розповсюдження та обробки повідомлень?
Пропускна здатність	Яка максимальна кількість даних, які можна перенести через мережу?
Масштабність	Скільки пристроїв підтримується?
Топологія	Хто з ким повинен спілкуватися?
Безпека	Наскільки безпечним є додаток?

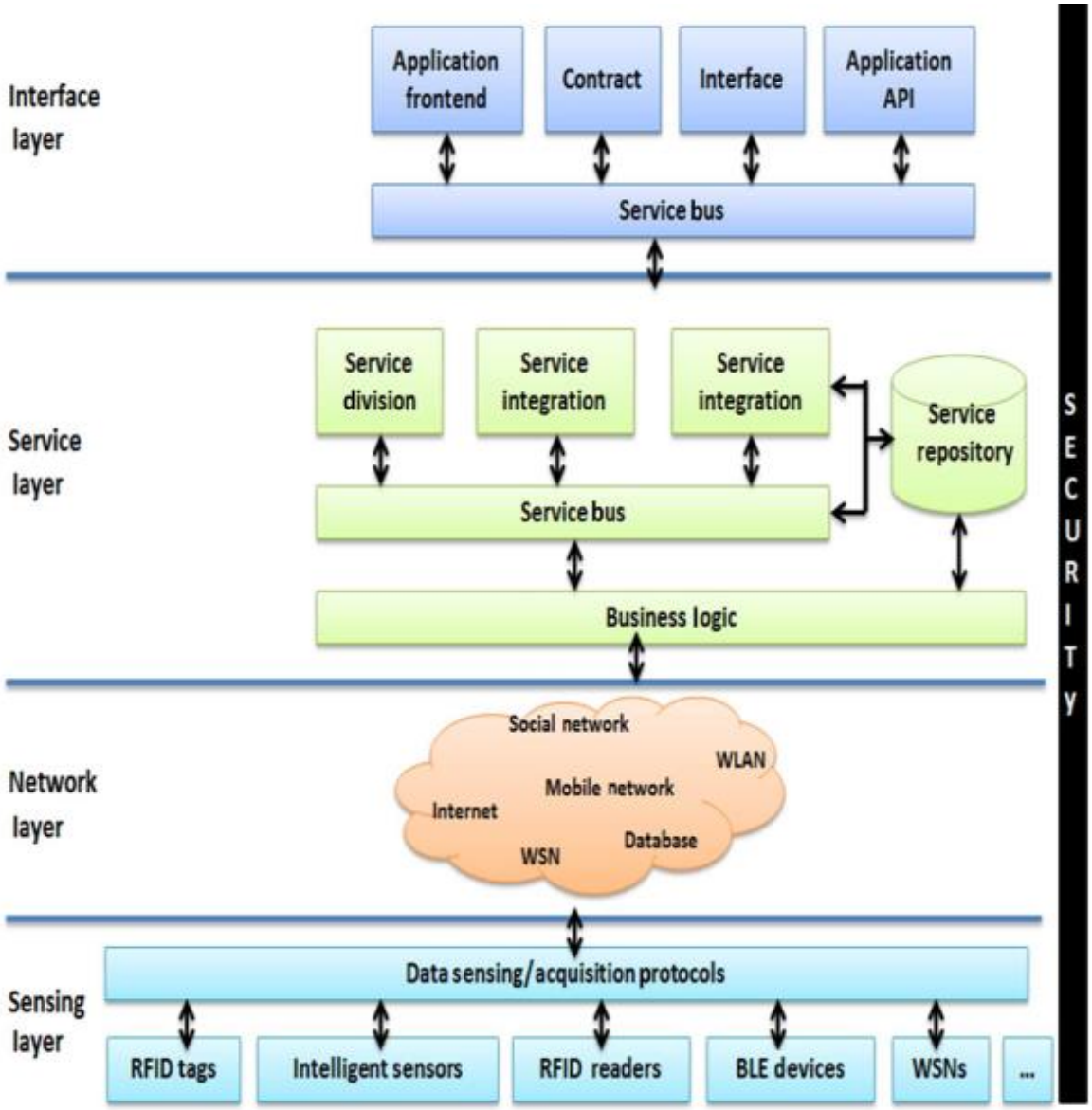

Рис. 2.2 Сервісно-орієнтована архітектура (SOA) для IoT

А) Сенсорний рівень

IoT можна розглядати як всесвітню фізичну внутрішню мережу, в якій все можна підключати і контролювати віддалено. Оскільки все більше і більше пристроїв оснащуються RFID або інтелектуальними датчиками, підключення стає набагато простіше. На рівні виявлення бездротові інтелектуальні системи з мітками або датчиками тепер можуть автоматично визначати і обмінюватися інформацією між різними пристроями. Ці технологічні досягнення значно покращують здатність IoT визначати і ідентифікувати речі або середу. У деяких галузях промисловості схеми розгортання інтелектуальних послуг і універсальний унікальний

ідентифікатор (UUID) призначаються кожній послузі або пристрою, які можуть знадобитися. Пристрій з UUID може бути легко ідентифіковано і вилучено. Таким чином, UUID мають вирішальне значення для успішного розгортання послуг в такій величезній мережі, як IoT.

Б) Мережевий рівень

Роль мережевого рівня полягає в тому, щоб з'єднати все разом і дозволити речам ділитися інформацією з іншими, пов'язаними речами. Крім того, мережевий рівень здатний агрегувати інформацію з існуючих IT-інфраструктур (наприклад, бізнес-систем, транспортних систем, електромереж, систем охорони здоров'я, систем ІКТ та т. Д.). У SOA-IoT сервіси, що надаються речами, зазвичай розгортаються в гетерогенної мережі, а всі пов'язані речі переносяться в сервіс Internet. Цей процес може включати управління і контроль QoS (quality of service — якість обслуговування) відповідно до вимог користувачів / додатків. З іншого боку, для динамічно змінюється мережі важливо автоматично виявляти і відображати об'єкти в мережі. Речі повинні бути автоматично призначені з ролями для розгортання, управління і планування поведінки речей і можливості перемикання на будь-які інші ролі в будь-який час у міру необхідності. Ці можливості дозволяють пристроям спільно виконувати завдання. Щоб спроектувати мережевий рівень в IoT, розробникам необхідно вирішити такі проблеми, як технології управління мережею для різнорідних мереж (таких як фіксовані, бездротові, мобільні і т. д.), Енергоефективність в мережах, вимоги до QoS, виявлення і пошук послуг, дані і сигнал обробка, безпеку і конфіденційність.

В) Сервісний рівень

Рівень обслуговування заснований на технології проміжного програмного забезпечення, яка забезпечує функціональні можливості для безперешкодної інтеграції служб і додатків в IoT. Технологія проміжного програмного забезпечення забезпечує IoT економічно ефективною платформою, в якій апаратні і програмні платформи можуть використовуватися повторно. Основним видом діяльності на рівні сервісів є

специфікації сервісів для проміжного програмного забезпечення, які розробляються різними організаціями. Добре спроектований рівень обслуговування зможе визначати загальні вимоги до додатків і надавати API і протоколи для підтримки необхідних послуг, додатків та потреб користувачів. Цей рівень також обробляє всі сервіс-орієнтовані питання, включаючи обмін і зберігання інформації, управління даними, пошукові системи і комунікації. Цей шар включає в себе наступні компоненти:

1) Виявлення послуг: пошук об'єктів, які можуть запропонувати необхідні послуги і інформацію ефективним способом.

2) Склад послуг: забезпечення взаємодії і зв'язку між пов'язаними речами. На етапі виявлення використовуються взаємозв'язку між різними речами для виявлення бажаної послуги, а компонентом композиції послуг є планування або повторне створення найбільш сприятливих послуг для придбання найбільш надійних послуг для задоволення запиту.

3) Управління надійністю: метою є визначення механізмів довіри і репутації, які можуть оцінювати і використовувати інформацію, надану іншими службами, для створення надійної системи.

4) Сервісні API: підтримка взаємодії між сервісами, необхідними в IoT.

Г) Інтерфейсний рівень

У IoT велика кількість задіяних пристроїв виготовляється різними виробниками / постачальниками, і вони не завжди дотримуються одних і тих же стандартів / протоколів. В результаті неоднорідності виникає багато проблем взаємодії з обміном інформацією, зв'язком між речами і спільною обробкою подій між різними речами. Крім того, постійне збільшення кількості речей, що беруть участь в IoT, ускладнює динамічне підключення, зв'язок, відключення і роботу. Існує також необхідність в рівні інтерфейсу, щоб спростити управління і взаємозв'язок речей. Профіль інтерфейсу (IFP) можна розглядати як підмножину стандартів обслуговування, які підтримують взаємодію з додатками, розгорнутими в мережі. Хороший інтерфейсний профіль пов'язаний з реалізацією універсальної технології Plug and Play (UPnP), яка визначає протокол для полегшення взаємодії зі

службами, які надаються різними речами. Профілі інтерфейсу використовуються для опису специфікацій між додатками і сервісами. Сервіси на рівні сервісів працюють безпосередньо в обмежених мережевих інфраструктурах, щоб ефективно знаходити нові сервіси для додатка в міру їх підключення до мережі. Нещодавно була запропонована архітектура інтеграції SOCRADES (SIA) для ефективної взаємодії між додатками і сервісами. Традиційно сервісний рівень надає універсальний API для додатків. Однак недавні результати досліджень SOA-IoT показали, що процес надання послуг (SPP) також може ефективно забезпечувати взаємодію між додатками і сервісами. Спочатку SPP виконує «запит типів», який відправляє запит на сервіси з загальним форматом WSDL, а потім використовує механізм «пошуку кандидатів» для пошуку потенційних сервісів. На основі «контексту програми» і «інформації QoS» всі екземпляри сервісів ранжуються, і для ідентифікації примірника сервісу, який відповідає вимогам програми, буде використовуватися механізм «надання послуг на вимогу». Зрештою, «Оцінка процесу» використовується для оцінки процесу [7].

2.3 Узагальнена архітектура IoT

Бачення Інтернету речей (IoT) описує майбутнє, де багато повсякденних об'єктів взаємопов'язані через глобальну мережу. Вони збирають та обмінюються даними про себе та оточення, щоб дозволити широкий моніторинг, аналіз, оптимізацію та контроль. Ще недавно це було лише баченням, але останнім часом років це повільно перетворилося на реальність. Постійне зменшення цін, габаритів та енергетичних потреб електроніки тепер дозволяє крихітним пристроям ненав’язливо вимірювати своє оточення. Багато пристроїв використовують низько енергетичну технологію зв'язку для надсилання цих вимірювань до інших, більш потужних компонентів, таких як Bluetooth шлюзи, мобільні телефони або гарячі точки WiFi. Пристрої все більше включають бездротові бездротові технології, такі як LoRa або існуючі мережі 2G та 3G. Місцеві крайові процесори, хаби або Інтернет-сервіси в свою чергу аналізують і обробляють

дані датчиків IoT, щоб створити нові знання, які можна використати для впливу на навколишнє середовище за допомогою пускачів. Коротше кажучи, IoT можна розглядати як гігантський кіберфізичний цикл управління. У цьому контексті для опису таких параметрів часто використовується термін «зв'язок між машиною» (M2M).

Компаніями та науково-дослідними установами протягом багатьох років були створені різні втілення систем IoT для різних випадків використання. Smart Homes — один із прикладів таких систем IoT. В інших областях ведуться подібні розробки, такі як підключені машини, розумні міста, управління попитом, смарт-сітки або системи Smart Factory.

Хоча можлива локальна обробка даних, що генеруються цими системами, і розумний підхід для випадків використання, коли потрібна низька затримка, хмарні платформи використовуються для обробки та аналізу великих наборів даних. В результаті за останні кілька років було створено понад сто таких платформ. Деякі приклади включають AWS IoT, FIWARE, OpenMTC та SmartThings.

Ці платформи бувають різних форм і розмірів. Поки тривають зусилля зі стандартизації, наразі не існує загально узгоджених стандартів для IoT. Швидше за все, розвиток цих платформ часто відбувався в бункерах. Ці різні середовища вплинули не лише на вибір понять та технології, але й на вибір термінології. В результаті ландшафт платформи став дуже неоднорідним. Однак у той же час усі ці рішення роблять приблизно однакові речі: вони дозволяють підключати різні пристрої, отримувати доступ та обробляти їх дані та використовувати знання, отримані за допомогою цієї діяльності, для створення автоматизованого управління.

Неоднорідність цих підходів створює проблему для того, хто повинен вибрати одне з цих рішень. Пошук потрібної платформи для випадку використання стає дуже трудомістким, коли кожне рішення використовує різні технології та термінологію. Необхідно прочитати та зрозуміти описи та документацію кожної платформи, щоб прийняти рішення. Для цього

потрібен не лише час, а й технічні знання, щоб можна було зрозуміти та порівняти різні поняття.

Довідкова архітектура, яка відображає існуючі описи архітектури та пропонує уніфіковану термінологію, допомагає в цьому процесі вибору. Він не тільки дозволяє простіше порівняти між собою платформами, але й надає корисну основу як вихідну точку для нових розробок. Тому далі визначимо довідкову архітектуру IoT на основі існуючих платформ. Вона зберігається цілеспрямовано абстрактною, щоб зробити її застосовною в широкому діапазоні ситуацій. Далі буде розглянуто уточнення опису, розширення аналізу восьми платформ IoT.

Нижче наведено довідкову архітектуру IoT (рис. 2.3), яка пропонує уніфіковану термінологію та відображення існуючих описів архітектури. Почнемо з визначення всіх компонентів, показаних на рис. 2.3, починаючи знизу. Для чіткого розмежування понять акцентуємо увагу на елементах опорної архітектури IoT, представлених у цій роботі курсивом.

Датчик (Sensor) — це апаратний компонент, який фіксує інформацію про фізичне середовище, "реагуючи (фізично) на фізичний стимул (як тепло, світло, звук, тиск, магнетизм чи певний рух)". Наприклад, вимірюючи вологість у приміщенні, *датчик*, розташований всередині цієї кімнати, фіксує рівень вологості приміщення. *Датчики* передають захоплену інформацію, використовуючи електричні сигнали, до *пристроїв*, до яких вони підключені. Це з'єднання може бути встановлено дротовим або бездротовим шляхом. Провідне з'єднання включає інтеграцію *датчиків* у *пристрій*. *Датчик* може бути налаштований за допомогою програмного забезпечення, але не може запускати програмне забезпечення самостійно.

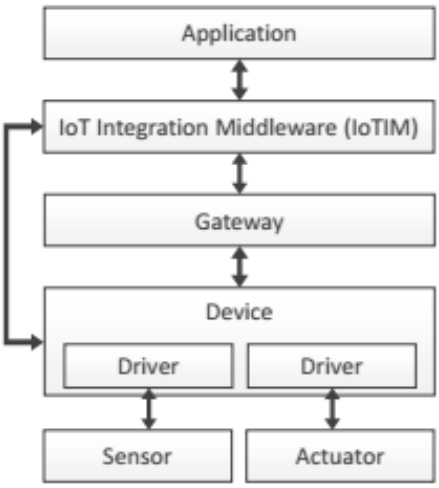

Рис. 2.3 Довідкова архітектура IoT

П*ускач (Actuator)* — це апаратний компонент, який маніпулює фізичним середовищем. *Пускачі* отримують команди від свого підключеного *пристрою* і переводять ці електричні сигнали в якусь фізичну дію. Наприклад, *пускач*, що включає або вимикає вентиляцію в приміщенні, впливає на фізичне середовище, впливаючи на вологість приміщення. Подібно до *датчиків*, з'єднання з пристроєм можна встановити дротовим або бездротовим шляхом, при цьому дротовий зв’язок включає інтеграцію в пристрій. Крім того, пускач може бути налаштований за допомогою програмного забезпечення, але не може запускати програмне забезпечення самостійно.

Пристрій (Device) — це апаратний компонент, який підключений до датчиків та / або виконавчих механізмів за допомогою дроту або бездротового зв’язку або інтегрує ці компоненти. На пристроях є процесор і ємність для зберігання програмного забезпечення та встановлення з'єднання *з Програмним забезпеченням для інтеграції IoT (IoTIM)*. Наприклад, зовнішній модуль метеостанції Netatmo являє собою пристрій із вбудованими датчиками. Таким чином, пристрої є точкою входу фізичного середовища в цифровий світ. *Драйвер (Driver)* — це програмне забезпечення, яке працює на пристрої, що забезпечує рівномірний доступ до неоднорідних датчиків та пускачів. Пристрої або є автономними, або підключені до іншої, більшої

системи. Програмне забезпечення для інтеграції IoT представляє таку систему.

Якщо пристрій не здатний безпосередньо підключитися до подальших систем, він підключається до *шлюзу (Gateway)*. *Шлюз* забезпечує необхідні технології та механізми для перекладу між різними протоколами, технологіями зв'язку та форматами корисного навантаження. Він передає зв'язок між пристроями та іншими системами. Наприклад, критий модуль метеостанції Netatmo являє собою пристрій із вбудованими датчиками, що діє як шлюз для зовнішнього модуля метеостанції. Коли шлюз отримує повідомлення у фірмовому бінарному форматі від пристрою, він переводить двійковий формат у більш поширений формат, наприклад, JSON, і передає дані в призначену систему, наприклад, через IP. При необхідності шлюз може також перекладати команди, що надсилаються із систем на пристрої, в комунікаційні технології, протоколи та формати, підтримувані відповідним пристроєм.

Програмне забезпечення для інтеграції IoT (IoTIM) служить інтеграційним шаром для різних типів *датчиків*, *пускачів*, *пристроїв* та *додатків*. Він несе відповідальність за отримання даних із підключених *пристроїв*, обробку отриманих даних, надання отриманих даних підключеним *додаткам* та контроль *пристроїв*. Прикладом для обробки є оцінка правил виконання дії та відправлення команд виконавцям на основі цієї оцінки. Пристрій може безпосередньо спілкуватися з програмним забезпеченням для інтеграції *IoTIM*, якщо він підтримує відповідну технологію зв'язку, наприклад IP через Ethernet або WiFi, відповідний транспортний протокол, наприклад HTTP або MQTT і сумісний формат корисного навантаження, наприклад, JSON. В іншому випадку пристрій зв’язується через шлюз із *IoTIM*. Проміжне програмне забезпечення для інтеграції IoT не обмежується функціями, описаними вище. Воно може містити всі види функціональності, необхідні певній кіберфізичній системі, наприклад, базу даних часових рядів або графічні інформаційні панелі. Додатково може здійснюватися управління пристроями та користувачами, а

також агрегація та використання отриманих даних. Наприклад, платформа SmartThings може використовуватися з пристроями Netatmo. Як правило, до проміжного програмного забезпечення *IoTIM* можна отримати доступ за допомогою API, наприклад, API REST на основі HTTP.

Компонент *додаток (Application)* являє собою програмне забезпечення, яке використовує *IoTIM* для розуміння фізичного середовища та / або для маніпулювання фізичним світом. Це робиться, запитуючи дані датчика або контролюючи фізичні дії за допомогою пускачів. Наприклад, програмна система, яка контролює температуру будівлі, представляє програму, підключену до програмного забезпечення IoT Integration Middleware. Додаток також може бути іншим програмним забезпеченням для інтеграції IoT.

Таким чином, ми розглянули довідкову архітектуру, що складається з шести типів компонентів. При реалізації архітектури компоненти можна опустити. Це може бути так, якщо платформа використовується лише для вимірювання змін фізичного світу. Наприклад, платформа, що збирає рівень CO2 у повітрі, може не мати підключених пускачів, якщо система використовується лише для вимірювання та збору даних. Іншим прикладом для пропущених компонентів є платформи із підключеними пристроями, здатними до необхідних технологій спілкування безпосередньо з програмним забезпеченням інтеграції IoT, тому шлюз не потрібен для відповідного обміну повідомленнями [8].

2.4 Порівняння платформ IoT

У цьому пункті довідкова архітектура IoT відображається на чотирьох платформах з відкритим кодом та чотирьох власних IoT-рішень. Вибраними платформами з відкритим кодом є FIWARE, OpenMTC, SiteWhere та Webinos. Патентовані рішення — це AWS IoT2, платформа Watson IoT від IBM10, Azure IoT Hub від Microsoft та SmartThings від Samsung. Під час порівняння функціональність компонента та їх іменування є ключовими сферами, які порівнюються з референтною архітектурою. Порівняння кожної

платформи буде детально описано. Додатково подано складене порівняння, де обговорюються основні аспекти.

2.4.1 Архітектура FIWARE

Архітектура та відображення платформи з відкритим кодом FIWARE на нашу довідкову архітектуру IoT показано на рис. 2.4. FIWARE фінансується Європейським Союзом та Європейською Комісією. Він забезпечує розширену хмару на основі OpenStack, де розміщені його можливості та Каталог. Каталог FIWARE містить загальні можливості (GE—Generic Enablers), що представляють собою багату бібліотеку компонентів. У межах архітектури, показаної на рис. 2.4, показані лише загальні можливості частини IoT. FIWARE розрізняє лише пристрої та пристрої NGSI. Оскільки в документації FIWARE описано, що пристрої можуть мати вбудовані датчики та пускачі, всі компоненти пристрою складаються з компонентів Пристрій, Датчик та Пускач. Пристрої можуть зв’язуватися напряму з IoT Back-End або через шлюз, який розміщений в межах IoT Edge. Як шлюз IoT, так і шлюз IoT NGSI забезпечують і керують зв’язком пристроїв із IoT Back-End. Отже, IoT Edge являє собою шлюз нашої довідкової архітектури IoT. Внутрішня частина IoT і брокер контексту даних надають основні функціональні можливості FIWARE, тому вони інкапсульовані проміжним програмним забезпеченням для інтеграції IoT. Документація FIWARE описує, як інші додатки можуть бути підключені через посередник Data Context Broker до платформи. Хоча компонент програми не представлений на діаграмі архітектури FIWARE, на підставі цього опису ми таким чином розміщуємо компонент Додаток (Application) поверх брокера контексту даних.

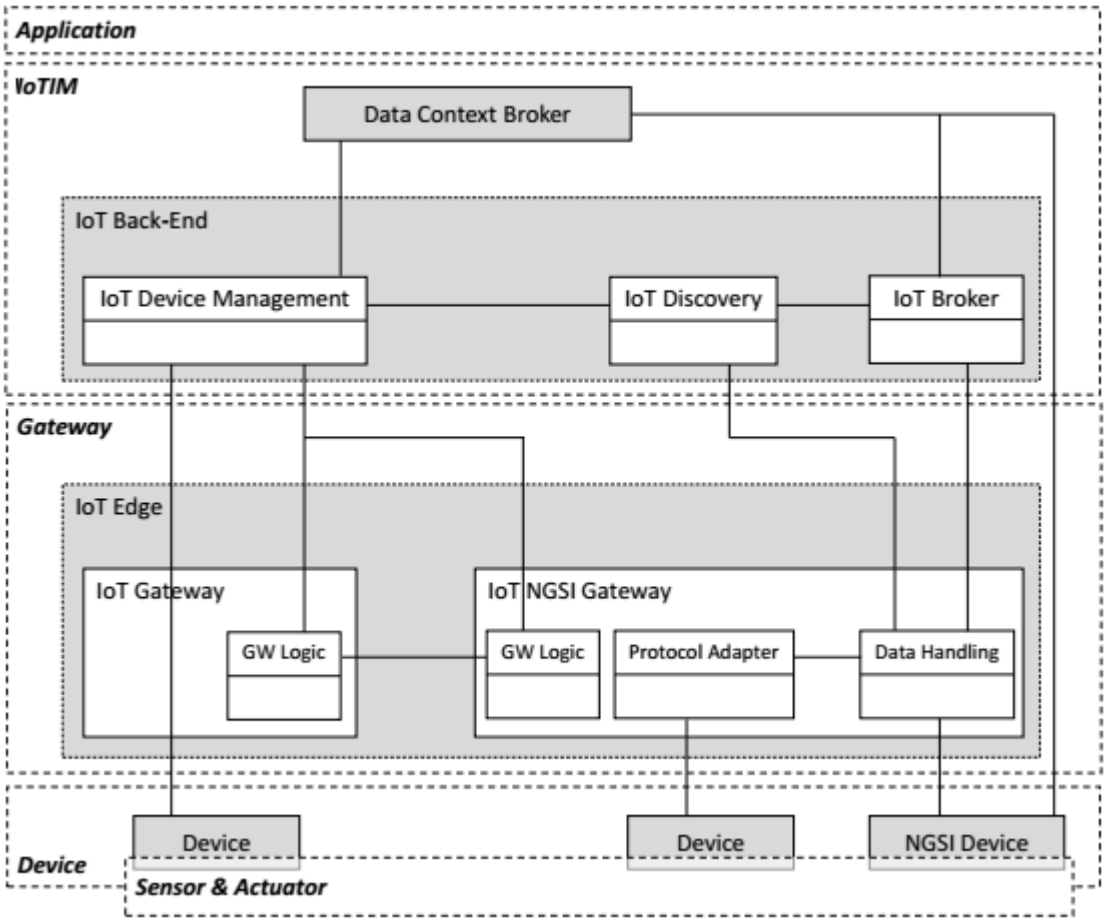

Рис. 2.4 Архітектура FIWARE

Щодо FIWARE, наша довідкова архітектура IoT охоплює кожен компонент архітектури. Як описано вище, визначення компонента пристрою відрізняється від нашого, отже, компоненти датчика, виконавчого механізму та пристрою частково перекриваються. Тим не менш, все ще існує відповідне відображення нашого визначення [9].

2.4.2 Архітектура OpenMTC

На рис. 2.5 показана архітектура OpenMTC, яка представляє собою cloud платформу IoT з відкритим вихідним кодом, і її порівняння з нашою еталонною архітектурою IoT. OpenMTC складається з наступних компонентів: зовнішній і внутрішній компоненти, компонент «Датчики та виконавчі механізми» під зовнішнім інтерфейсом, з'єднання між зовнішнім і внутрішнім інтерфейсом, додатки, розташовані у верхній частині внутрішнього інтерфейсу і на правий бік Front-End, а також компонент для підключення інших платформ M2M до Back-End. Очевидно, що компонент

«Sensors and Actuators» представляє наші датчики і пускачі. Компоненти «Датчики та виконавчі механізми» поряд з найнижчим рівнем Front-End, який забезпечує зв'язок, представляють пристрої, еквівалентні нашій еталонній архітектурі. Компоненти Core Features і Connectivity в Front-End, з'єднання між Front-Back і Back-End, а також в компоненті Connectivity Back-End надають всі необхідні функціональні можливості для забезпечення зв'язку між пристроєм і проміжним ПО, наприклад як переклад повідомлення. Отже, ці компоненти представляють наш шлюз. Оскільки компонент OpenEPC (забезпечує зв'язок між зовнішнім і внутрішнім інтерфейсом) надає додаткові функціональні можливості, такі як застосування правил і фільтрацію, він також охоплюється проміжним програмним забезпеченням інтеграції IoT. Крім того, проміжне ПО IoT Integration включає в себе компоненти «Можливості підключення», «Використання основних функцій» і «Активація додатків» серверної частини OpenMTC. Ці компоненти забезпечують основну функціональність платформи. Компоненти Application Enablement надають всі функціональні можливості для підключення інших додатків до проміжного програмного забезпечення. Отже, додатки та інші компоненти платформи M2M охоплюються компонентом додатків нашої еталонної архітектури.

Зважаючи на OpenMTC, його архітектуру можна відобразити на нашій довідковій архітектурі IoT. Тим не менш, компоненти пристрою, датчика та виконавчого механізму, а також проміжне програмне забезпечення шлюзу та IoT інтеграції частково перекриваються, що все ще відповідає визначенню нашої довідкової архітектури.

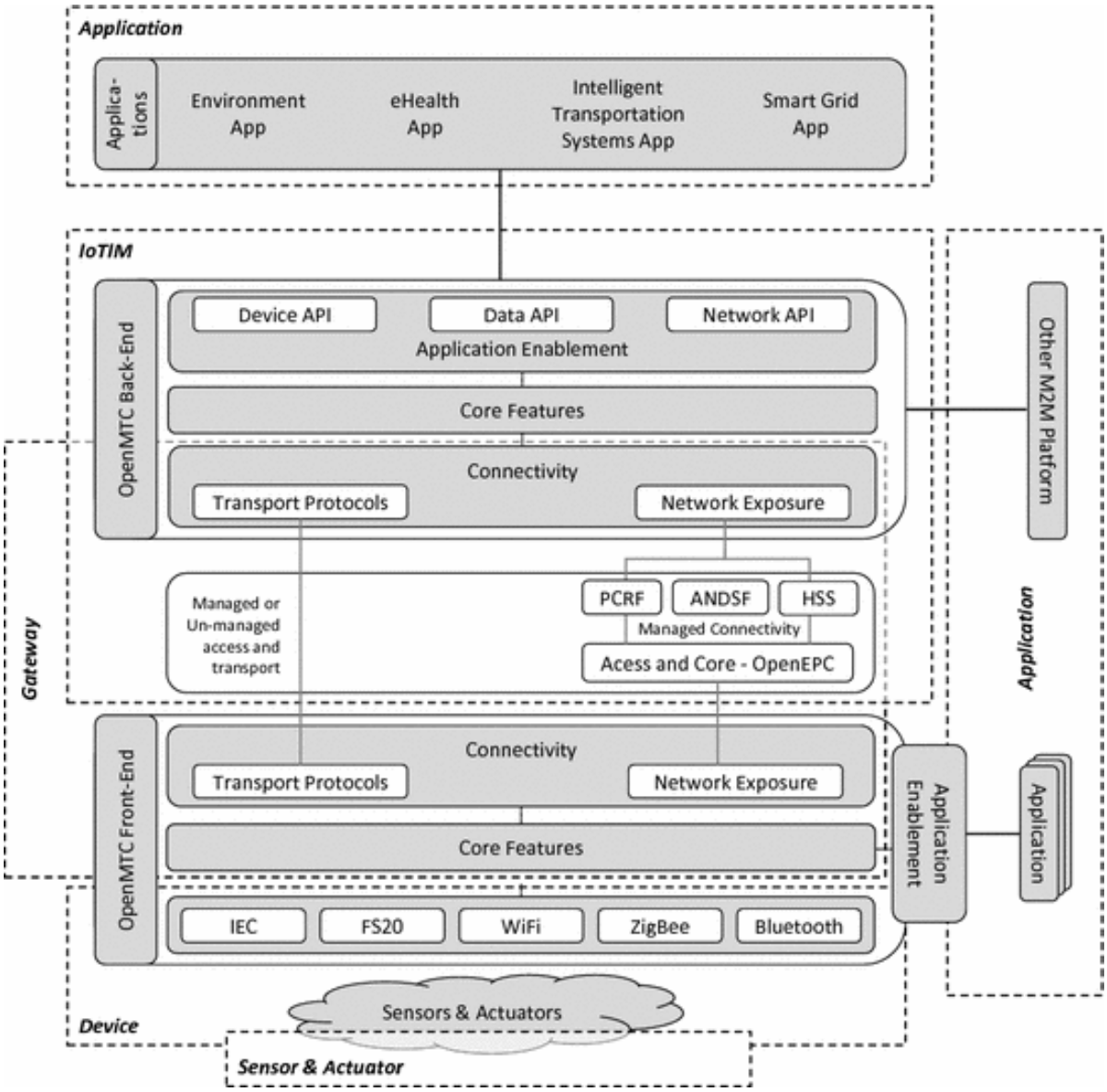

Рис. 2.5 Архітектура на основі OpenMTC (Детальний аналіз архітектури платформи IoT)[10]

2.4.3 Архітектура SiteWhere

На рис. 2.6 показана архітектура та відображення платформи IoT з відкритим кодом IoT на платформі SiteWhere на основі нашої довідкової архітектури. Дані від пристроїв та команд до компонентів пристроїв містять компонент Device довідкової архітектури. Крім того, вони також представляють наші компоненти датчика та виконавчого механізму, оскільки вони явно не зображені в архітектурі. Оскільки пристрої можуть спілкуватися через різноманітні протоколи з платформою, концепція шлюзу присутня між пристроями та платформою, але не зображена як окремий

компонент. Основну функціональність платформи забезпечує движок SiteWhereTenant, включаючи управління пристроями та механізм комунікацій. Отже, ці компоненти містяться в програмному забезпеченні IoT Integration Middleware нашої довідкової архітектури. Компонент API REST та інтеграція дозволяє підключити подальші програми до платформи.

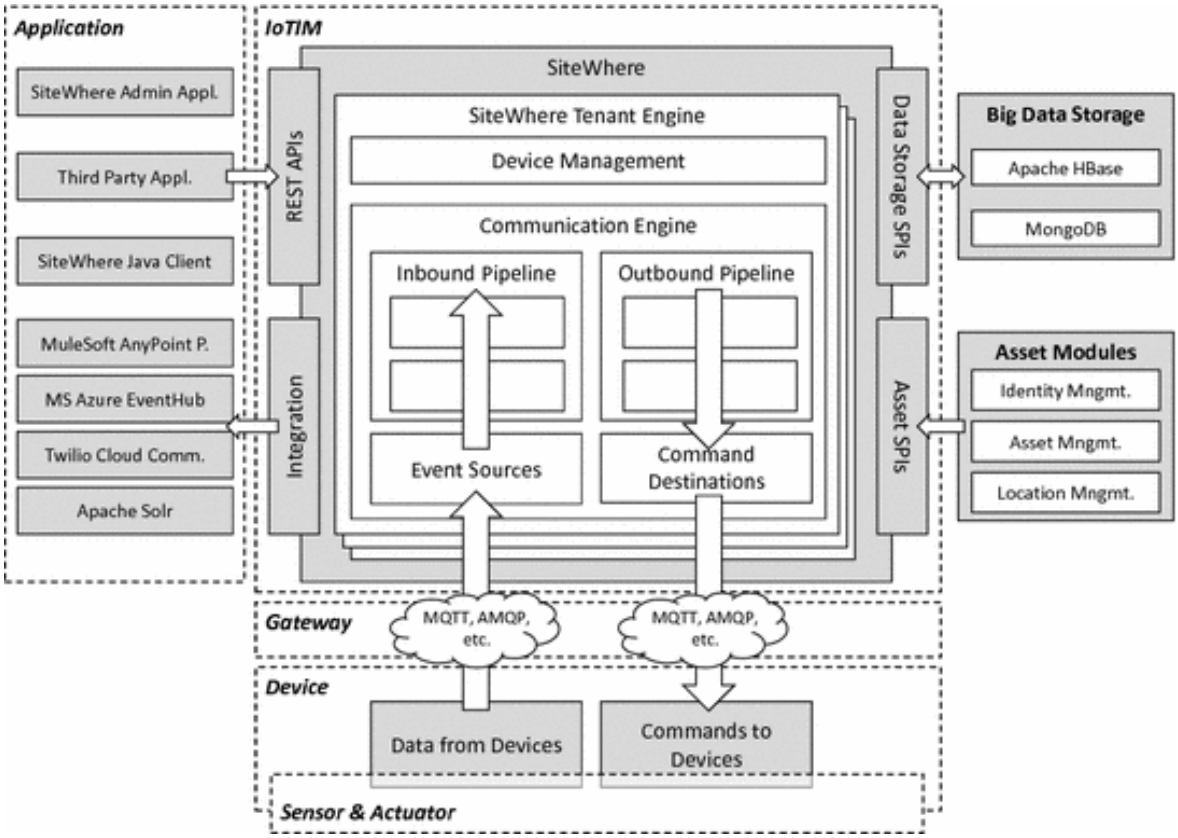

Рис. 2.6 Архітектура SiteWhere

Враховуючи архітектуру SiteWhere, представлений кожен компонент нашої довідкової архітектури IoT. Незважаючи на те, що деякі компоненти перекриваються, наше визначення компонентів базової архітектури IoT все ще має місце.

2.4.4 Архітектура Webinos

Посереднє програмне забезпечення та архітектура Webinos - це програмне забезпечення з відкритим кодом для IoT та мобільних пристроїв, спонсороване проектом Європейського Союзу FP7. Мета Webinos — забезпечити безпечну основу для спілкування персональних пристроїв та для осіб, які публікують дані третім сторонам та іншим особам. Як такий, він

потребує іншого підходу до платформи IoT, зосереджуючись навколо людини. Зображення підходу Вебінос на нашу довідкову архітектуру IoT показано на рис. 2.7. Основними компонентами архітектури Вебінос є центр персональної зони (PZH) та проксі-сервер персональної зони (PZP). PZH забезпечує шлюз, до якого підключається кожен пристрій. PZH також забезпечує локальну комунікацію між пристроями, виконуючи функції центру обміну повідомленнями. У зв'язку з цим він виконує функції нашого програмного забезпечення для інтеграції IoT. PZH по суті не підтримує жодних додатків для локального запуску, але він надає API, які дозволяють будувати сторонні програми та спілкуватися з пристроями, що є основною функцією інтеграції IoT проміжний рівень в нашій архітектурі. Кожен пристрій працює за допомогою локального компонента PZP, який агрегує дані сенсора та команди виконавчого механізму та спілкується з PZH. Одним із унікальних аспектів Webinos є те, що коли декілька PZP (на декількох пристроях) підключено до однієї і тієї ж PZH, вони можуть спілкуватися одночасно. Синхронізація PZP та PZH дозволяє це зробити.

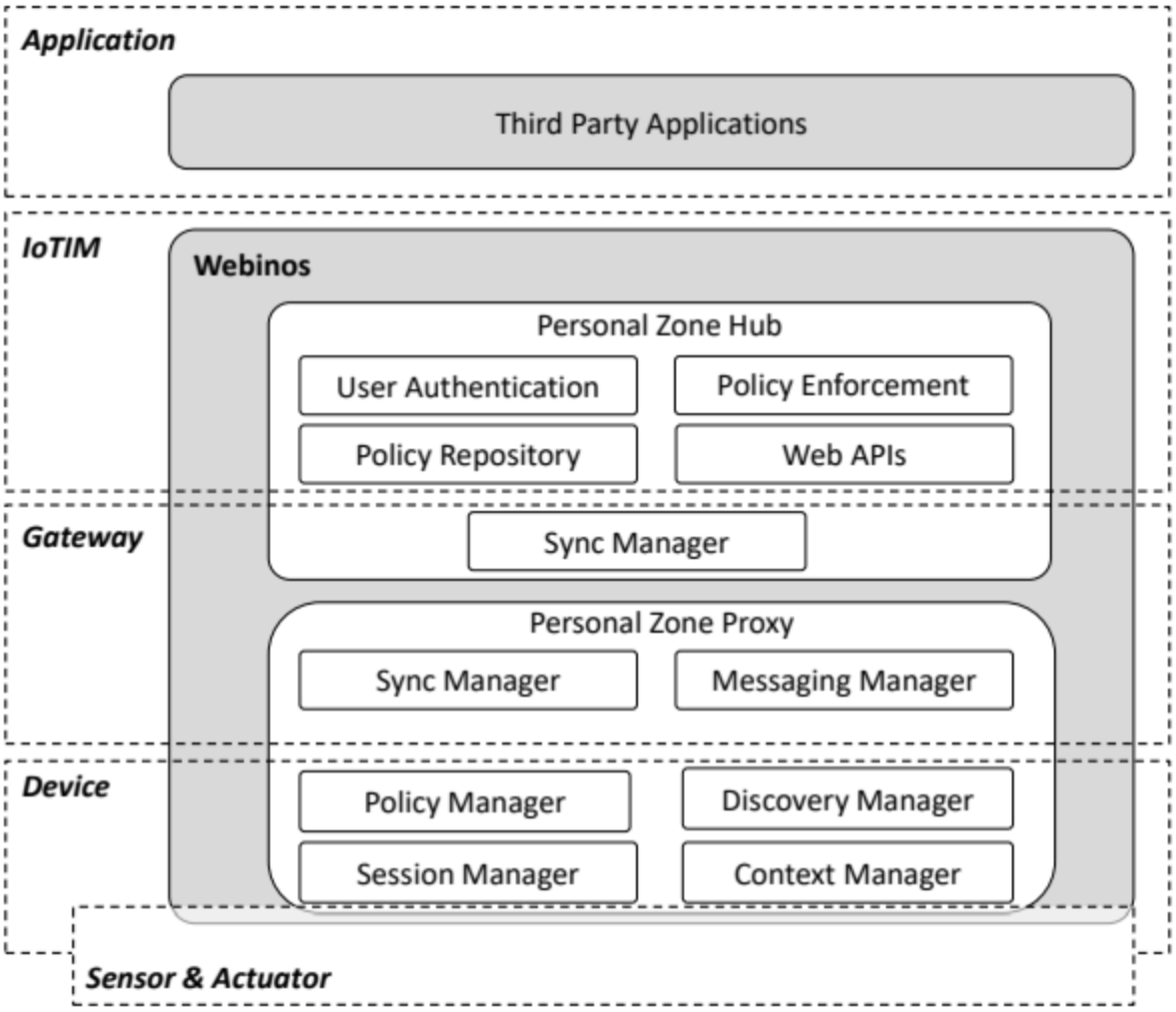

Рис. 2.7 Архітектура Webinos (A Detailed Analysis of IoT Platform Architectures)

Кожен компонент нашої довідкової архітектури IoT представлений у системі Webinos. Оскільки PZH і PZP перекриваються у функціях, розділення на нашу референтну архітектуру є складнішим. Однак існує чітке відображення того, що певні компоненти виконують функції Gateway та IoT Integration Middleware, як зображено на схемі. Як приклад, оскільки PZP можуть спілкуватися одночасно, без PZH виступати в якості посередника, ми повинні присвоїти деяку функцію шлюзу PZP. Кожен PZP розгорнуто на пристрої. Знову ж таки система Webinos явно не усуває різницю між пристроями, датчиками та виконавчими механізмами, але в Webinos є повна підтримка як датчиків, так і виконавчих механізмів, і, отже, вона підтримує довідкову архітектуру.

Хоча довідкова архітектура не стосується явно орієнтованого на особи підходу Webinos, ми можемо чітко зіставити систему Webinos кожної людини на окремий екземпляр еталонної архітектури.

2.4.5 Архітектура AWS IoT

На рис. 2.8 показана архітектура Amazon Web Services IoT (AWS IoT) та її відображення на нашій довідковій архітектурі. AWS IoT — це керована хмарна платформа для IoT, що переслідує концепцію Речей замість пристроїв. Оскільки AWS використовує речі, синонімічні пристроям із вбудованими датчиками та виконавчими механізмами, компонент "Налаштування" охоплюється нашими компонентами "Пристрій", "Датчик" та "Привід". Крім того, компонент шлюзу не представлений в архітектурі, але, виходячи з документації, він знаходиться між компонентами Things і Message Broker. Брокер повідомлень, Thing Shadows, Thing Registry, Rules Engine та компоненти безпеки та ідентичності забезпечують основну функціональність платформи. Отже, вони представляють компонент середнього програмного забезпечення IoT Integration Middleware нашої довідкової архітектури IoT. Наш компонент Application інкапсулює вже інтегровані служби обробки даних, такі як AWS Lambda або Amazon Kinesis, і, крім того, компонент IoT Applications, що дозволяє підключати подальші програми. Враховуючи AWS IoT, представлений кожен компонент нашої довідкової архітектури IoT. Незважаючи на те, що AWS дотримується іншої концепції пристроїв, це все одно відповідає нашому визначенню компонентів.

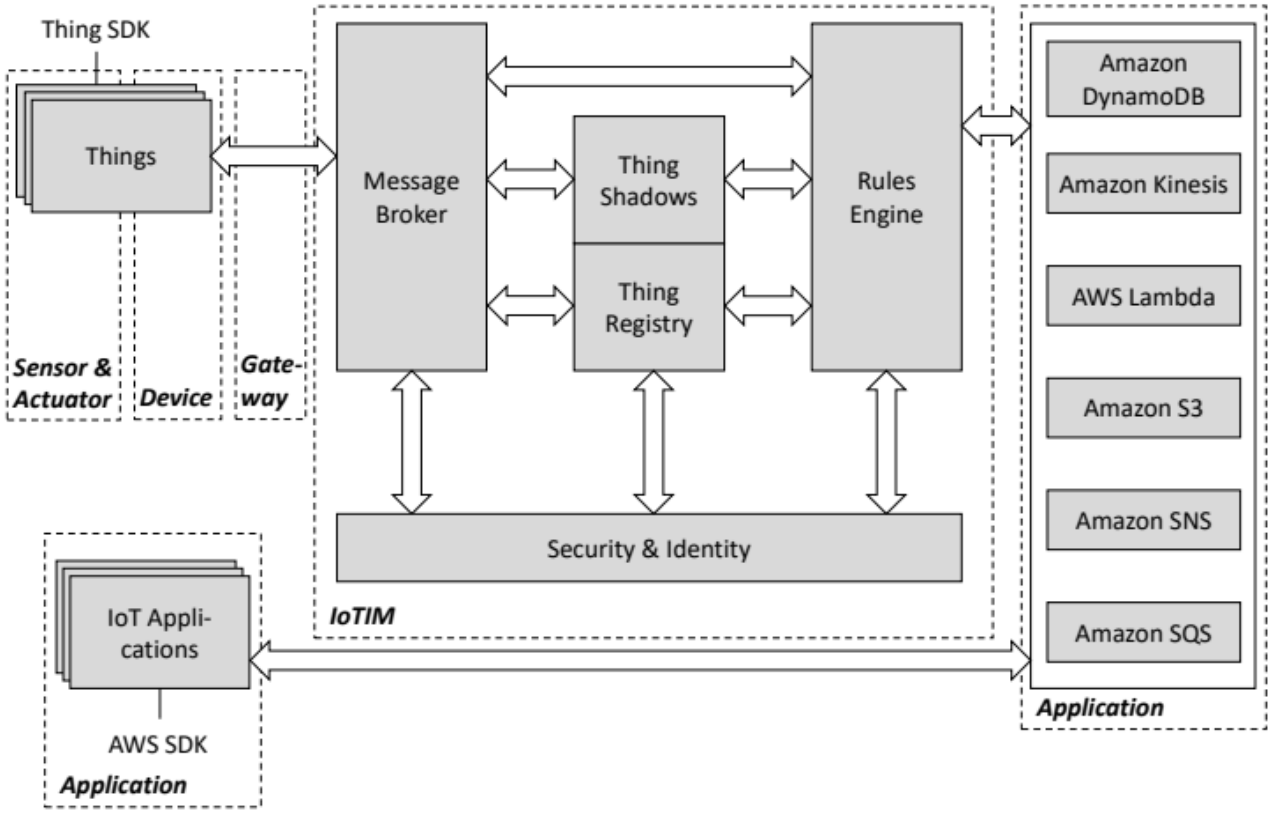

Рис. 2.8 Архітектура AWS IoT

2.4.6 Платформа IBM Watson IoT

На рис. 2.9 показана архітектура хмарної платформи Watson IoT на базі IBM. На рисунку пристрої, датчики та приводи не представлені. Оскільки компонент Connect піклується про підключення пристроїв до платформи, компоненти пристрою, датчика та виконавчого механізму нашої термінології частково охоплюють компонент Connect. Крім того, компонент Connect відповідає за відповідний переклад повідомлень, отже, він охоплений нашим компонентом шлюзу. Компонент Connect також забезпечує подальшу функціонування подій, таким чином, він також охоплюється нашим програмним забезпеченням IoT Integration Middleware. Крім того, компоненти Analytics, управління ризиками та управління інформацією забезпечують основну функціональність платформи, таким чином, вони охоплені програмним забезпеченням Інтеграції IoT нашої термінології. Компонент служб на основі відкритих стандартів Bluemix та компонент гнучких розгортань створюють основу платформи. Компоненти програм IoT

Industry Solutions і сторонні додатки охоплені нашим компонентом Application, оскільки вони дозволяють підключити подальші програми.

Зважаючи на платформу IBM Watson IoT, наша термінологія IoT може бути відображена на ній. Навіть незважаючи на те, що компоненти пристрою, датчика, виконавчого механізму та шлюзу не представлені зокрема, вони є частиною компонента Connect.

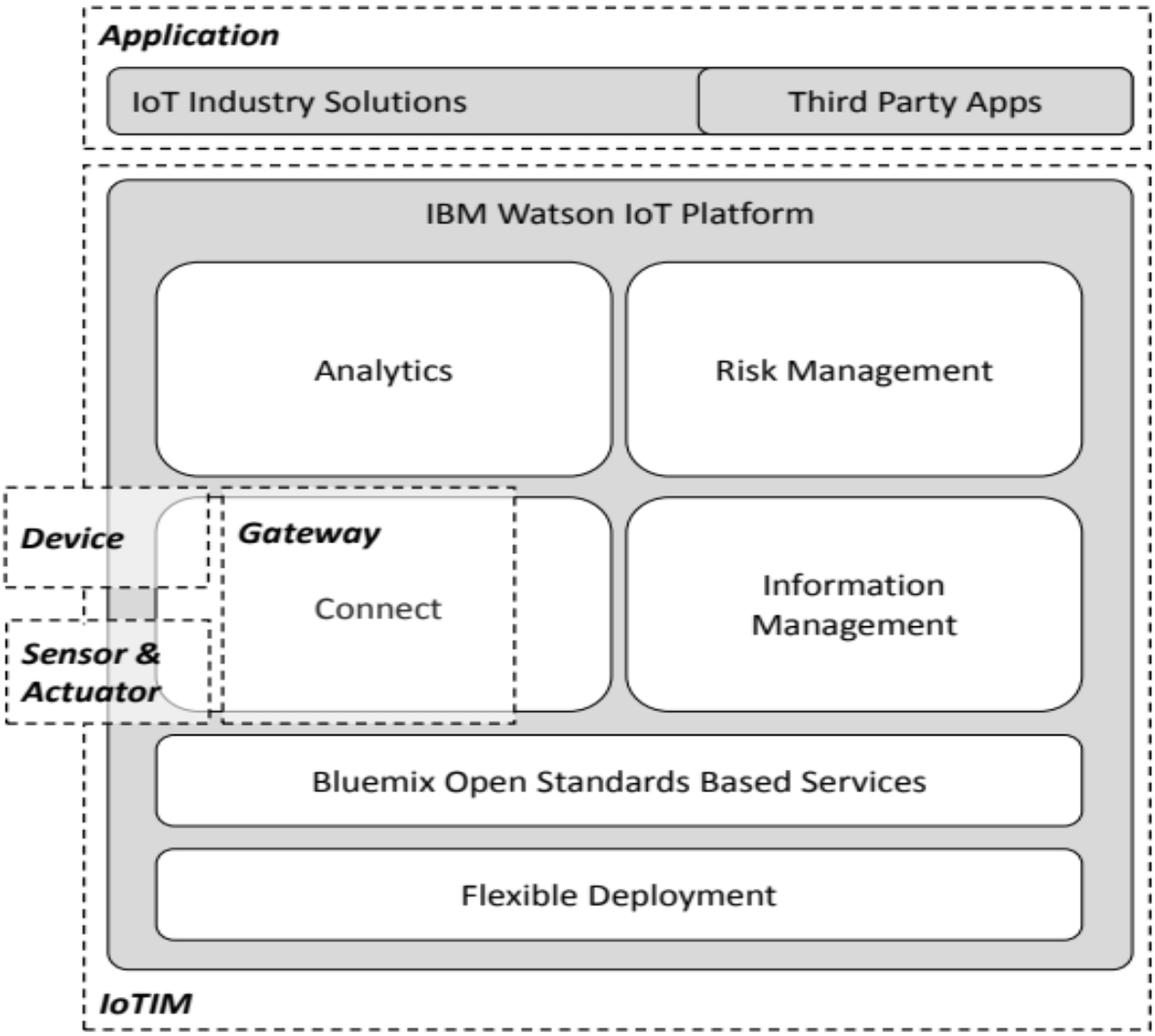

Рис. 2.9 Архітектура платформи IBM Watson IoT [8]

2.4.7 Microsoft Azure IoT Hub

Azure IoT Hub — це керована хмарна послуга, що надається Microsoft, яка виступає в якості центрального концентратора повідомлень для двобічного зв'язку між вашим додатком IoT і пристроями, якими він управляє. Ви можете використовувати Azure IoT Hub для створення рішень IoT з надійним і безпечним зв'язком між мільйонами пристроїв IoT і

серверною частиною хмарного рішення. До IoT Hub можна підключити практично будь-який пристрій.

IoT Hub підтримує зв'язок як з пристрою в хмару, так і з хмари на пристрій; кілька шаблонів обміну повідомленнями, таких як телеметрія між пристроями і хмарами; завантаження файлів з пристроїв і методи запиту-відповіді для управління вашими пристроями з хмари. Моніторинг IoT Hub допомагає підтримувати працездатність вашого рішення, відстежуючи такі події, як створення пристрою, збої пристрою і підключення пристрою.

Можливості IoT Hub допомагають створювати масштабовані, повнофункціональні рішення IoT, такі як управління промисловим обладнанням, яке використовується у виробництві, відстеження цінних активів в охороні здоров'я та моніторинг використання офісних будівель [11].

Архітектура Azure IoT Hub та її відображення на нашій довідковій архітектурі IoT показано на рис.2.10. Основним компонентом є концентратор IoT, до якого підключені всі інші компоненти. Оскільки Microsoft розмежовує лише пристрої з підтримкою IP (IP-сумісні) та пристрої персональної мережі (PAN—Personal Area Network), вони співставляються з нашими компонентами Пристрій, Датчик та Пускач.

Пристрої, що підтримують IP, можуть спілкуватися безпосередньо або через шлюзовий протокол хмарного протоколу з концентратором IoT, внаслідок чого пристроям PAN додатково потрібен польовий шлюз для виконання локальних служб управління, таких як управління доступом та інформаційним потоком. Отже, хмарний протокол та польовий шлюз охоплені нашим компонентом шлюзу. Основна функціональність рішення забезпечується IoT-концентратором, обробкою подій та інформацією, бізнес-логікою пристроїв, моніторингом підключення та компонентами забезпечення та управління прикладними пристроями, отже, вони охоплені нашим програмним забезпеченням IoT Integration Middleware. Крім того, компонент забезпечення пристрою та управління пристроєм також дозволяє підключити подальші програми.

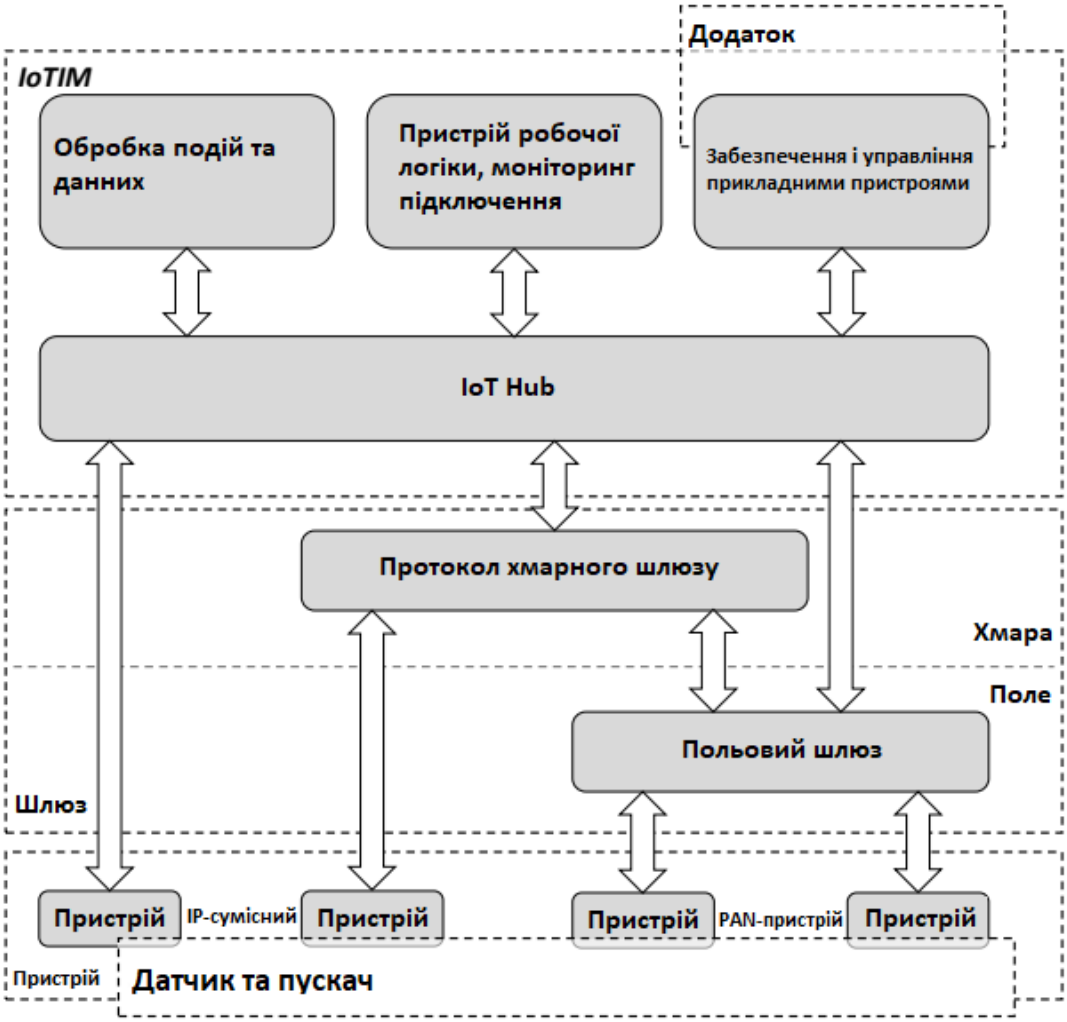

Рис. 2.10 Архітектура Microsoft Azure IoT Hub [8]

2.4.8 SmartThings

SmartThings — це платформа IoT, що надається Samsung для розумного домашнього середовища. Архітектура та відповідне відображення з нашою довідковою архітектурою IoT показано на рис. 2.11. Вона складається з трьох основних елементів, а саме: Обробники типів пристроїв, Обробка підписки та Система управління додатками, включаючи управління SmartApp & Execution, де всі додаткові компоненти підключені до. SmartThings поєднує датчики, пускачі, пристрої, користувачі та речі в межах одного компонента. Далі вони розрізняють цей складений компонент та інший компонент Clients (-Devices). Оскільки Клієнти (-пристрої) можуть також містити датчики та виконавчі пристрої, обидва компоненти охоплені нашим компонентом пристрою, при цьому частки датчика та приводу частково перекриваються. Оскільки обробники типів пристроїв переводять повідомлення про події в

нормалізовану подію SmartThings, а Hub Connectivity та Client Connectivity дозволяють підключити Пристрої до платформи, ці компоненти представляють Шлюз. Основна функціональність платформи забезпечується Обробкою підписки та Системою управління додатками, включаючи компоненти SmartApp Management & Execution.

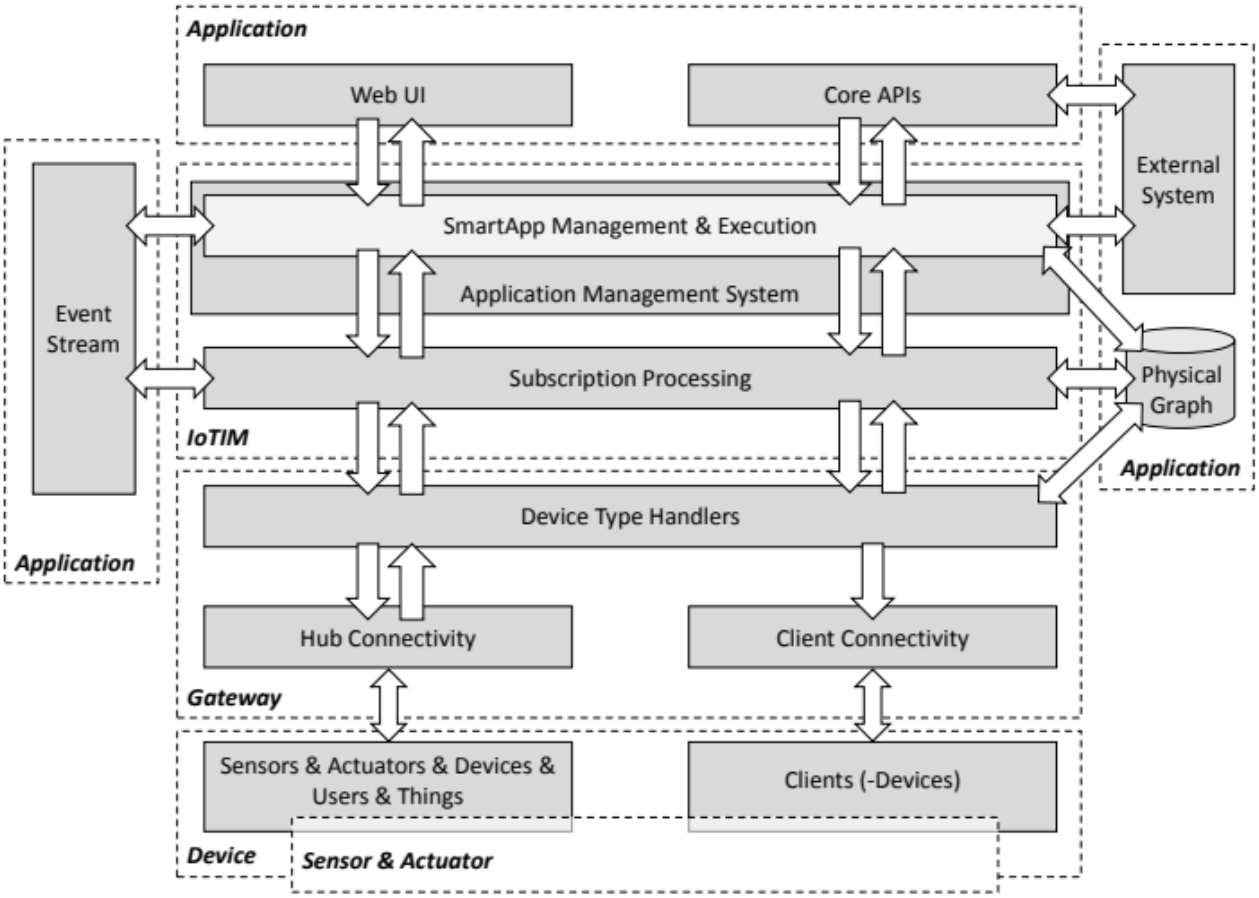

Рис. 2.11 Архітектура SmartThings

Отже, вони охоплюють проміжне програмне забезпечення IoT Integration з нашої довідкової архітектури. Компоненти потоку подій, веб-інтерфейс, основні API, зовнішня система та фізичний графік представляють можливі підключені програми до платформи. Щодо SmartThings, наша довідкова архітектура IoT охоплює кожен компонент архітектури. Як описано вище, компоненти датчика, виконавчого механізму та пристрою перекриваються, оскільки SmartThings використовує їх у складених компонентах. Тим не менш, це відповідає нашому визначенню [8].

2.4.9 Підсумок порівняння

Наша довідкова архітектура IoT, відображаючи кожне порівняння, описане вище, є представлена у межах кожної розглянутої платформи. У таблиці 1.3 показаний узагальнений огляд порівняння: рядки визначаються компонентами нашої довідкової архітектури, а стовпці відображають кожну розглянуту платформу IoT, внаслідок чого комірки таблиці вказують, що кожен компонент платформи IoT відповідає нашим компонентам довідкової архітектури.

Тільки архітектура OpenMTC та SmartThings являє собою компонент Sensor and Actuator. Усі решта платформ, окрім Microsoft Azure IoT Hub, лише згадують про ці компоненти у своїй документації. Компонент Device не представлений в архітектурі OpenMTC та платформи IBM Watson IoT, але згадується в документації. Решта платформ представляють собою компонент Device в їх архітектурі. Крім того, платформи FIWARE, AWS IoT та Microsoft Azure IoT Hub далі розрізняють поняття «Інтелектуальні» пристрої, які вже включають певну логічну функціональність. Далі ці «інтелектуальні» пристрої охоплені нашими компонентами Device, Gateway та IoT Integration Middleware, відповідно до рівня інтегрованої логічної функціональності. Оскільки ці відмінності у визначенні та деталізації компонентів платформ IoT є, ми відобразили їх на наші чітко відокремлені компоненти, щоб уточнити поняття та використану деталізацію. Крім SiteWhere та AWS IoT, всі платформи представляють концепцію нашого шлюзу в межах їх архітектури. Тим не менш, документація SiteWhere та AWS IoT також охоплює функціональність нашого шлюзу. Очевидно, кожна платформа являє собою основну функціональність, тобто наше програмне забезпечення для інтеграції IoT в архітектурі. Відмінності полягають у деталізації та кількості компонентів, що містять функціональні можливості програмного забезпечення IoT Integration Middleware. Крім того, кожна платформа забезпечує можливість підключення подальших додатків. FIWARE не є відповідним компонентом у своїй архітектурі, але він згадується в документації.

Таблиця 2.3 Підсумок порівняння платформ IoT [8]

Платформа	**FIWARE**	**OpenMTC**	**SiteWhere**	**Webinos**	**AWS IoT**	**IBM Watson IoT Platform**	**MS Asure Iot Hub**	**Samsung SmartThings**
Датчик /Пускач	-	Датчик та пускач	-	-	-	-	-	Датчик, пускач, прилад, користувач, Things
Прилад	Прилад/NGSI прилад	-	Дані від приладів + команди для приладів	PZP: політика + сесія	Things	-	Прилад	Датчик, пускач, прилад, користувач, Things + клієнт
Шлюз	IoT Кінцева точка	Основні особливості ядра + Підключення + Зворотній звязок	-	PZP: синхронізація + менеджер повідомлень + PZP: менеджер синхронізації	-	Підключення	Протокол хмарного шлюзу + польовий шлюз	Хаб та клієнт підключення + обробники типів пристроїв
Програмне забезпечення інтеграції IoT	IoT бек-енд + брокер контексту даних	Зворотній звязок + Особливості ядра + застосування програми	SiteWhere Terant Engine	PZH: аутентифікація користувача + політика сховища + політика примусового виконання + Web APIs	Брокер повідомлень Thing Shadows + Thing Registry + дані двигуна + безпека та ідентичність	Аналіз + Управління ризиками + підключення + управління інформацією + послуги, засновані на відкритих стандартах Bluemix + гнучке розгортання	IoT hub + обробка подій та даних + пристрій робочої логіки, моніторинг підключення + забеспечення і управління прикладними пристроями	Система управління додатками + обробка підписки

Додато к	-	Додатки + Інші M2M платформи	Інтеграція + REST компоненти	Додатки третьої сторони	Amazon сервіси + IoT додатки	IoT рішення в галузі індустрії + Додатки третьої сторони	Забезпечення та управління прикладними пристроями	Потік подій + Web UI + Core APIs + Зовнішня система + Фізичний графік

Висновки:

IoT повільно перетворюється з бачення в реальність: платформи IoT відіграють центральну роль у цій еволюції, забезпечуючи значні складові. Відсутність стандартизації та розвитку в бункерах призвело до неоднорідного ландшафту платформи. Внаслідок такої неоднорідності порівняння та вибір однієї з цих платформ є складним завданням. Вони не тільки використовують різні поняття та технології, але й термінологія чітко не визначена. Багато понять та частини цих платформ описуються синонімами чи омонімами або відрізняються деталізацією.

Для вирішення цих проблем було представлено довідкову архітектуру IoT, яка базується на існуючих платформах. Було визначено кожен компонент і описано зв'язок між ними. Ці компоненти не обов'язково повинні залишатися відокремленими, але їх можна комбінувати. Ми порівняли нашу довідкову архітектуру з вісьма існуючими платформами, чотири з яких є відкритим кодом. Компоненти довідкової архітектури співпадають із компонентами існуючих платформ. Порівнюючи або оцінюючи різні платформи, наша довідкова архітектура IoT може бути корисним інструментом, якщо забезпечити загальну основу, на якій базуватимуться нові проекти платформи IoT.

Ця робота створить основу для підходу щодо прийняття рішень, який передбачає вибір платформ IoT на основі заданих користувачем переваг. Це допоможе користувачеві визначити відповідне рішення IoT для свого випадку.

3. ОЦІНКА ПРОДУКТИВНОСТІ АРХІТЕКТУРИ AWS IoT

3.1 Основні компоненти AWS IoT

AWS IoT задовольняє безпечне двостороннє спілкування між пристроями, підключеними до Інтернету, такими як датчики, пускачі, вбудовані мікроконтролери або інтелектуальні пристрої та Cloud AWS. Це дає можливість отримувати телеметричні дані з кількох пристроїв, а також зберігати та аналізувати дані.

AWS IoT містить наступні компоненти:

- Шлюз пристроїв. Формує основу взаємодії між підключеними пристроями і функціональними можливостями хмари, такими як сервіс правил, тіні пристроїв, а також іншими сервісами AWS і сторонніми сервісами. Він підтримує шаблон передачі повідомлень «видавець-підписник», який дозволяє забезпечити масштабовану взаємодію з мінімальною затримкою і низькою вартістю. Це особливо корисно для тих сценаріїв IoT, де очікується регулярна взаємодія з мільярдом пристроїв з мінімальною затримкою. Шаблон «видавець-підписник» передбачає публікацію повідомлень клієнтами в логічних каналах зв'язку, званих «темами», при цьому для отримання повідомлень клієнти повинні підписуватися на ці теми. Шлюз пристроїв забезпечує взаємодію між видавцями і підписниками. До сих пір організаціям доводилося виділяти, експлуатувати, масштабувати і підтримувати свої власні сервери в якості шлюзів пристроїв, щоб користуватися всіма перевагами шаблону «видавець-підписник». Сервіс усунув цей бар'єр, створивши шлюз пристроїв AWS IoT Core.

- Брокер повідомлень. Забезпечує захищений механізм для пристроїв і програм IWT AWS для публікації та отримання повідомлень один від одного. Для публікації та підписки може використовуватись протокол MQTT або MQTT через WebSocket, чи інтерфейс HTTP REST.

- Сервіс правил. Забезпечує обробку повідомлень та інтеграцію з іншими службами AWS. Може використовуватись мова на базі SQL, щоб

вибрати дані з корисних інформаційних повідомлень, а потім обробляти та надсилати дані іншим службам, таким як Amazon S3, Amazon DynamoDB та AWS Lambda. Також може використовуватись брокер повідомлень, щоб повторно опублікувати повідомлення іншим абонентам.

- Служба безпеки та ідентифікації. Забезпечує спільну відповідальність за безпеку. Пристрої повинні зберігати свої облікові дані в безпеці, щоб надсилати їх до брокера повідомлень. Брокер повідомлень та сервіс правил використовують функції захисту AWS для безпечного надсилання даних на пристрої або інші служби AWS.

- Тіні пристроїв. Дозволяють хмарним і мобільним додаткам просто взаємодіяти з підключеними пристроями, зареєстрованими в AWS IoT. Тінь пристрою в AWS IoT містить властивості підключеного пристрою. Можна визначити будь-який набір властивостей, що відносяться до конкретного прикладу використання. Наприклад, для «розумної» лампочки можна визначити наступні властивості: «увімкнена або вимкнена», «колір» і «яскравість». Очікується, що підключений пристрій буде повідомляти про фактичні значення цих властивостей, які будуть зберігатися в тіні пристрою. Додатки отримують і оновлюють ці властивості, просто використовуючи API RESTful, що надається сервісом AWS IoT. Синхронізацію значень властивостей між підключеним пристроєм і його тінню забезпечує сервіс AWS IoT і SDK для пристроїв AWS IoT.

- Реєстр. Сценарії IoT можуть варіюватися від декількох критично важливих пристроїв і до величезної групи пристроїв. Реестр дозволяє впорядковувати і відслідковувати ці пристрої. У реєстрі можна підтримувати логічний ідентифікатор для кожного пристрою, який підключений до AWS IoT. Кожен пристрій в реєстрі може бути однозначно визначеним і описаним з допомогою метаданих, наприклад, номера моделі, контактної інформації для підтримки, а також сертифікатів, пов'язаних з цим пристроєм. Пошук підключених пристроїв у реєстрі можна виконати на основі цих метаданих.

- Груповий реєстр. Дозволяє управляти декількома пристроями одночасно, розділивши їх на групи. Групи також можуть містити підгрупи,

що створює певну ієрархію. Будь-які дії, які виконуються над батьківською групою, застосовуватимуться до її дочірніх груп, а також до всіх пристроїв у ній та у всіх її дочірніх групах. Дозволи, надані групі, будуть застосовуватися до всіх пристроїв та у всіх її дочірніх групах.

- Цифровий двійник. Документ JSON, який використовується для зберігання та отримання поточної інформації про стан для пристрою. Забезпечує стійкі уявлення про пристрої в хмарі. Оновлену інформацію про стан пристрою можна опубліковувати в службі, а пристрій синхронізує цей стан при підключенні. Пристрої також можуть опублікувати свій поточний стан у тінь для використання додатками або іншими пристроями.

- Служба автентифікації за замовчуванням. Ви можете визначити власні авторизатори, які дозволяють керувати вашою власною стратегією автентифікації та авторизації за допомогою спеціальної служби автентифікації та функції лямбда. Користувальницькі авторизатори дозволяють AWS IoT перевіряти автентичність пристроїв та дозволяти операції за допомогою автентифікації та стратегій авторизації токенів носіїв. Користувальницькі авторизатори можуть реалізовувати різні стратегії автентифікації (наприклад, перевірка JWT, виклик провайдера OAuth тощо) і повертати документи політики, які використовуються шлюзом пристрою для авторизації операцій MQTT.

- Jobs service. Дозволяє визначити набір віддалених операцій, які відправляються і виконуються на одному або декількох пристроях, підключених до AWS IoT. Наприклад, можна визначити завдання, яке наказує пристроям завантажити та встановити оновлення програм, перезавантаження, обертання сертифікатів або виконання віддалених способів усунення несправностей. Для виконання ціє задачі необхідно вказати опис віддалених операцій, які потрібно виконати, і список цілей, які повинні виконувати їх. Цілі можуть бути окремими пристроями або групами [12].

3.1.1 Принцип роботи AWS IoT

AWS IoT дозволяє підключеним до Інтернету пристроям підключитися до Cloud AWS і дозволяє додаткам у хмарі взаємодіяти з пристроями, підключеними до Інтернету. Звичайні додатки IoT збирають та обробляють телеметрію з пристроїв або дозволяють користувачам дистанційно керувати пристроєм. Пристрої звітують про стан, публікуючи повідомлення в форматі JSON на теми MQTT. Кожна тема MQTT має ієрархічне ім'я, яке ідентифікує пристрій, стан якого оновлюється. Коли повідомлення публікується в темі MQTT, повідомлення надсилається брокеру повідомлень AWS IoT MQTT, який відповідає за надсилання всіх повідомлень, опублікованих у MQTT-темі, для всіх клієнтів, які підписалися на цю тему.

Зв'язок між пристроєм та AWS IoT захищено за допомогою сертифікатів X.509. Сертифікат може бути створено автоматично, або завантажено користувачем. У будь-якому випадку сертифікат повинен бути зареєстрований та активований за допомогою AWS IoT, а потім скопійовано на ваш пристрій. Коли ваш пристрій спілкується з AWS IoT, він подає сертифікат AWS IoT як посвідчення.

Користувачем створюються правила, які визначають одну чи кілька дій для виконання на основі даних у повідомленні. Наприклад, можна, оновлювати або запитувати дані з таблиці DynamoDB або запустити певну функцію. Правила використовують вирази для фільтрування повідомлень. Коли правило збігається з повідомленням, двигун правил викликає дію, використовуючи вибрані властивості. Правила також містять роль IAM, яка надає AWS IoT дозвіл на ресурси AWS, які використовуються для виконання дії. Архітектура AWS IoT зображена на рисунку 3.1.

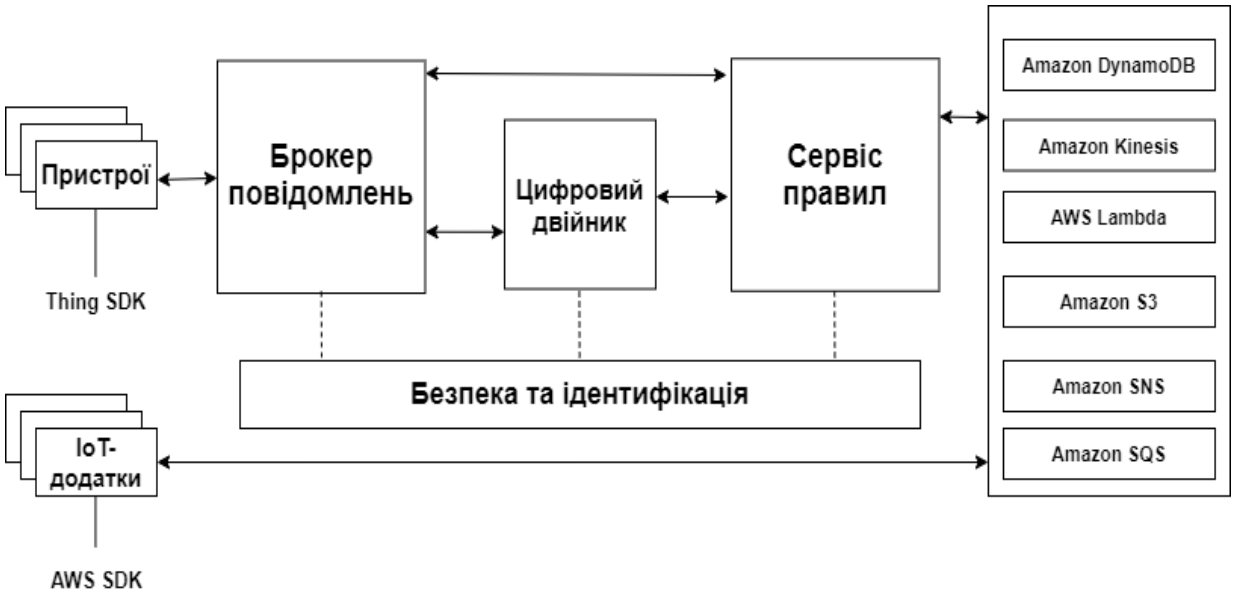

Рисунок 3.1 Архітектура AWS IoT

Кожен пристрій має цифрового двійника, який зберігає та витягує інформацію про стан. Кожен елемент у інформації про стан має дві записи: стан, останній з якого повідомляється пристроєм, і бажаний стан запитується програмою. Програма може запитати поточну інформацію про стан для пристрою. Двійник відповідає на запит, надаючи документ JSON з інформацією про стан (як повідомляється, так і за бажанням), метадані та номер версії. Програма може керувати пристроєм, надіславши запит на зміну свого стану. Цифровий двійник приймає запит на зміну стану, оновлює інформацію про стан та надсилає повідомлення про те, що інформація про стан була оновлена. Пристрій отримує повідомлення, змінює стан і повідомляє про його новий стан [13].

3.1.2 Принцип роботи сервісів

Щоб організувати роботу сервісу у першу чергу треба створити акаунт AWS. Щойно відбудеться підключення AWS IoT Button до мережі Wi-Fi та отримання сертифікату і приватного ключа для даної кнопки, одразу буде здійснено безпечне підключення до AWS IoT Core. Кнопка публікуватиме повідомлення в темі під час натискання. Налаштування маршрутизації подій можна зробити за допомогою сервісу правил AWS IoT одиничним, подвійним або тривалим натисканням кнопки. Також можна налагодити відправку повідомлень собі самому через Amazon SNS або збереження даних

при натисканні в таблиці Amazon DynamoDB. За допомогою функції AWS Lambda, написаної на Node.js, Python або Java, можна запрограмувати власну логіку і налаштувати функцію на підключення до інших пристроїв або чужих сервісів , які підключено до IoT.

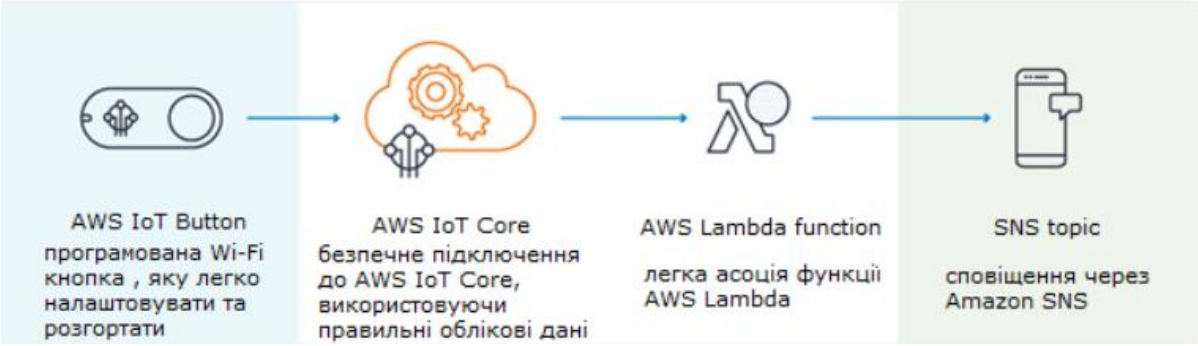

Рисунок 3.2 Основні блоки AWS IoT

AWS IoT безпосередньо синхронізується з такими службами AWS:

• *Amazon Simple Storage Service* — забезпечує масштабоване зберігання в Cloud AWS. Сучасні компанії повинні вміти просто та безпечно збирати, зберігати та аналізувати дані у величезному масштабі. Amazon S3 - це об'єкт зберігання, призначений для зберігання та вилучення будь-яких обсягів даних з будь-яких джерел: веб-сайтів і мобільних додатків, корпоративних додатків, а також даних з датчиків або пристроїв IoT. Сервіс гарантує надійність зберігання на рівні 99,999999999% і використовується для зберігання мільйонів прикладних програм, що застосовуються у всіх галузях промисловості. S3 відкриває найширші можливості для забезпечення безпеки та відповідає самим строгим нормативним вимогам. Сервіс надає клієнтам гнучкість у керуванні даними, забезпечуючи оптимізацію витрат, контроль доступу та відповідність вимогам. S3 надає функціональні можливості для виконання запитів до сервісу без вилучення, що дозволяє використовувати потужні аналітичні інструменти для безпосередньої обробки даних, що зберігаються в S3.

• *Amazon DynamoDB* — надає керовану базу даних NoSQL. Це швидкий і гнучкий сервіс баз даних NoSQL. Він підходить для будь-яких додатків, що вимагають стабільної роботи із затримкою не більше кількох мілісекунд на будь-який масштаб. Ця повністю керована обласна база даних підтримує

роботу на базі як документів, так і пар «ключ-значення». Гнучка модель даних, стабільна продуктивність та автоматичне масштабування пропускної здатності роблять цей сервіс відмінною платформою для мобільних і інтернет-додатків, ігор, реклами, IoT та багатьох інших додатків. Amazon DynamoDB Accelerator (DAX) - це повністю керований високоактивний кеш пам'яті, який дозволяє зменшити час відгуку DynamoDB з мілісекунд до мікросекцій навіть при обробці мільйонів запитів у секунду.

• *Amazon Kinesis*—дозволяє в реальному часі обробляти потокові дані в масовому масштабі. За допомогою Amazon Kinesis можна просто збирати, обробляти і аналізувати потокові дані в режимі реального часу, щоб своєчасно отримувати аналітичні результати і швидко реагувати на нову інформацію. Надані службою Amazon Kinesis основні можливості дозволяють економічно обробляти поточні дані на будь-якому масштабі, а також забезпечують гнучкість при виборі інструментів, які оптимально відповідають вимогам програми. Amazon Kinesis дає можливість в режимі реального часу збирати такі дані, як відео- та аудіопотоки, додатки журналів, події навігації користувачів по веб-сайтам і телеметричні дані Інтернету для машинного навчання, аналізу та інших додатків. Amazon Kinesis дозволяє обробляти і аналізувати дані за моментом поступу і реагувати миттєво, а не чекати, поки всі дані будуть зібрані, щоб розпочати їх обробку.

• *AWS Lambda* — запускає ваш код на віртуальних серверах від Amazon EC2 у відповідь на події. AWS Lambda дозволяє запускати програмні коди без виділення серверів та керування ними. За допомогою Lambda можна запускати практично будь-які види програм і серверних сервісів, при цьому виконувати які-небудь операції адміністрації не потрібно. Просто завантажте програмний код, і Lambda забезпечить всі ресурси, необхідні для його виконання та масштабування, з високою ступенем доступності. Можна налаштувати автоматичний запуск програмного коду з інших сервісів AWS або безпосередньо з будь-якого мобільного або веб-додатка.

• *Amazon Simple Notification Service* — це гнучка, повністю керована служба передачі повідомлень по моделі «публікація-підписка» та

повідомлення для мобільних пристроїв, що дозволяє координувати доставку повідомлень для підписаних кінцевих точок і клієнтів. За допомогою SNS можна надсилати повідомлення більшій кількості отримувачів, включаючи розподілені системи та сервіси, а також мобільні пристрої. Сервіс дозволяє надійно надсилати повідомлення з усіма кінцевими точками при будь-якому масштабі. Розпочати роботу з SNS можна за кілька хвилин, використовуючи Консоль управління AWS інтерфейс командного рядка AWS або AWS SDK всього з трьома простими API. SNS дозволяє забути про складності та додаткові витрати, пов'язані з обслуговуванням виділеної ПО та інфраструктури для відправлення повідомлень, а також з керуванням такими.

• *Amazon Simple Queue Service* — зберігає дані в черзі, які потрібно завантажити за допомогою додатків. Служба простої черги Amazon (SQS) — повністю керована послугами повідомлень, що дозволяє легко ізолювати та масштабувати мікросервіси, розподілені системи та безсерверні додатки. Одна з сучасних рекомендацій по розробці ПО — створення архітектури, в якій кожний окремий компонент рішення виконує свою конкретну задачу. Така підхід дозволяє підвищити масштабність та надійність готової системи. SQS дозволяє легко та економічно ізолювати компоненти обласного додатка та координувати їх роботу. За допомогою SQS можна відправляти, зберігати та отримувати повідомлення компонентів програмного забезпечення в будь-якому масштабі без втрати повідомлень і необхідності забезпечити постійну доступність інших сервісів [13].

3.2 Фактори, що впливають на продуктивність Web-служб

Розглядаються фактори, що впливають на продуктивність Web-служб, а також приводиться методика розробки простих кількісних моделей для оптимального вибору протоколу по критерію продуктивності, що залежить від числа користувачів. Web-служба — це послуга, що отримується через Internet, яка виконує поставлені завдання або виробляє транзакції. Web-служби мають модульну структуру і є самодостатніми модулями, які можна описати, опублікувати і виконати через Internet. Web-служба може являти

собою бізнес-процес, додаток і навіть обчислювальний ресурс. До прикладів Web-служб можна зарахувати все, починаючи з послуг електронних платежів і закінчуючи послугами поширення вмісту і зберігання інформації. Web-служби можна автоматично викликати з додатків, а не тільки з браузерів.

Бізнес стає все більш залежним від використання Web-служб, тому їх робочі характеристики (тобто продуктивність) стають життєво важливими. Чим більше інформації і послуг можна отримати від компанії, тим більше запитів він отримує. А чим більше запитів отримує Web-служби, тим більше ймовірність, що користувачі занадто довго чекатимуть реакції на свій запит. І тоді, як правило, Web-користувачі (можливі покупці) будуть розчаровані і будуть відмовлятися від надання послуг.

Час відгуку і пропускна здатність — це два найбільш важливих показника продуктивності Web-систем. Зазвичай під пропускною спроможністю Web-служби розуміють швидкість обслуговування запитів, що вимірюється кількістю операцій в секунду. Через великого розмаїття розмірів запитуваних Web-об'єктів, пропускну здатність також вимірюють в бітах в секунду (біт / с, bps).

Незважаючи на те, що затримка установки зв'язку впливає на сприйняття користувачами сайту, в той же час вона є нетривалим, одноразово діючим фактором продуктивності сайту, тобто він вимірюється один раз на запит і не відображає якість всього сеансу [13].

Отже, до найбільш часто використовуваних показників продуктивності Web- служб відносяться:

• пропускна здатність, запитів / с;

• пропускна здатність, Мбіт / с;

• час відгуку при наскрізний передачі даних;

• навантаження центрального процесору.

3.3 Опис системи та розрахунок пропускної здатності

В офісній будівлі розміщено кавомашини, на які закріплено кнопку Amazon Dash. Коли в машині закінчується кава, користувач натискає кнопку.

Після цього на смартфон адміністратора надходить сповіщення. Адміністратор виконує багатофакторну автентифікацію, обирає підходящі дату і час та погоджує замовлення. У цифровому двійнику кнопки запрограмована вся потрібна інформація: від цифрового ідентифікатора до виду та виробника необхідної кави. Виконується обробка запиту сервером та відправка кур'єру. Оплата відбувається після доставки кави та підтвердження менеджером оплати. Дослідження було проведено шляхом реалізації сервісу IoT, який можна реалізувати на транспортному рівні протоколами MQTT чи HTTP. Сервіс розгорнуто на інфраструктурі Amazon.

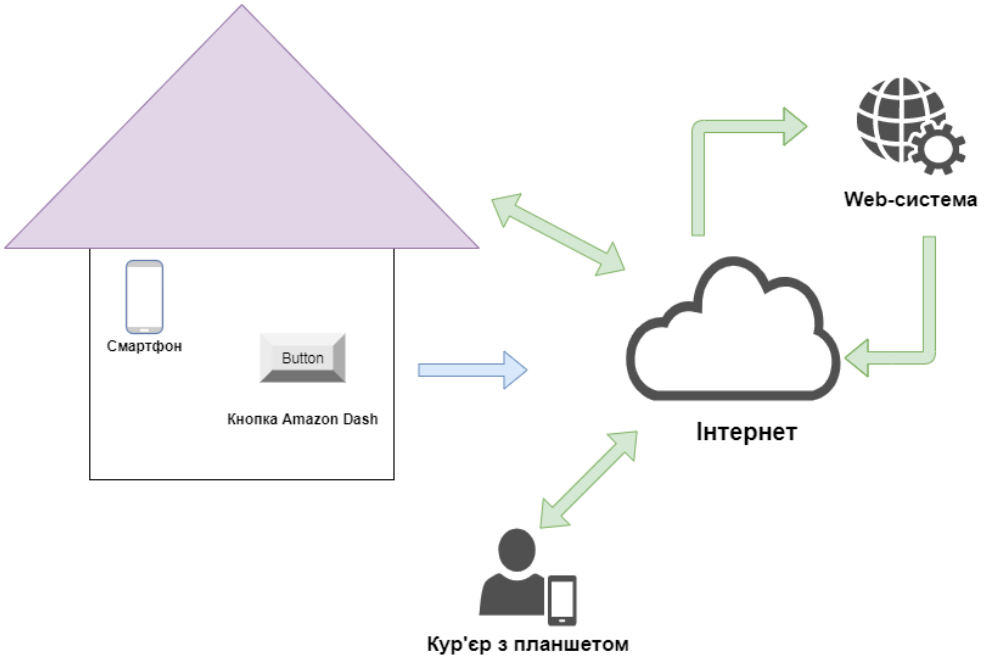

Рисунок 3.3 Схематичне відображення бізнес-процесу

Далі приводяться типи запитів, що беруть участь у повному циклі від натискання кнопки до підтвердження замовлення:

1 тип – замовлення кави;

2 тип – вибір зручної дати та підтвердження замовлення;

3 тип – багатофакторна автентифікація;

4 тип – підтвердження доставки;

5 тип – оплата.

Зауважимо, що для кожного типу може бути різний об'єм даних. Наприклад представимо інформацію для простого замовлення кави. Далі наведено об'єм інформації простого цифрового двійника та детального.

а) Простий варіант (10 B)

1) цифровий ідентифікатор – 0225536658

б) Детальний (2 kB):

1) цифровий ідентифікатор — 0548946844

2) вид кави — арабіка

3) тип кави — зернова

4) адреса — м. Київ, вул.Анни Ахматової, буд 23

5) вага — 3 кг

6) вимоги до кави - кожна упаковка кави в зернах повинна мати етикетку, на якій буде вказано: назва кави; рівень обсмажування; походження; дата виготовлення; строк придатності до споживання чи дата закінчення строку придатності до споживання; умови зберігання; показники якості; найменування, місцезнаходження виробника і місце виготовлення.

В таблицях 3.1 та 3.2 наведено розміри повідомлень для різних типів запитів.

Таблиця 3.1 Варіації розміру повідомлень при використанні HTTP

	Розміри повідомлень, Біт									
Тип запиту	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
1	1290	2290	6290	8290	9290	11290	16290	20290	36290	40290
2	5290	7290	12290	15290	18290	20290	25290	30290	50290	77290
3	10290	12290	16290	17290	19290	22290	24290	26290	30290	33290
4	3290	6290	10290	12290	14290	17290	20290	28290	33290	100290
5	100290	110290	120290	140290	165290	170290	200290	250290	260290	270290

Таблиця 3.2 Варіації розміру повідомлень при використанні MQTT

	Розміри повідомлень, Біт									
Тип запиту	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
1	1002	2002	6002	8002	9002	11002	16002	20002	36002	40002
2	5002	7002	12002	15002	18002	20002	25002	30002	50002	77002
3	10002	12002	16002	17002	19002	22002	24002	26002	30002	33002
4	3002	6002	10002	12002	14002	17002	20002	28002	33002	100002
5	100002	110002	120002	140002	1650002	170002	200002	250002	260002	270002

Мережевий трафік має пульсуючий характер. Мається на увазі, що передача даних відбувається випадковим чином, з піковими швидкостями, які перевищують середні швидкоссті в 8-10 разів. Також відомо, що мережевий трафік варіюється в декількох масштабах часу. Нехай у даній компанії працює 1000000 робітників, кожен з яких

Припустимо, що компанія обслуговує 1000000 клієнтів. Кожен клієнт замовляє каву 1 раз на тиждень. Навантаження має піковий характер. Замовлення відбуваються з 07:00 до 10:00 та з 17:00 до 21:00. З цього випливає, що кожного дня тривалість замовлень становить 7 годин.

Пропускна здатність сервера, виражена в запитах/с, складатиме:

$$\frac{1000000}{7 * 7 * 3600} = 5{,}66 \frac{\text{запитів}}{\text{с}}, \qquad (3.1)$$

Проте, цей показник не дає ніякого уявлення з приводу пропускної здатності мережі, що використовується в період спостереження і не дає

уявлення про розміри типів запитів. Отже, для оцінки пропускної здатності мережі чи мережевого адаптера потрібно розрахувати пропускну здатність в Біт/с.

На рисунку 3.4 наведено ймовірності того, що кожен з типів запитів відбудеться.

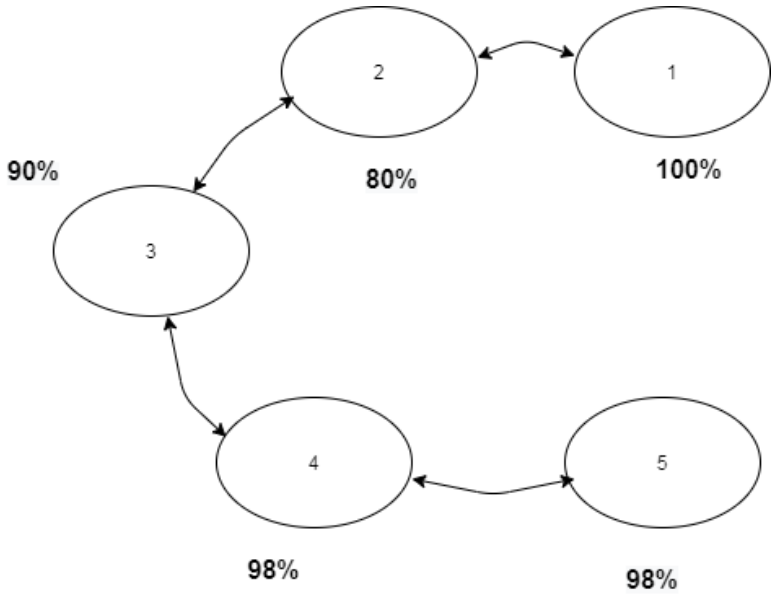

Рисунок 3.4 Ймовірності виконання запитів

Для того, щоб визначити, яку частку складає кожен з типів в усій передачі, виконаємо наступні розрахунки:

$$\frac{1}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)}=0.255, \qquad (3.2)$$

$$\frac{0.8}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)}=0.204, \qquad (3.3)$$

$$\frac{(0.8*0.9)}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)}=0.184, \qquad (3.4)$$

$$\frac{(0.8*0.9*0.98)}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)}=0.180, \qquad (3.5)$$

$$\frac{(0.8*0.9*0.98*0.98)}{1+0.8+(0.8*0.9)+(0.8*0.9*0.98)+(0.8*0.9*0.98*0.98)}=0.177, \qquad (3.6)$$

Розрахуємо пропускну здатність для кожного типу запиту:

$$\text{Пропускна здатність} = \frac{\text{загальна к} - \text{кість запитів} * \text{частка кожного з типів} * \text{середній розмір}}{\text{час спостереження}}, \quad (3.7)$$

Розрахунки для всіх типів першого варіанту:

$$\frac{1000000 * 0{,}255 * (1290 * 8)}{176400} = 14918{,}3 \left[\frac{\text{Біт}}{\text{с}}\right] \quad (3.8)$$

$$\frac{1000000 * 0{,}204 * (5290 * 8)}{176400} = 48941{,}5 \left[\frac{\text{Біт}}{\text{с}}\right] \quad (3.9)$$

$$\frac{1000000 * 0{,}184 * (10290 * 8)}{176400} = 85866{,}67 \left[\frac{\text{Біт}}{\text{с}}\right] \quad (3.10)$$

$$\frac{1000000 * 0{,}180 * (3290 * 8)}{176400} = 26857{,}14 \left[\frac{\text{Біт}}{\text{с}}\right] \quad (3.11)$$

$$\frac{1000000 * 0{,}177 * (100290 * 8)}{176400} = 805049{,}00 \left[\frac{\text{Біт}}{\text{с}}\right] \quad (3.12)$$

Загальна пропускна здатність – це сума пропускних здатностей для кожного з типів:

$$\text{Загальна пропускна здатність} = 14918{,}3 + 48941{,}5 + 85866{,}67 + 26857{,}14 + 805049$$
$$= 981632{,}7 \left[\frac{\text{Біт}}{\text{с}}\right] \quad (3.13)$$

Отже, пропускна здатність для першого варіанту складає 981,633 Кбіт/с.

Розрахунки для інших варіантів виконані за аналогією в Excel, результати наведені в таблицях 3.3 та 3.4.

Таблиця 3.3 Пропускна здатність при використанні HTTP

	Пропускна здатність, Біт/с HTTP									
Тип запиту	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
1	14918,37	26482,99	72741,5	95870,75	10743 5,4	13056 4,6	18838 7,8	23464 6,3	41968 0,3	46593 8,8

2	48941,5	67444,9	113703,4	14145,5	169213,6	187717	233975,5	280234	465268	715063,9
3	85866,67	102556	135934,7	144279,4	160968,7	186002,7	202692,1	219381,4	252760,1	277794,1
4	26857,14	51346,94	84000	100326,5	116653,1	141142,9	165632,7	230938,8	271755,1	818693,9
5	805049	885321,1	965593,2	1126137	1326818	1366954	1607770	2009131	2089403	2169675
Σ	981632,7	1133152	1371973	1608072	1881089	2012381	239458	2974332	3498867	4447166

Таблиця 3.4 Пропускна здатність при використанні MQTT

	Пропускна здатність, Біт/с MQTT									
Тип запиту	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
1	11587,76	23152,38	69410,88	92540,14	104104,8	127234	185057,1	231315,6	416349,7	462608,2
2	46277,01	64780,41	111038,9	138794	166549,1	185052,5	231310	277569,5	462603,5	712399,5
3	83463,4	100152,7	133531,4	141876,1	158565,4	183599,5	200288,8	216978,1	250356,8	275390,8
4	24506,12	48995,92	81648,98	97975,51	114302	138791,8	163281,6	228587,8	269404,1	816342,9
5	802737,1	883009,3	963281,4	1123826	1324506	1364642	1605458	2006819	2087091	2167363
Σ	968571,4	1120091	1358912	1595012	1868027	1999320	2385397	2961270	3485805	4434104

Таблиця 3.5 Загальна пропускна здатність при використанні MQTT

	Пропускна здатність, Біт/с									
Тип протоколу	Варіант 1	Варіант 2	Варіант 3	Варіант 4	Варіант 5	Варіант 6	Варіант 7	Варіант 8	Варіант 9	Варіант 10
MQTT	968571,4	1120091	1358912	1595012	1868027	1999320	2385397	2961270	3485805	4434104
HTTP	981632,7	1133152	1371973	1608072	1881089	2012381	2398458	2974332	3498867	4447166
Δ	1,013485	1,011661	1,009611	1,008188	1,006992	1,006533	1,005475	1,004411	1,003747	1,002946

Рисунок 3.5 Відношення загальної пропускної здатності

З отриманих результатів можна зробити висновок, що мережеве з'єднання повинно матиж мінімальну пропускну здатність 5 Мбіт/с, рекомендовано 10 Мбіт/с. А при збільшенні розмірів запитів, різниці між протоколами майже немає. Тому при великих файлах доцільніше використовувати HTTP, бо він не потребує додаткових налаштувань при

розгортанні. При передачі запитів, та відповідей невеликих розмірів, доцільніше використовувати MQTT.

3.4 Середній час відгуку

Визначення затримок, та “вузьких місць” відіграє важливу роль для того, щоб робити зміни в програмному забезпеченні, модернізації обладнання або установки, використовувати більш швидкісні ліній передачі даних. Затримки можна поділити на 3 масштабні складові:

1. Процесор. Центральний процесор відповідальний за обробку даних з пристроїв, а також за ініціалізацію зв’язку.
2. Мережа. Мережа породжує затримки під час доставки інформації від клієнта до сервера і назад від сервера до клієнта. Ці затримки є функціями різних компонентів на шляху між клієнтом і сервером, таких як модеми, маршрутизатори, лінії передачі, мости і комутатори.
3. Сервер. Час перебування на сервері, R'_{Server} - це час, що витрачається на виконання запиту. Воно включає в себе час обробки і час очікування на різних компонентах сервера, таких як процесор, диск і мережевий адаптер.

Коли запитуваний документ не перебуває в кеші клієнта, час відгуку R на запит являє собою суму часу перебування запитів на всіх ресурсах:

$$R_{miss} = R'_{CPU} + R'_{Network} + R'_{Server}, \quad (3.14)$$

де R'_{CPU} — час обробки процесором.

$R'_{Network}$ — час передачі в мережі.

R'_{Server} — час перебування на сервері.

Розглянемо середній час відгуку для запропонованих протоколів. Припустимо, що документи не кешуються. Середній час перебування на сервері (R'_{Server}) – 3,6 с., а час, затрачений процесором на обробку отриманих даних (R'_{CPU}) – 0,3с. Пропускна здатність мережі складає 100 Мбіт/с.

Час передачі в мережі ($R'_{Network}$) дорівнює розміру даних в бітах, поділеному на пропускну здатність мережі. Продемонструємо розрахунки

для першого варіанта. Розрахунки для інших варіантів було проведено в програмі Excel, результати наведено в таблиці 3.6

$$R'_{Network(HTTP)} = \frac{120450 * 8}{100000000} = 0.009639, [c] \quad (3.15)$$

$$R'_{Network(MQTT)} = \frac{119010 * 8}{100000000} = 0{,}009521, [c] \quad (3.16)$$

$$R'_{miss(HTTP)} = 0{,}009639 + 0{,}3 + 3{,}6 = 3{,}909636\ [c] \quad (3.17)$$

$$R'_{miss(MQTT)} = 0{,}009521 + 0{,}3 + 3{,}6 = 3{,}909521\ [c] \quad (3.18)$$

$$\Delta = \frac{R_{miss(HTTP)}}{R_{miss(MQTT)}}; \quad (3.19)$$

$$\Delta = \frac{3{,}909636}{3{,}909521} = 1{,}000029467 \quad (3.20)$$

Таблиця 3.6 Середный час відгуку

	Варіант									
Протокол	1	2	3	4	5	6	7	8	9	10
HTTP	3,9096	3,9110	3,9132	3,9154	3,9181	3,9193	3,9229	3,9284	3,9328	3,9417
MQTT	3,9095	3,9109	3,9131	3,9153	3,9180	3,9192	3,92280	3,92832	3,9327	3,94160
Δ	1,000029467	1,000029456	1,000029439	1,000029423	1,000029403	1,000029394	1000029367	1,000029326	1,000029293	1,000029227

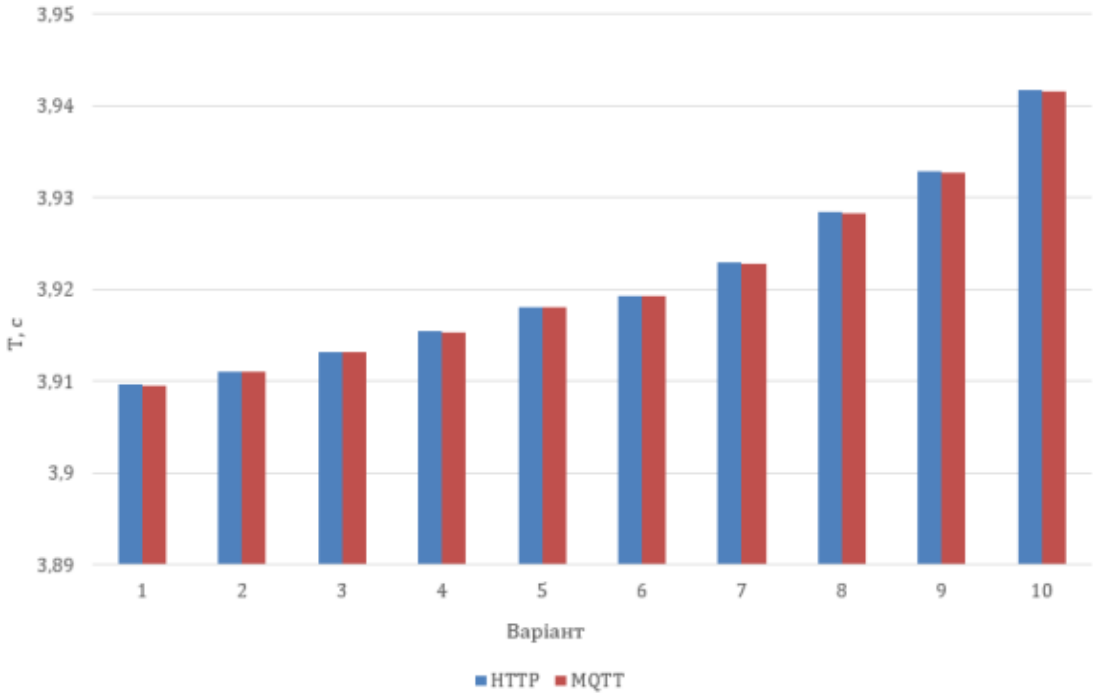

Рисунок 3.6 Порівняння середнього часу відгуку для протоколів

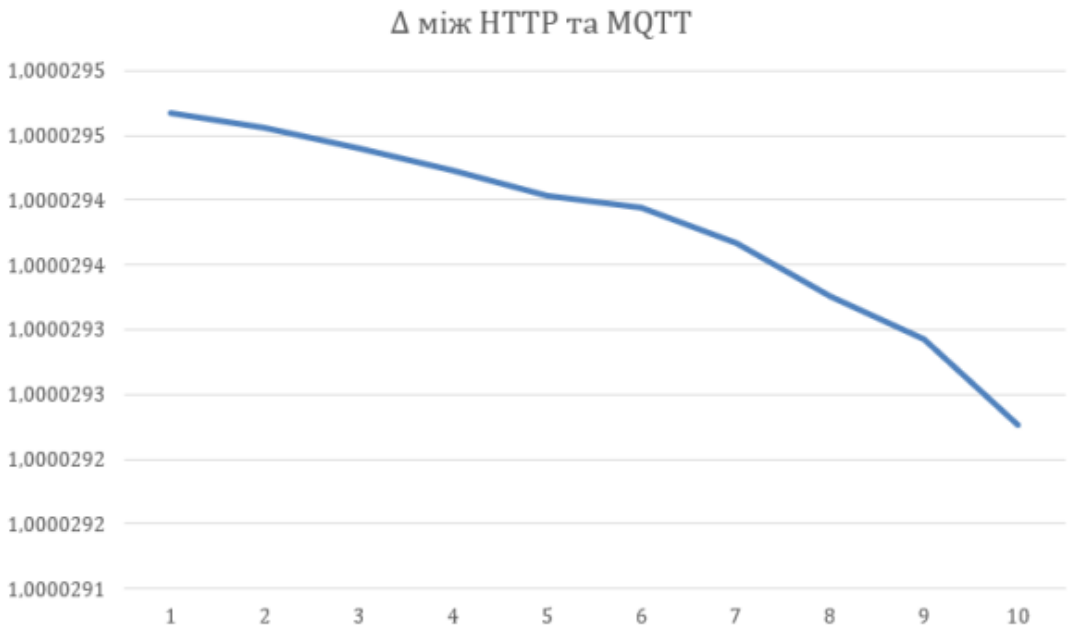

Рисунок 3.7 Відношення середнього часу відгуку

Висновки:

У третьому розділі було проведено оцінку продуктивності мережі AWS IoT.

Проаналізовано сервіс IoT, який розгорнуто на інфраструктурі AWS Amazon та реалізований протоколами MQTT або HTTP на транспортному рівні .

AWS IoT забезпечує безпечний двосторонній обмін даними між пристроями, підключеними до Інтернету, такими як датчики, пускачі, вбудовані мікроконтролери або інтелектуальні пристрої та Cloud AWS. Це дає змогу збирати телеметричні дані з кількох пристроїв, а також зберігати та аналізувати їх.

Запропонований метод складається з наступних розрахунків:

1) Розрахунок пропускної здатності для 5 типів запитів та різних розмірів повідомлень. Отримані результати демонструють, що для передачі маленьких повідомлень краще використовувати протокол MQTT. При передачі великих файлів, краще HTTP, бо він не потребує додаткових налаштувань при розгортанні.

2) Середній час відгуку. З отримах результатів, бачимо, що різниці між протоколами майже не існує.

ВИСНОВКИ

У ході виконання магістерської дисертації було проаналізовано технології IoT та загальні принципи їх використання в різних галузях. Було здійснено детальний огляд сервіс-орієнтованої архітектури IoT.

У другому розділі розглянуто узагальну архітектуру для системи Інтернету речей та її компоненти. Проаналізовано різні архітектурні рішення, складено таблицю порівняння функіональних компонентів цих рішень з компонентами уніфікованої архітектури. У результаті цього стало зрозуміло, що довідкова архітектура IoT може бути корисним інструментом, якщо забезпечити загальну основу, на якій базуватимуться нові проекти платформи IoT.

Ця робота створить основу для підходу щодо прийняття рішень, який передбачає вибір платформ IoT на основі заданих користувачем переваг. Це допоможе користувачеві визначити відповідне рішення IoT для свого випадку.

У третьому розділі було проведено оцінку продуктивності мережі AWS IoT. Запропонований метод складається з наступних розрахунків:

1) Розрахунок пропускної здатності для 5 типів запитів та різних розмірів повідомлень. Отримані результати демонструють, що для передачі маленьких повідомлень краще використовувати протокол MQTT. При передачі великих файлів, краще HTTP, бо він не потребує додаткових налаштувань при розгортанні.

2) Середній час відгуку. З отримах результатів, бачимо, що різниці між протоколами майже не існує.

Даний метод розрахунку можна використовувати для будь-якого архітектурного рішення IoT.

ПЕРЕЛІК ПОСИЛАНЬ

1. IoT Technology, Applications and Challenges: A Contemporary Survey [Електронний ресурс] — Режим доступу: https://link.springer.com/article/10.1007/s11277-019-06407-w .

2. Применение BLUETOOTH-технологии в интернете вещей [Електронний ресурс] — Режим доступу: https://cyberleninka.ru/article/n/primenenie-bluetooth-tehnologii-v-internete-veschey/viewer .

3. Internet of Things in Industries: A Survey [Електронний ресурс] — Режим доступу: https://www.researchgate.net/publication/270742269_Internet_of_Things_in_Industries_A_Survey .

4. The history and future of the internet of things [Електронний ресурс] — Режим доступу: https://www.itransition.com/blog/iot-history .

5. IoT Technology & Protocols – 7 Important IoT Communication Protocols [Електронний ресурс] — Режим доступу: https://data-flair.training/blogs/iot-technology/ .

6. Сети нового поколения и IoT: действующие сетевые протоколы и перспективы 5G [Електронний ресурс] — Режим доступу: https://iotconf.ru/ru/article/seti-novogo-pokoleniya-i-iot-deystvuyushchie-setevie-protokoli-i-perspektivi-5g-95783 .

7. R. van Kranenburg, The Internet of Things: A Critique of Ambient Technology and the All-Seeing Network of RFID. Amsterdam, The Netherlands: Institute of Network Cultures, 2007 .

8. A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences [Електронний ресурс] — Режим доступу: https://www.researchgate.net/publication/320454222_A_Detailed_Analysis_of_IoT_Platform_Architectures_Concepts_Similarities_and_Differences .

9. WHAT IS FIWARE? [Електронний ресурс] — Режим доступу: https://www.fiware.org/about-us/ .

10. Fraunhofer FOKUS: OpenMTC Platform Architecture (2016). URL http://www.openmtc.org/index.html#MainFeatures .

11. What is Azure IoT Hub? [Електронний ресурс] — Режим доступу: https://docs.microsoft.com/en-us/azure/iot-hub/about-iot-hub .

12. What Is AWS IoT? [Електронний ресурс] — Режим доступу: https://docs.aws.amazon.com/iot/latest/developerguide/what-is-aws-iot.html .

13. Обзор Amazon Web Services [Електронний ресурс] — Режим доступу: https://d1.awsstatic.com/whitepapers/ru_RU/aws-overview.pdf .

14. V. A. F. Almeida, D. A. Menasc.e, R. Riedi, F. P. Ribeiro, R. Fonseca, and W. Meira Jr., "Analyzing Web Robots and their Impact on Caching," Proc. Sixth Workshop on WebCa ching and Content Distribution, Boston, Massachusetts, June 20-22, 2001.