Міністерство освіти і науки України

Запорізька державна інженерна академія
(повне найменування вищого навчального закладу)

Факультет енергетики, електроніки та інформаційних технологій
( назва факультету)

Кафедра програмного забезпечення автоматизованих систем
(повна назва кафедри )

# Пояснювальна записка

до магістерської роботи

рівень вищої освіти другий (магістерський)
(другий (магістерський) рівень)

на тему Дослідження та порівняння IoT хмарних платформ

Виконав: студент 2 курсу, групи ІПЗ-16-1мд
Чабан І. В.
(ПІБ) (підпис)

спеціальності
121 Інженерія програмного забезпечення
(шифр і назва)

освітньо-професійна програма
121.00.11 Інженерія програмного забезпечення
(шифр і назва)

Керівник Полякова Н. П.
(прізвище та ініціали) (підпис)

Запоріжжя - 2018 року

Запорізька державна інженерна академія
( повне найменування вищого навчального закладу )

Факультет енергетики, електроніки і інформаційних технологій
Кафедра програмного забезпечення автоматизованих систем
Рівень вищої освіти другий (магістерський)
(другий (магістерський) рівень)
Спеціальність 121 Інженерія програмного забезпечення
(шифр і назва)
Спеціалізація _____
(шифр і назва)
Освітньо-професійна програма Інженерія програмного забезпечення
(шифр і назва)

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
В.Г. Вербицький
"04" вересня 2017 року

# З А В Д А Н Н Я
## НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Чабан Ілля Володимирович
(прізвище, ім'я, по батькові)

1. Тема магістерської роботи Дослідження та порівняння IoT хмарних платформ
керівник магістерської роботи доц. Полякова Н. П.,
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "04" вересня 2017 року № 352-01
2. Строк подання студентом магістерської роботи 09.01.2018
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми дипломної роботи;
- дослідження предметної області;
- огляд та аналіз існуючих порівнянь;
- дослідження обраних платформ;
- підключення до платформ за допомогою мови Python;
- порівняльний аналіз досліджених платформ;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
15 слайдів презентації

6. Консультанти розділів магістерської роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата завдання прийняв |
|---|---|---|
| рмоконтроль | Полякова Н.П., доц.каф.ПЗАС | 29.12.17 |
| | | |

7.Дата видачі завдання 04.09.2017

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Прим |
|---|---|---|---|
| 1. | Ознайомлення з предметною областю. | 04.09-08.09.17 | вико |
| 2. | Ознайомлення з технічним завданням. | 09.09-11.09.17 | вико |
| 3. | лювання основної задачі дипломної роботи та узгодження її з науковим керівником. | 12.09-14.09.17 | вико |
| 4. | омлення з існуючими методами та підходами до вирішенням подібних задач. | 15.09-18.09.17 | вико |
| 5. | Аналіз та опис існуючих рішень. | 19.09-27.09.17 | вико |
| 6. | едній аналіз результатів та уточнення задачі у відповідності з дослідженням. | 28.09-10.10.17 | вико |
| 7. | першої платформи. Підключення до платформи за допомогою мови Python. | 11.10-30.10.17 | вико |
| 8. | другої платформи. Підключення до платформи за допомогою мови Python. | 31.10-10.11.17 | вико |
| 9. | третьої платформи. Підключення до платформи за допомогою мови Python. | 11.11-22.11.17 | вико |
| 10. | наліз четвертої платформи. Підключення до платформи за допомогою мови Python. | 23.11-3.12.17 | вико |
| 11. | Виведення критеріїв порівняння. | 4.12-10.12.17 | вико |
| 12. | Виведення порівняльної характеристики. | 11.12-16.12.17 | вико |
| 13. | Доопрацювання отриманих зауважень. | 17.12-20.12.17 | вико |
| 14. | Оформлення звіту та документації. | 21.12-26.12.17 | вико |
| 15. | Представлення результатів науковому керівникові. | 27.12-29.12.17 | вико |

**Студент** _____ **Чабан І.В.**
( підпис ) (прізвище та ініціали)

**Керівник магістерської роботи** _____ **Полякова Н.П**
( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

Сторінок: 106

Рисунків: 21

Таблиць: 5

Джерел: 34

**Мета роботи:** Мета роботи полягає у вивченні основ Інтернету речей, варіантів архітектури та реалізації Інтернету речей, протоколів передачі даних в Інтернеті речей, аналізі наявних порівнянь IoT хмарних платформ. Подальшому дослідженні та підключенні до обраних IoT хмарних платформ, порівнянні їх по основним параметрам та функціоналу.

**Результати:** Досліджено основи Інтернету речей, варіанти архітектури та реалізації Інтернету речей, протоколи передачі даних в Інтернеті речей, проаналізовано наявні порівняння IoT хмарних платформ. Обрані IoT хмарні платформи досліджені та зроблені підключення до них, платформи порівняні по основним параметрам та функціоналу.

**Публікації:** Чабан І.В., магістрант, Полякова Н.П., доцент, науковий керівник. Порівняння IoT хмарних платформ./ І.В. Чабан, Н.П. Полякова. // Матеріали XXII науково-технічної конференції студентів, магістрантів, аспірантів і викладачів ЗДІА. Електроніка, автоматизовані системи та сучасні інформаційні технології. Том III. — Запоріжжя: Видавництво ЗДІА, 2017. — с. 146-148.

*IOT ХМАРНІ ПЛАТФОРМИ, ІНТЕРНЕТ РЕЧЕЙ, ПРОГРАМУВАННЯ, PYTHON, КЕРУВАННЯ ПРИСТРОЯМИ, МОНІТОРИНГ ПРИСТРОЇВ.*

# ABSTRACT

Pages: 106

Figures: 21

Tables: 5

References: 34

**Thesis goal:** The aim of the work is to study basics of the Internet of things, architecture options and the realization of the Internet of things, data transfer protocols in the Internet of things, analysis of available comparisons of IoT cloud platforms. Further research and connection to selected IoT cloud platforms, their comparison by main parameters and functionality.

**Results:** Basics of the Internet of things, variants of architecture and realization of the Internet of things, protocols of data transmission in the Internet of things are researched, available comparisons of IoT cloud platforms are analyzed. Selected IoT cloud platforms are explored and connections to them are made, platforms are compared by main parameters and functionality.

**Publications:** Чабан І.В., магістрант, Полякова Н.П., доцент, науковий керівник. Порівняння IoT хмарних платформ./ І.В. Чабан, Н.П. Полякова. // Матеріали XXII науково-технічної конференції студентів, магістрантів, аспірантів і викладачів ЗДІА. Електроніка, автоматизовані системи та сучасні інформаційні технології. Том III. — Запоріжжя: Видавництво ЗДІА, 2017. — с. 146-148.

*IOT CLOUD PLATFORMS, INTERNET OF THINGS, PROGRAMMING, PYTHON, MANAGEMENT OF DEVICES, MONITORING OF DEVICES.*

# РЕФЕРАТ

Страниц: 106

Рисунков: 21

Таблиц: 5

Источников: 34

**Цель работы:** Цель работы заключается в изучении основ Интернета вещей, вариантов архитектуры и реализации Интернета вещей, протоколов передачи данных в Интернете вещей, анализе имеющихся сравнений IoT облачных платформ. Дальнейшем исследовании и подключении к выбранным IoT облачным платформам, сравнении их по основным параметрам и функционалу.

**Результаты:** Исследовано основы Интернета вещей, варианты архитектуры и реализации Интернета вещей, протоколы передачи данных в Интернете вещей, проанализированы имеющиеся сравнения IoT облачных платформ. Выбранные IoT облачные платформы исследованы и сделаны подключения к ним, платформы сравнены по основным параметрам и функционалу.

**Публикации:** Чабан І.В., магістрант, Полякова Н.П., доцент, науковий керівник. Порівняння IoT хмарних платформ./ І.В. Чабан, Н.П. Полякова. // Матеріали XXII науково-технічної конференції студентів, магістрантів, аспірантів і викладачів ЗДІА. Електроніка, автоматизовані системи та сучасні інформаційні технології. Том ІІІ. — Запоріжжя: Видавництво ЗДІА, 2017. — с. 146-148.

*IOT ОБЛАЧНЫЕ ПЛАТФОРМЫ, ИНТЕРНЕТ ВЕЩЕЙ, ПРОГРАММИРОВАНИЕ, PYTHON, УПРАВЛЕНИЕ УСТРОЙСТВАМИ, МОНИТОРИНГ УСТРОЙСТВ.*

# ЗМІСТ

# ВСТУП

**Актуальність теми**

У зв’язку з бурним розвитком мереж з пакетною комунікацією і перед усім Інтернету, на початку 2000-х років всесвітня телекомунікаційна спільнота спочатку розробила, а потім і приступила до реалізації нової парадигми розвитку комунікацій – мереж наступного покоління NGN (Next Generation Networks). Технології NGN вже пройшли еволюційний шлях розвитку від гнучких комутаторів (Softswitch) до підсистем мультимедійного зв’язку IMS (IP Multimedia Subsystem) та бездротових мереж довготривалої еволюції LTE (Long Term Evolution). При цьому завжди припускалось, що головним користувачем мереж NGN будуть люди і, отже, максимальне число абонентів в таких мережах завжди буде обмежено чисельністю населення планети Земля.

Проте останнім часом значний розвиток отримали методи радіочастотної ідентифікації RFID (Radio Frequency IDentification), бездротові сенсорні мережі WSN (Wireless Sensor Network), комунікації малого радіуса дії NFC (Near Field Communication) і між машинні комунікації M2M (Machine-to-Machine), котрі, інтегруючись з Інтернетом, дозволяють забезпечити простий зв'язок різноманітних технічних пристроїв («речей»), число яких може бути величезним. За розрахунками консалтингового підрозділу Cisco IBSG в проміжку між 2008 та 2009 роками кількість підключених до Інтернету предметів перевищила кількість людей, в 2015 році досягла 25 мільярдів, а к 2020 року досягне 50 мільярдів. Таким чином, в даний час відбувається еволюційний перехід від «Інтернету людей» до «Інтернету речей», IoT (Internet of Things) [1].

Тому сьогодні IoT хмарні платформи як ніколи користуються популярністю. Основними перевагами використання хмарних платформ є швидкість створення нових додатків, гнучкість і масштабованість системи.

З ростом кількості платформ та потребою їх використання, зростає і потреба в їх дослідженні та порівняльній характеристиці можливостей та інструментів, які вони надають.

**Мета і завдання дослідження**

Мета дослідження полягає у вивченні та порівнянні можливостей та інструментів, які надають представлені на ринку IoT хмарні платформи.

**Предмет дослідження**

Інтернет речей та IoT хмарні платформи.

**Методи дослідження**

Для розв’язання поставлених завдань використовуються такі методи дослідження:

- Аналіз літератури про Інтернет речей та хмарні платформи.
- Проведення аналогії з-поміж існуючих порівнянь IoT хмарних платформ.
- Синтез отриманих результатів досліджень.
- Порівняльний аналіз найпопулярніших IoT хмарних платформ.

**Наукова новизна одержаних результатів**

На ринку представлена дуже велика кількість IoT хмарних платформ та все більше розробників починають працювати в цьому руслі, але майже відсутні порівняльні характеристики.

**Практичне значення одержаних результатів**

На основі отриманих результатів можна зробити висновок про те, що дослідження та порівняльна характеристика можуть спростити процес вибору платформи, яка відповідає усім потребам для реалізації програмного рішення IoT.

**Апробація результатів кваліфікаційної роботи магістра**

Основні положення й результати досліджень представлені та обговорені на XXII Науково-технічній конференції студентів, магістрантів, аспірантів і викладачів ЗДІА на секції «Електроніка, автоматизовані системи та сучасні інформаційні технології» 26 квітня 2017 року.

**Публікації**

За результатами досліджень опублікована друкована робота в періодичних виданнях і працях наукових конференцій.

Чабан І.В., магістрант, Полякова Н.П., доцент, науковий керівник. Порівняння IoT хмарних платформ./ І.В. Чабан, Н.П. Полякова. // Матеріали XXII науково-технічної конференції студентів, магістрантів, аспірантів і викладачів ЗДІА. Електроніка, автоматизовані системи та сучасні інформаційні технології. Том III. — Запоріжжя: Видавництво ЗДІА, 2017. — с. 146-148.

**Глосарій**

*Інтернет речей (Internet of Things, IoT)* — концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані датчики, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами, за допомогою використання стандартних протоколів зв'язку. Окрім датчиків, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові чи бездротові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

*HTTP* — протокол передачі даних, що використовується в комп'ютерних мережах. Назва скорочена від Hyper Text Transfer Protocol, протокол передачі гіпер-текстових документів. Основним призначенням протоколу HTTP є передача веб-сторінок (текстових файлів з розміткою HTML), хоча за допомогою нього успішно передаються і інші файли, які пов'язані з веб-сторінками (зображення і застосунки), так і не пов'язані з ними (у цьому HTTP конкурує зі складнішим FTP). HTTP припускає, що клієнтська програма — веб-браузер — здатна відображати гіпертекстові веб-сторінки та файли інших типів у зручній для користувача формі. Для правильного відображення HTTP дозволяє клієнтові дізнатися мову та кодування веб-сторінки й/або запитати версію сторінки в потрібних мові/кодуванні, використовуючи позначення із стандарту MIME. Зазвичай номер порту HTTP — 80, а HTTPS — 443.

*MQTT (Message Queuing Telemetry Transport)* — це простий відкритий протокол, розроблений спеціально для IoT, який застосовується для обміну даними між пристроями. MQTT-мережа включає в себе MQTT-брокера, який служить посередником у взаємодії MQTT-агентів – видавців (Publishers) і підпис-

ників (Subscribers). Видавці публікують інформацію, призначену для підписників.

*Хмарні платформи* — платформи для хмарних обчислень, що представляють собою готове програмне і апаратне забезпечення, які здаються в оренду через Інтернет для розгортання, розробки, тестування застосунків.

*Хмарні платформи IoT* — сукупність взаємодіючих між собою хмарних сервісів (хмара управління), яка забезпечує безпосереднє, без участі людини і проміжних АСУ управління підключеними об'єктами. Ця хмара управління володіє всім необхідним функціоналом (програмними алгоритмами обробки даних і управління) як низових систем управління, так і систем управління рівня підприємства. Тобто IoT-платформа одночасно виконує функції універсального засобу інтеграції та реалізує як завгодно складні і різноманітні алгоритми управління.

*RFID (Radio frequency identification)* — радіочастотна ідентифікація. Радіочастотне розпізнавання здійснюється за допомогою закріплених за об'єктом спеціальних міток, що несуть ідентифікаційну та іншу інформацію. Цей метод вже став основою побудови сучасних безконтактних інформаційних систем, і має стійку назву RFID-технології.

*Python* — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

*Application Programming Interface (API)* — це зазвичай (але не обов'язково) метод абстракції між низькорівневим та високорівневим програмним забезпеченням.

*Back-end* — програмно-адміністративна частина застосунку.

*Callback* — передача виконуваного коду в якості одного з параметрів іншого коду. Callback дозволяє в функції виконувати код, який задається в аргументах під час її виклику. Цей код може бути визначений в інших контекстах програмного коду і бути недоступним для прямого виклику з цієї функції. Деякі алгоритмічні завдання в якості своїх вхідних даних мають не тільки числа або об'єкти, а й дії (алгоритми), які природним чином задаються як зворотні виклики.

*DOM* — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами (як правило, документів XML). З точки зору об'єктно-орієнтованого програмування, DOM визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленням документів у вигляді дерева. Все це призначено для того, аби надати комп'ютерній програмі можливість доступу та динамічної модифікації структури, змісту та оформлення документа.

*Front-end* — це інтерфейс для взаємодії між користувачем і back-end. Front-end та back-end можуть бути розподілені між однією або кількома системами.

*SDK (Software Development Kit)* — набір із засобів розробки, утиліт і документації, який дозволяє програмістам створювати прикладні програми за визначеною технологією або для певної платформи (програмної або програмно-апаратної). Програміст, як правило, отримує SDK безпосередньо від розробника цільової технології або системи. Часто SDK розповсюджується через Інтернет. Багато SDK розповсюджуються безкоштовно для того, щоб заохотити розробників використовувати дану технологію або платформу. Постачальники SDK

інколи підміняють термін Software у словосполуці Software Development Kit на точніше слово. Наприклад «Microsoft» і «Apple» надають Driver Development Kits (DDK) для розробки драйверів пристроїв, а «Palmsource» називає свій інструментарій для розробки «PALMOS Development Kit (PDK)».

*JSON* — це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті та може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією).

*Kanban* — метод управління розробкою, який реалізує принцип «точно в строк» і сприяє рівномірному розподілу навантаження між працівниками. При цьому підході, весь процес розробки прозорий для всіх членів команди. Завдання по мірі надходження заносяться в окремий список, звідки кожен розробник може отримати необхідну задачу.

*MIT* — група ліцензій, розроблених Массачусетським технологічним інститутом для розповсюдження вільного програмного забезпечення.

*SMB* — ринковий сегмент підприємств малого і середнього бізнесу. Маркетинговий термін SMB походить від англійського - small-medium business. Характерною рисою товарів і послуг SMB сегмента є: висока функціональність, універсальність, велика гнучкість в налаштуванні під вимоги бізнесу окремого клієнта, низька вартість володіння, висока масштабованість, простота впровадження та використання і високий рівень зовнішньої сервісної підтримки. Термін SMB в основному застосовується в IT-сфері та індустрії обслуговування діяльності компаній (b2b ринок).

*Бібліотека в програмуванні* (*library*) — збірка підпрограм або об’єктів, що використовуються для розробки програмного забезпечення.

*Веб-застосунок* — розподілений застосунок, в якому клієнтом виступає браузер, а сервером — веб-сервер. Браузер може бути реалізацією так званих

тонких клієнтів — логіка застосунку зосереджується на сервері, а функція браузера полягає переважно у відображенні інформації, завантаженої мережею з сервера, і передачі назад даних користувача.

*Графічний інтерфейс користувача* (*GUI, Graphical user interface*) — тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

*Дедлайн* - це остаточний, затверджений термін здачі проекту. Також цим тепрміном позначають кінцеву дату або час, коли дія мала завершитися, інакше, якщо прострочити і заступити за цю межу, то дія вже не матиме сенсу.

*Клієнт* в інформатиці (*client*) — апаратний або програмний компонент обчислювальної системи, який надсилає запити серверу.

*Людино-машинний інтерфейс* (*Human machine interface*) — широке поняття, що охоплює інженерні рішення, котрі забезпечують взаємодію оператора з керованими ним машинами.

*Модель-вигляд-контролер* або Модель-вид-контролер (*MVC, Model-view-controller*) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

*Облік* — належним чином організована система збору, нагромадження, обробки, групування, узагальнення і реєстрації (фіксації) необхідної інформації або її сукупних даних, що відображають кількісну чи якісну характеристику подій, явищ, фактів, процесів, об'єктів тощо. Є важливою складовою соціального (в т. ч. державного) регулювання суспільних відносин та управління процесами (насамперед економічними) й відповідними об'єктами. Відіграє значну роль при прийнятті правових, економічних, управлінських та інших рішень повноважними органами або компетентними особами.

*Сервер або серверне програмне забезпечення в інформатиці* (*server*) — програмний компонент обчислювальної системи, що виконує сервісні (обслуговуючі) функції за запитом клієнта, надаючи йому доступ до певних ресурсів або послуг.

*Словник* — лексикографічний продукт, який містить впорядкований перелік мовних одиниць (слів, словосполучень тощо) з короткими їх характеристиками або характеристиками позначених ними понять, або з перекладом на іншу мову.

*Фреймворк* (*Framework*) — інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою.

# РОЗДІЛ 1 ДОСЛІДЖЕННЯ ІНТЕРНЕТУ РЕЧЕЙ ТА ХМАРНИХ ПЛАТФОРМ

## 1.1 Інтернет речей

Інтернет речей (Internet of Things, IoT) — концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані датчики, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами, за допомогою використання стандартних протоколів зв'язку, як схематично представлено на рис. 1. Окрім датчиків, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові чи бездротові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

Рис. *1 – Схематичне представлення інтернету речей.*

### 1.1.1 Історія та розвиток Інтернету речей

У зв’язку з бурним розвитком мереж з пакетною комунікацією і перед усім Інтернету, на початку 2000-х років всесвітня телекомунікаційна спільнота спочатку розробила, а потім і приступила до реалізації нової парадигми розвитку комунікацій – мереж наступного покоління NGN (Next Generation Networks). Технології NGN вже пройшли еволюційний шлях розвитку від гнучких комутаторів (Softswitch) до підсистем мультимедійного зв’язку IMS (IP Multimedia Subsystem) та бездротових мереж довготривалої еволюції LTE (Long Term Evolution). При цьому завжди припускалось, що головним користувачем мереж NGN будуть люди і, отже, максимальне число абонентів в таких мережах завжди буде обмежено чисельністю населення планети Земля.

Проте останнім часом значний розвиток отримали методи радіочастотної ідентифікації RFID (Radio Frequency IDentification), бездротові сенсорні мережі WSN (Wireless Sensor Network), комунікації малого радіуса дії NFC (Near Field Communication) і між машинні комунікації M2M (Machine-to-Machine), котрі, інтегруючись з Інтернетом, дозволяють забезпечити простий зв'язок різноманітних технічних пристроїв («речей»), число яких може бути величезним. За розрахунками консалтингового підрозділу Cisco IBSG в проміжку між 2008 та 2009 роками кількість підключених до Інтернету предметів перевищила кількість людей, в 2015 році досягла 25 мільярдів, а к 2020 року досягне 50 мільярдів (рис. 2). Таким чином, в даний час відбувається еволюційний перехід від «Інтернету людей» до «Інтернету речей», IoT (Internet of Things).

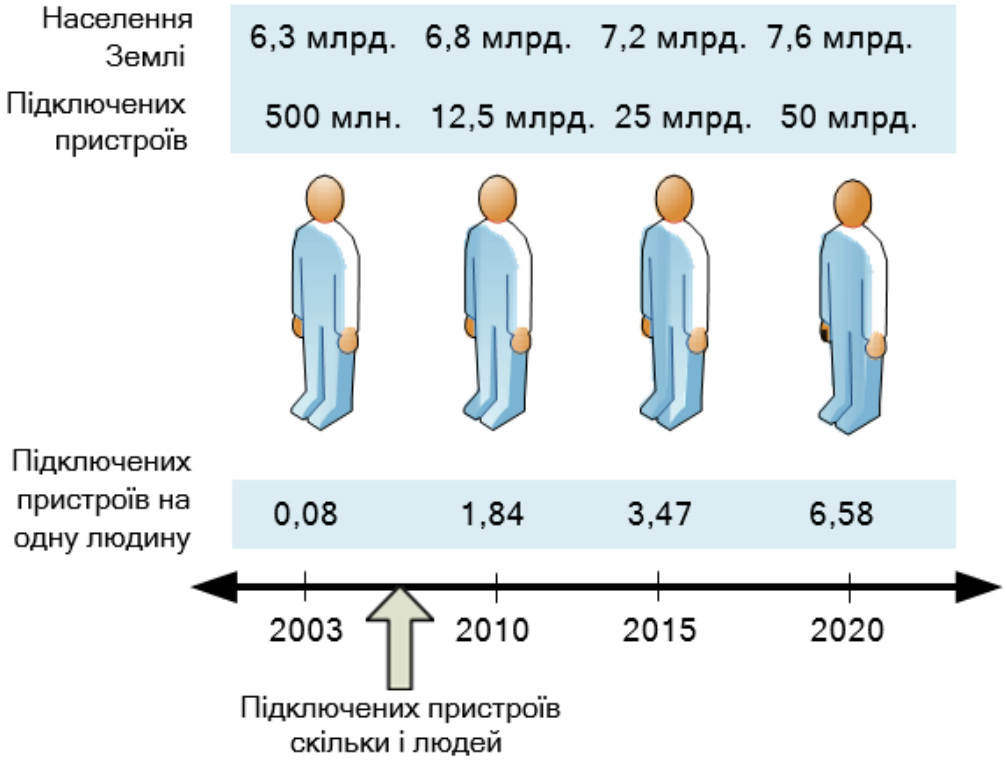



Рис. *2 – Шкала зміни кількості людей і предметів, підключених до Інтернет.*

В загальному випадку під Інтернетом речей розуміється сукупність різноманітних приладів, датчиків, пристроїв, об'єднаних в мережу за допомогою будь-яких доступних каналів зв'язку, які використовують різноманітні протоколи взаємодії між собою і єдиний протокол доступу до глобальної мережі. В ролі глобальної мережі для Інтернету-речей в даний момент використовується мережа Інтернет. Загальним протоколом є IP.

Слід особливо відзначити, що Інтернет речей не виключає участь людини. IoT не повністю автоматизує речі, оскільки він орієнтований на людину і надає їй можливість доступу до цих речей. Але багато речей зможуть вести себе інакше, ніж ми уявляємо собі сьогодні. У IoT кожна річ має свій унікальний ідентифікатор, які спільно утворюють континуум речей, здатних взаємодіяти одна з одною, створюючи тимчасові або постійні мережі. Так речі можуть брати участь в процесі їх переміщення, ділячись інформацією про поточну геолокацію, що дозволяє повністю автоматизувати процес логістики, а маючи вбудований інте-

лект, речі можуть змінювати свої властивості і адаптуватися до навколишнього середовища, в тому числі для зменшення енергоспоживання. Вони можуть виявляти інші, так чи інакше пов'язані з ними речі, і налагоджувати з ними взаємодію. IoT дозволяє створювати комбінацію з інтелектуальних пристроїв, об'єднаних мережами зв'язку, і людей. Спільно вони можуть створювати найрізноманітніші системи, наприклад, для роботи в середовищах, незручних або недоступних для людини (в космосі, на великій глибині, на ядерних установках, в трубопроводах і т. д.). Вважається, що першу в світі Інтернет-річ створив один з батьків протоколу TCP / IP Джон Ромки в 1990 році, коли він підключив до мережі свій тостер. Але тільки в 21 столітті в зв'язку з бурхливим розвитком інформаційно-комунікаційних технологій сформувалася концепція IoT і отримала своє практичне втілення.

Процес розвитку Інтернету речей проілюстрований технологічної дорожньою картою, наведеної на рис. 3. Усе почалося з необхідності оптимізації системи логістики та управління системою постачання підприємств. Друга хвиля інновацій була обумовлена необхідністю скорочення витрат в системах спостереження, безпеки, транспорту та ін. Третя була викликана потребою в геолокаційних сервісах. Четверта хвиля буде обумовлена необхідністю дистанційної присутності людини на місці скоєння подій, які вимагають його уваги, що стане можливим завдяки мініатюрним вбудованим процесорам. А наступним кроком буде можливість створення майбутніх мереж (Future Networks) з комірчастою топологією, що містять мітки, датчики, засоби вимірювання і керуючі пристрої.


Рис. *3 – Технологічна дорожня карта Інтернету речей*

З розвитком Інтернету речей все більше предметів будуть підключатися до глобальної мережі, тим самим створюючи нові можливості в сфері безпеки, аналітики та управління, відкриваючи все нові і більш широкі перспективи і сприяючи підвищенню якості життя населення. Передбачається, що в майбутньому «речі» стануть активними учасниками бізнесу, інформаційних і соціальних процесів, де вони зможуть взаємодіяти і спілкуватися між собою, обмінюючись інформацією про навколишнє середовище, реагуючи і впливаючи на процеси, що відбуваються в навколишньому світі, без втручання людини.

### 1.1.2 Базові принципи IoT

Інтернет речей грунтується на трьох базових принципах. По-перше, всюди поширену комунікаційну інфраструктуру, по-друге, глобальну ідентифікацію кожного об'єкту і, по-третє, можливість кожного об'єкту відправляти і отриму-

вати дані за допомогою персональної мережі або мережі Інтернет, до якої він підключений.

Найбільш важливими відмінностями Інтернету речей від існуючого Інтернету людей є:

- фокус на речах, а не на людині;
- істотно більше число підключених об'єктів;
- істотно менші розміри об'єктів і невисокі швидкості передачі даних;
- фокус на зчитуванні інформації, а не на комунікаціях;
- необхідність створення нової інфраструктури і альтернативних стандартів.

Концепція мереж наступного покоління NGN припускала можливість комунікацій людей (безпосередньо або через комп'ютери) у будь-який час і у будь-якій точці простору. Концепція Інтернету речей включає ще один напрям - комунікація будь-яких пристроїв або речей (рис. 4).


Рис. *4 – Новий напрям комунікацій, що реалізовується Інтернетом речей*

Концепція IoT і термін для неї уперше сформульовані засновником дослідницької групи Auto - ID при Массачусетському технологічному інституті Кевіном Ештоном в 1999 році на презентації для керівництва компанії Procter & Gamble. У презентації розповідалося про те, як усеосяжне впровадження радіочастотних міток RFID зможе видозмінити систему управління логістичними ланцюгами в корпорації.

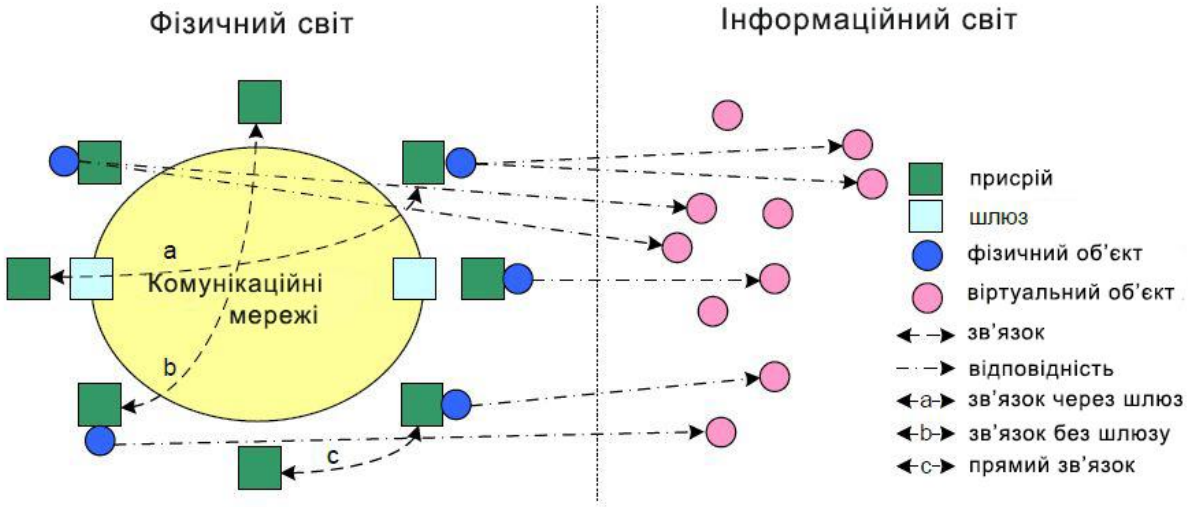
Офіційне визначення Інтернету речей приведене в Рекомендації МСЕ-Т Y.2060 згідно з якої IoT - глобальна інфраструктура інформаційного суспільства, що забезпечує передові послуги за рахунок організації зв'язку між речами (фізичними або віртуальними) на основі існуючих сумісних інформаційних і комунікаційних технологій, що розвиваються.

Під «речами» (things) тут розуміється фізичний об'єкт (фізична річ) або об'єкт віртуального (інформаційного) світу (віртуальна річ, наприклад мультимедійний контент або прикладна програма), які можуть бути ідентифіковані і об'єднані через комунікаційні мережі.

Окрім поняття «річ», МСЕ-Т також використовує поняття «пристрій» (device), під яким розуміється частина устаткування з обов'язковими можливостями з комунікації і необов'язковими можливостями з сенсорингу/зондування, приведення в дію речі, збору, обробці і зберіганню даних. Звідси витікає, що МСЕ-Т більшою мірою приділяє увагу аспектам комунікацій і між з’єднанню, ніж застосункам IoT.

Схема відображення фізичних і віртуальних речей представлена на рис. 5. З рисунку виходить, що віртуальні речі можуть існувати без їх фізичних втілень, тоді як фізичним об'єктам/речам обов'язково відповідає мінімум один віртуальний об'єкт. При цьому провідну роль грають саме пристрої, які можуть збирати різну інформацію і поширювати її по комунікаційних мережах різними способами: через шлюзи і через мережу; без шлюзів, але через мережу; безпосередньо між собою. Рекомендація Y.2060 описує різне поєднання перерахованих

способів з'єднань. Це вказує на те, що MCE-T передбачає використання для IoT безлічі мережевих технологій - глобальних мереж, локальних мереж, безпровідних мереж, що самоорганізуються (ad - hoc) і комірчастих (mesh). Вказані мережі зв'язку переносять дані, зібрані пристроями, до відповідних програмних застосунків, а також передають команди від програмних застосунків до при-




строїв.

Рис. *5 – Схема відображення фізичних і віртуальних речей*

Слід зазначити, що речі і пов'язані з ними пристрої можуть мати повноцінні процесори, для обробки даних у вигляді «системи-на-кристалі», у тому числі з власною операційною системою, блоком сенсорингу/зондування довкілля і блоком комунікації.

Слід розрізняти поняття «Інтернет речей» і «Інтернет-річ». Під Інтернет-річчю розуміється будь-який пристрій, який:

- має доступ до мережі Інтернет з метою передачі або запиту яких-небудь даних;

- має конкретну адресу в глобальній мережі або ідентифікатор, по якому можна здійснити зворотний зв'язок з річчю;

- має інтерфейс для взаємодії з користувачем.

Інтернет-речі мають єдиний протокол взаємодії, згідно з яким будь-який вузол мережі рівноправний в наданні своїх сервісів. На шляху переходу до втілення ідеї Інтернету речей стояла проблема, пов'язана з протоколом IPv4, ресурсом вільних мережевих адрес, який вже практично вичерпав себе. Проте підготовка до повсюдного впровадження версії протоколу IPv6 дозволяє розв'язати цю проблему і наближає ідею Інтернету речей до реальності.

Кожен вузол мережі Інтернет-речей надає свій сервіс, надаючи деяку послугу постачання даних. В той же час вузол такої мережі може приймати команди від будь-якого іншого вузла. Це означає, що усі Інтернет-речі можуть взаємодіяти одне з одним і вирішувати спільні обчислювальні завдання. Інтернет-речі можуть утворювати локальні мережі, об'єднані якою-небудь однією зоною обслуговування або функцією.

### 1.1.3 Стандартизація IoT

Питаннями стандартизації і практичного впровадження окремих складових Інтернету речей (M2M, RFID, всепроникні сенсорні мережі та ін.) займаються багато міжнародних організацій, неурядові асоціації, ал'янси виробників і операторів, партнерські проекти. В цілому для Інтернету речей, як нового напряму розвитку інфокомунікацій, нині визначені найзагальніші концептуальні і архітектурні рішення. Найближчим часом основною проблемою буде гармонізація різних стандартів з метою формування єдиної і несуперечливої нормативної бази для практичної реалізації Інтернету речей.

У рамках діяльності сектору стандартизації телекомунікацій Міжнародного союзу електрозв'язку (МСЕ-Т) є три глобальні ініціативи GSI (Global Standards Initiative). Під глобальною ініціативою розуміється комплекс робіт, що виконуються паралельно різними дослідницькими комісіями МСЕ відповідно до скоординованого плану роботи. Одна з таких ініціатив присвячена стандартизації Інтернету речей - IoT - GSI (Global Standards Initiative on Internet of Things).

Дві інші глобальні ініціативи - зі стандартизації мереж подальших поколінь NGN - GSI і систем телебачення на основі протоколу Інтернет IPTV - GSI - також базуються на використанні IP- технологій, як і IoT - GSI.

IoT - GSI будує свою роботу на основі зусиль MCE -T в таких областях, як мережеві аспекти ідентифікаційних систем (Network Identificator, NID), всепроникні сенсорні мережі (Ubiquitous Sensor Networks, USN), між машинний зв'язок (M2M), WEB речей (WoT) і тому подібне. У рамках серії MCE -T Y.2xxx, яка присвячена мережам наступного покоління NGN, вже затверджені перші рекомендації, присвячені спеціально Інтернету речей : Y.2060 «Огляд Інтернету речей», Y.2063 «Основа WEB речей» і Y.2069 «Терміни і визначення Інтернету речей» та ін.

У Рекомендації Y.2060 приведена еталонна модель IoT, яка дуже схожа на модель NGN і також включає чотири базові горизонтальні рівні (рис. 6):

- рівень застосунків IoT;
- рівень підтримки застосунків і послуг;
- мережевий рівень;
- рівень пристроїв.

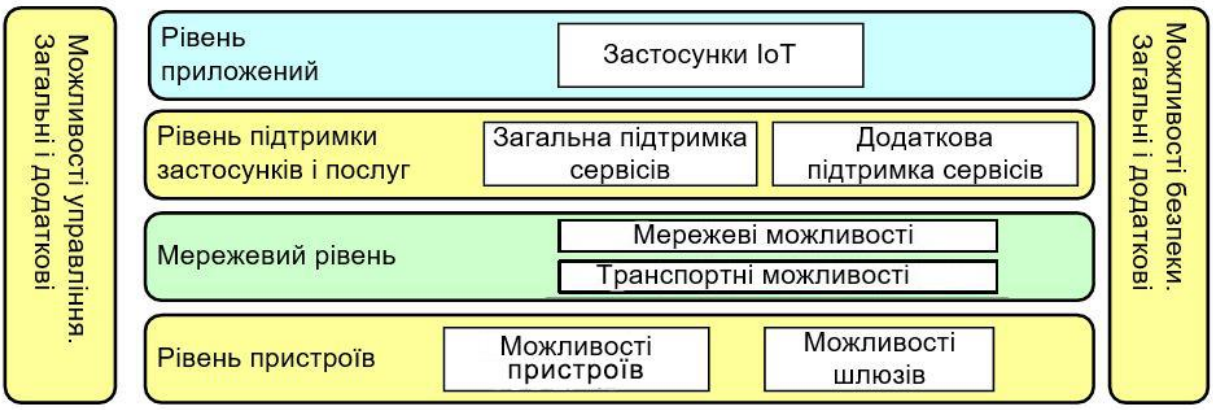



Рис. *6 – Еталонна модель IoT згідно MCE -T Y.2060*

Рівень застосунків IoT в Рекомендації Y.2060 детально не розглядається. Рівень підтримки застосунків і послуг включає загальні можливості для різних

об'єктів IoT з обробки і зберігання даних, а також можливості, необхідні для деяких застосунків IoT або груп таких застосувань. Мережевий рівень включає мережеві можливості (функція управління ресурсами мережі доступу і транспортної мережі, управління мобільністю, функції авторизації, аутентифікації і розрахунків, AAA) і транспортні можливості (забезпечення зв'язності мережі для передачі інформації застосунків і послуг IoT). Нарешті, рівень пристроїв включає можливості пристрою і можливості шлюзу. Можливості пристрою припускають прямий обмін з мережею зв'язку, обмін через шлюз, обмін через безпровідну динамічну ad - hoc мережу, а також тимчасовий зупин і відновлення роботи пристрою для енергозбереження. Можливості шлюзу припускають підтримку безлічі інтерфейсів для пристроїв (шина CAN, ZigBee, Bluetooth, WiFi та ін.) і для мереж доступу/транспортних мереж (2G/3G, LTE, DSL та ін.). Іншою можливістю шлюзу є підтримка конверсії протоколів, у разі, якщо протоколи інтерфейсів пристроїв і мереж відрізняються один від одного.

Існує також два вертикальні рівні - рівень управління і рівень безпеки, що охоплюють усі чотири горизонтальні рівні. Можливості вертикального рівня експлуатаційного управління передбачають управління наслідками відмов, можливостями мережі, конфігурацією, безпекою і даними для білінгу. Основними об'єктами управління є пристрої, локальні мережі і їх топологія, трафік і перевантаження на мережах.

Можливості вертикального рівня безпеки залежать від горизонтального рівня. Для рівня підтримки застосунків і послуг визначені функції AAA, антивірусний захист, тести цілісності даних. Для мережевого рівня - можливості авторизації, аутентифікації, захисту інформації протоколів сигналізації. На рівні пристроїв - можливості авторизації, аутентифікації, контроль доступу і конфіденційність даних.

Основною метою Європейського інтеграційного проекту IoT - A (Internet of Things - Architecture), учасниками якого є різні компанії, є розробка еталонної

архітектурної моделі Інтернету речей з описом основних складових компонентів, яка б дозволила інтегрувати різнорідні технології IoT в єдину взаємозв'язану архітектуру.

Функціональна модель IoT - A (рис. 7) дещо відрізняється від моделі MCE (див. рис. 6), хоча вона також є ієрархічною, але складається вже з семи горизонтальних рівнів, що доповнюються двома вертикальними (управління і безпека), які беруть участь в усіх процесах.

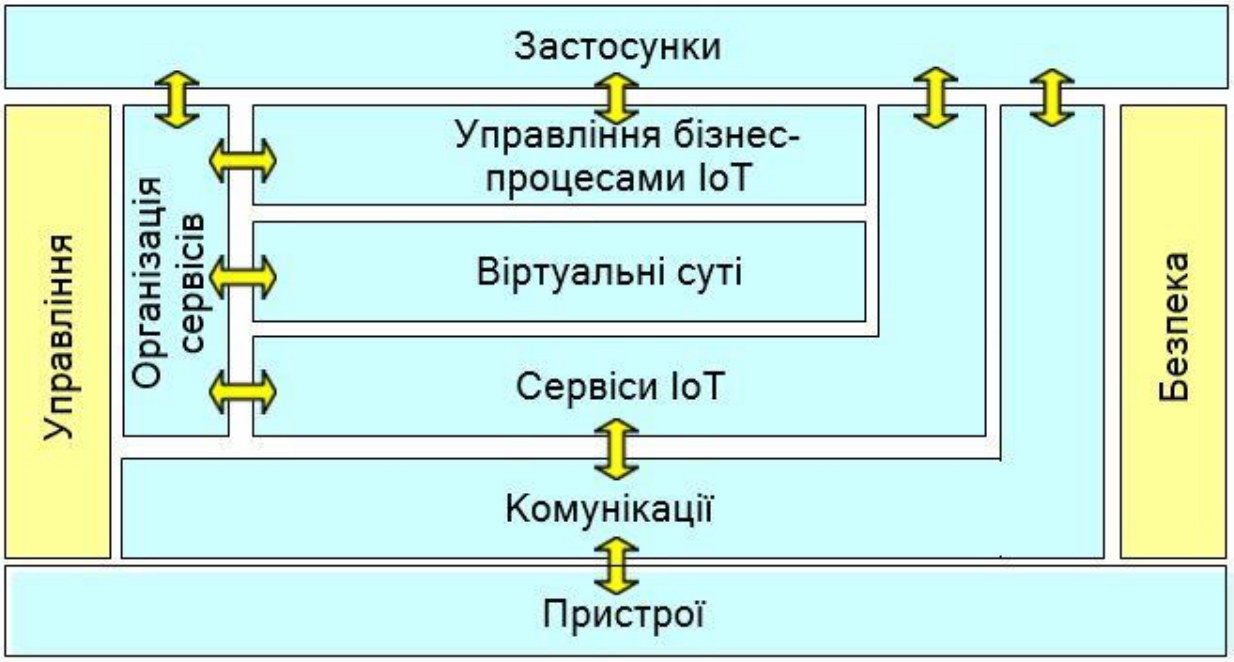



Рис. *7 – Функціональна модель архітектури IoT-A*

Якщо звернутися до технічних особливостей моделі на рис. 7, то можна сказати, що модель передачі даних в Інтернеті речей IoT-A відрізнятиметься від існуючої моделі передачі даних через Інтернет. У моделі архітектури IoT - A фігурують два важливі поняття. Мережа з обмеженнями характеризується відносно низькими швидкостями передачі — менше 1 Мбіт (наприклад, стандарт IEEE 802.15.4) і досить високими затримками. Мережа без обмежень відповідно характеризується високими швидкостями передачі даних (десятки Мбіт/с і бі-

льше) і схожа на існуючу мережу Інтернет. Відмінність цих моделей мереж показана на рис. 8.

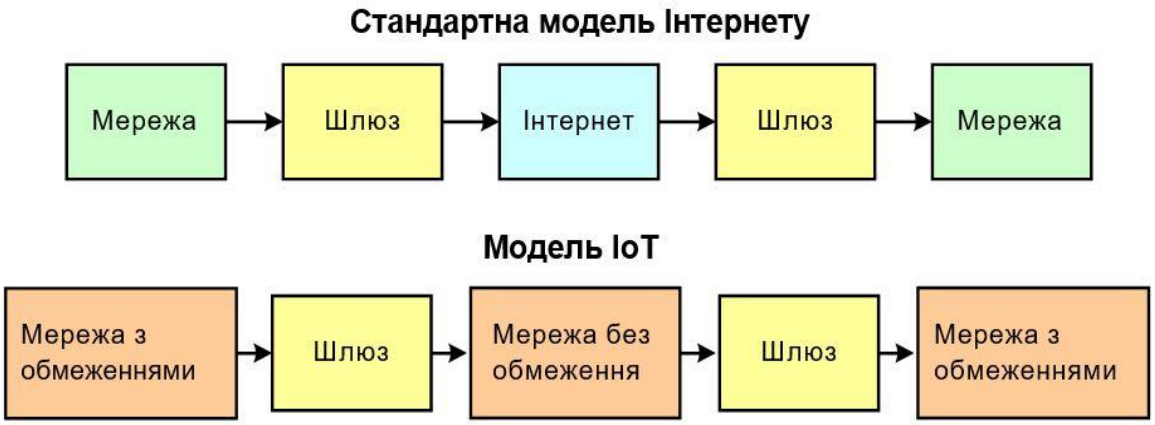



Рис. *8 – Порівняння моделей передачі даних в Інтернеті і в IoT*

### 1.1.4 Архітектура IoT

Інтернет речей концептуально належить до мереж наступного покоління, тому його архітектура багато в чому схожа з відомою чотиришаровою архітектурою NGN. IoT складається з набору різних інфокомунікаційних технологій, що забезпечують функціонування Інтернету речей, і його архітектура показує, як ці технології пов'язані одне з одним. Архітектура IoT включає чотири функціональні рівні (рис. 9), описаних нижче.

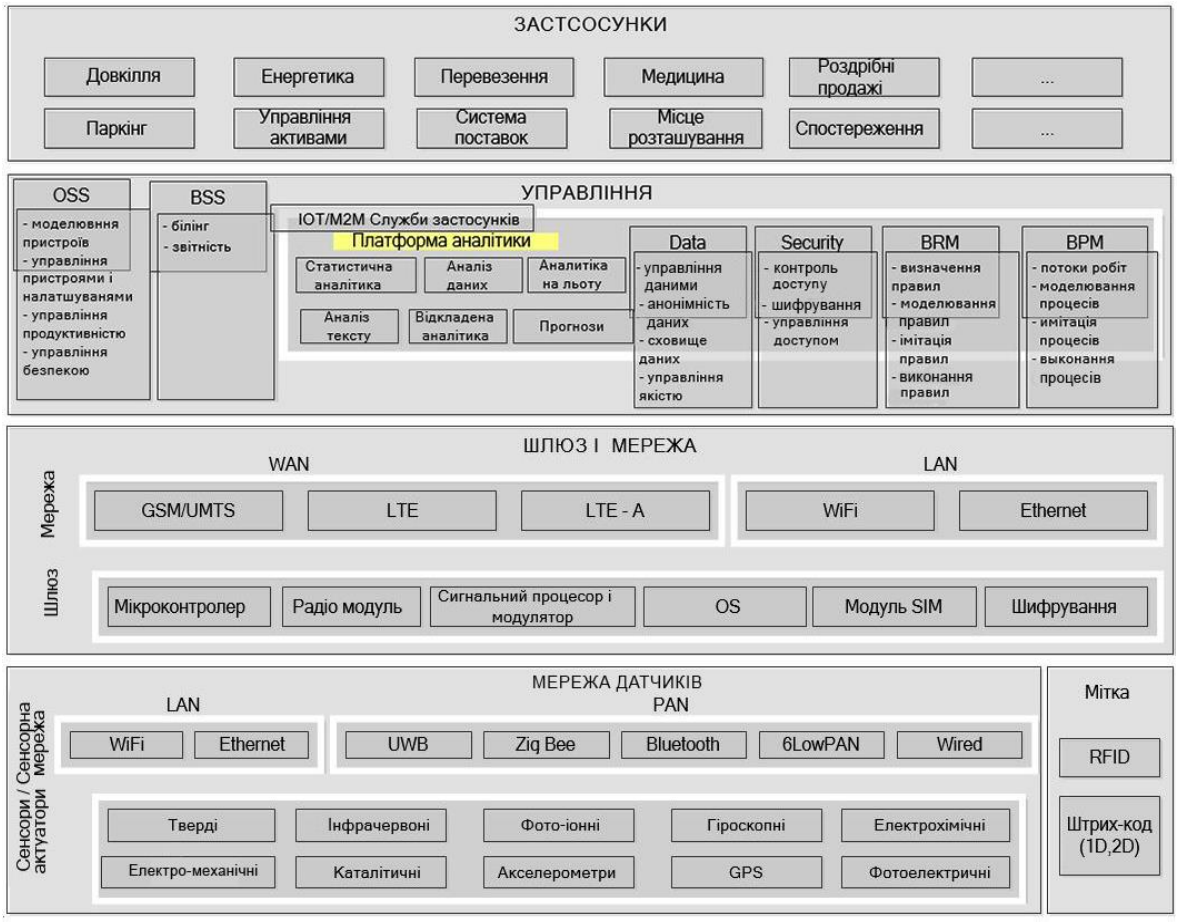



Рис. *9 – Архітектура IoT*

**1. Рівень сенсорів і сенсорних мереж.** Самий нижній рівень архітектури IoT складається з «розумних» (smart) об'єктів, інтегрованих з сенсорами (датчиками). Сенсори реалізують з'єднання фізичного і віртуального (цифрового) світів, забезпечуючи збір і обробку інформації в реальному масштабі часу. Мініатюризація, що призвела до скорочення фізичних розмірів апаратних сенсорів, дозволила інтегрувати їх безпосередньо в об'єкти фізичного світу. Існують різні типи сенсорів для відповідних цілей, наприклад, для виміру температури, тиску, швидкості руху, місця розташування та ін. Сенсори можуть мати невелику пам'ять, даючи можливість записувати деяку кількість результатів вимірів. Сенсор може вимірювати фізичні параметри контрольованого об'єкту/явища і перетворювати їх в сигнал, який може бути прийнятий відповідним пристроєм. Сенсори

класифікуються відповідно до їх призначення, наприклад, сенсори довкілля, сенсори для тіла, сенсори для побутової техніки, сенсори для транспортних засобів і так далі.

Більшість сенсорів вимагають з'єднання з агрегатором сенсорів (шлюзом), які можуть реалізуватися з використанням локальної обчислювальної мережі (LAN, Local Area Network), таких як Ethernet і Wi - Fi або персональної мережі (PAN, Personal Area Network), таких як ZigBee, Bluetooth і ультраширокополосного безпровідного зв'язку на малих відстанях (UWB, Ultra - Wide Band). Для сенсорів, які не вимагають підключення до агрегатору, їх зв'язок з серверами/ застосунками може надаватися з використанням глобальних бездротових мереж WAN, таких як GSM, GPRS і LTE. Сенсори, які характеризуються низьким енергоспоживанням і низькою швидкістю передачі даних, утворюють широко відомі безпровідні сенсорні мережі (WSN, Wireless Sensor Network). WSN набувають все більшої популярності, оскільки вони можуть містити значно більше сенсорів з підтримкою роботи від батарей і охоплюють великі площі.

**2. Рівень шлюзів та мереж.** Великий об'єм даних, що створюються на першому рівні IoT численними мініатюрними сенсорами, вимагає надійної і високопродуктивної дротової або бездротової мережевої інфраструктури в якості транспортного середовища. Існуючі мережі зв'язків, що використовують різні протоколи, можуть бути використані для підтримки міжмашинних комунікацій M2M і їх застосувань. Для реалізації широкого спектру послуг і застосунків в IoT необхідно забезпечити спільну роботу безлічі мереж різних технологій і протоколів доступу в гетерогенній конфігурації. Ці мережі повинні забезпечувати необхідні значення якості передачі інформації, і передусім по затримці, пропускній спроможності і безпеці. Цей рівень складається з конвергентної мережевої інфраструктури, яка утворюється шляхом інтеграції різнорідних мереж в єдину мережеву платформу. Конвергентний абстрактний мережевий рівень в IoT дозволяє через відповідні шлюзи декільком користувачам використовувати

ресурси в одній мережі незалежно і спільно без збитку для конфіденційності, безпеки і продуктивності.

**3. Сервісний рівень.** Сервісний рівень містить набір інформаційних послуг, покликаних автоматизувати технологічні і бізнес операції в IoT: підтримка операційної і бізнес діяльності (OSS/BSS, Operation Support System/Business Support System), різної аналітичної обробки інформації (статистичний, інтелектуальний аналіз даних і текстів, прогностична аналітика та ін.), зберігання даних, забезпечення інформаційної безпеки, управління бізнес-правилами (BRM, Business Rule Management), управління бізнес-процесами (BPM, Business Process Management) та ін.

**4. Рівень застосунків.** На четвертому рівні архітектури IoT існують різні типи застосунків для відповідних промислових секторів і сфер діяльності (енергетика, транспорт, торгівля, медицина, утворення та ін.). Застосунки можуть бути «вертикальними», коли вони є специфічними для конкретної галузі промисловості, а також «горизонтальними», (наприклад, управління автопарком, відстежування активів та ін.), які можуть використовуватися в різних секторах економіки.

### 1.1.5 Веб речей WoT

Складовою частиною Інтернету речей є Веб речей (WEB of Things, WoT), який забезпечує взаємодію різних інтелектуальних об'єктів («речей») з використанням стандартів і механізмів Інтернет, таких як уніфікований ідентифікатор ресурсу URI (Uniform Resource Identifier), протокол передачі гіпертексту HTTP (HyperText Transfer Protocol), стиль побудови архітектури розподіленого застосунка REST (Representational State Transfer) та ін. Фактично WoT передбачає реалізацію концепції IoT на прикладному рівні з використанням вже існуючого архітектурного рішення, орієнтованого на розробку web-застосування. Іншими словами дані з розумних речей або управління ними мають бути доступні через

WWW-сторінки. На рис. 10 показаний приклад, як використовуючи спеціальну сторінку в Інтернеті через браузер, можна рахувати дані з датчика світла у безпровідній сенсорній мережі або змінити колір четвертого індикатора в сенсорі.

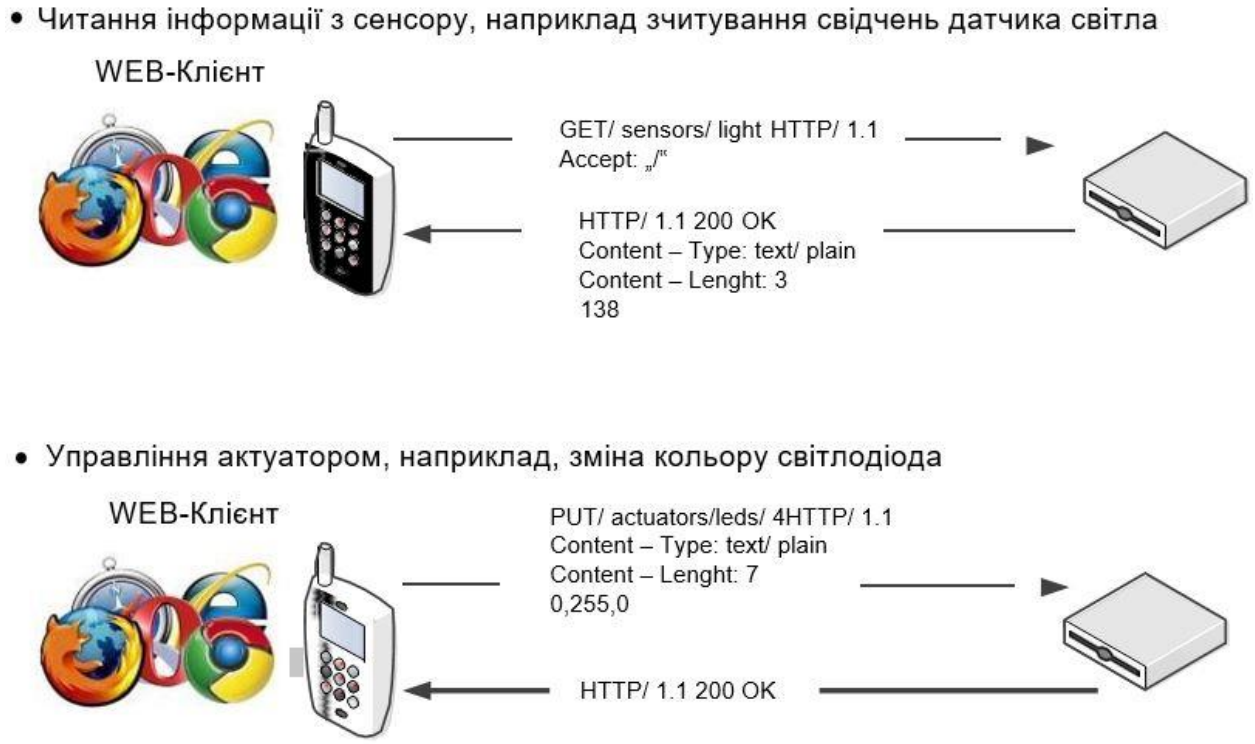



Рис. *10 – Приклади веб-взаємодії з облаштуваннями сенсорної мережі*

Основні властивості WoT:

1. Використовує протокол HTTP як застосунок а не в якості транспортного механізму передачі даних, як він застосовується для традиційних WWW- послуг.

2. Забезпечує синхронну роботу інтелектуальних (смарт) об'єктів через прикладний програмний інтерфейс REST (також відомий як RESTful API) і в цілому відповідає ресурсно-орієнтованій архітектурі ROA (Resource - Oriented Architecture).

3. Надає асинхронний режим роботи інтелектуальних об'єктів з використанням значною мірою стандартних Web- технологій, таких як Atom, що містить формат для опису ресурсів на веб-сайтах і протокол для їх публікації, або Web-механізмів передачі даних, таких як модель роботи веб-застосування Comet, при якій постійне HTTP- з'єднання дозволяє веб-серверу відправляти дані браузеру без додаткового запиту зі сторони браузера.

Ці характеристики WoT забезпечують просту взаємодію інтелектуальних об'єктів через Інтернет, крім того вони реалізують однаковий інтерфейс для доступу і підтримки функціональності смарт-об’єктів.

З концепцією WoT перекликається ідея Семантичного павутиння (Semantic Web) - цей напрям розвитку Всесвітнього павутиння WWW, метою якого є представлення інформації у вигляді, придатному для машинної обробки. Термін «семантичне павутиння» було уперше введене Тімом Бернерсом-Ли (винахідником Всесвітнього павутиння) в травні 2001 року. Концепція семантичного павутиння була прийнята і просувається Консорціумом Всесвітнього павутиння W3C (World Wide Web Consortium).

У звичайному Павутинні, яке ґрунтується на HTML-сторінках, інформація закладена в тексті сторінок і витягається людиною за допомогою браузеру. Семантичне ж павутиння припускає запис інформації у вигляді семантичної мережі за допомогою онтологій. Під онтологією розуміється формальний явний опис понять в даній предметній області (класів). Онтологія разом з набором індивідуальних екземплярів класів утворює базу знань. Таким чином, програма-клієнт може безпосередньо витягати з павутиння факти і робити з них логічні висновки. Семантичне павутиння працює паралельно із звичайним Павутинням і на його основі, використовуючи протокол HTTP і ідентифікатори URI.

Незважаючи на усі переваги, що надаються семантичним павутинням у разі його впровадження, існують певні сумніви в можливості його повної реалізації. Вказуються різні причини, які можуть бути перешкодою до цього, почи-

наючи з людського чинника (люди схильні уникати роботи з підтримки документів з метаданими, відкритими залишаються проблеми істинності метаданих і т. д.). Крім того необхідність опису метаданих так чи інакше призводить до дублювання інформації. Кожен документ має бути створений в двох екземплярах: розміченим для читання людьми, а також в машинно-орієнтованому форматі.

### 1.1.6 MQTT як кращий протокол передачі данних в IoT

MQTT (Message Queuing Telemetry Transport) - це простий відкритий протокол, розроблений спеціально для IoT, який застосовується для обміну даними між пристроями. MQTT-мережа включає в себе MQTT-брокера, який служить посередником у взаємодії MQTT-агентів – видавців (Publishers) і підписників (Subscribers) (рис. 11). Видавці публікують інформацію, призначену для підписників.

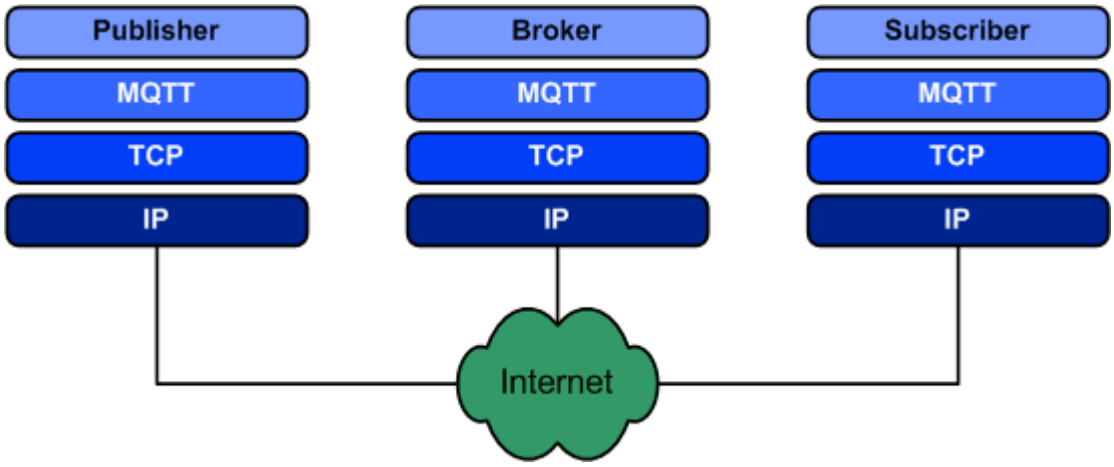



Рис. *11 – Брокер, видавець і підписник в MQTT-мережі*

MQTT розроблений з розрахунку на малопотужні вбудовані пристрої, тому обчислювальні вимоги для його реалізації мінімальні. Додатково до дуже низького навантаження на систему, MQTT відрізняється високою ефективністю зв'язку навіть в мережах з низькою пропускною здатністю. При цьому в структурі переданих з його допомогою даних дуже мало службової інформації, тобто, в порівнянні з багатьма іншими протоколами, наприклад - з HTTP, він майже не

навантажує мережі передачею інформації, яка потрібна лише для функціонування протоколу. За вимірами, зробленими в 3G-мережах, пропускна здатність MQTT в 93 рази вище, ніж протоколу REST (Representational State Transfer), що працює поверх HTTP.

MQTT реалізує модель «видавець - підписник», використовуючи мінімальну кількість методів. Вони служать для вказівки дій, які потрібно виконувати. Ці дії зводяться до взаємодії з брокером і до роботи з темами і повідомленнями. Агенти підключаються до брокера, а потім або публікують теми і повідомлення в них, або підписуються на теми і отримують повідомлення, опубліковані в цих темах. Завершивши роботу, агент відключається від брокера. Ось як виглядають методи MQTT:

Connect - встановити з'єднання з брокером.

Disconnect - розірвати з'єднання з брокером.

Publish - опублікувати тему на брокері.

Subscribe - підписатися на тему на брокері.

Unsubscribe - відписатися від теми на брокері.

На рис. 12 представлена спрощена схема взаємодії між підписником і видавцем з використанням MQTT-брокера.

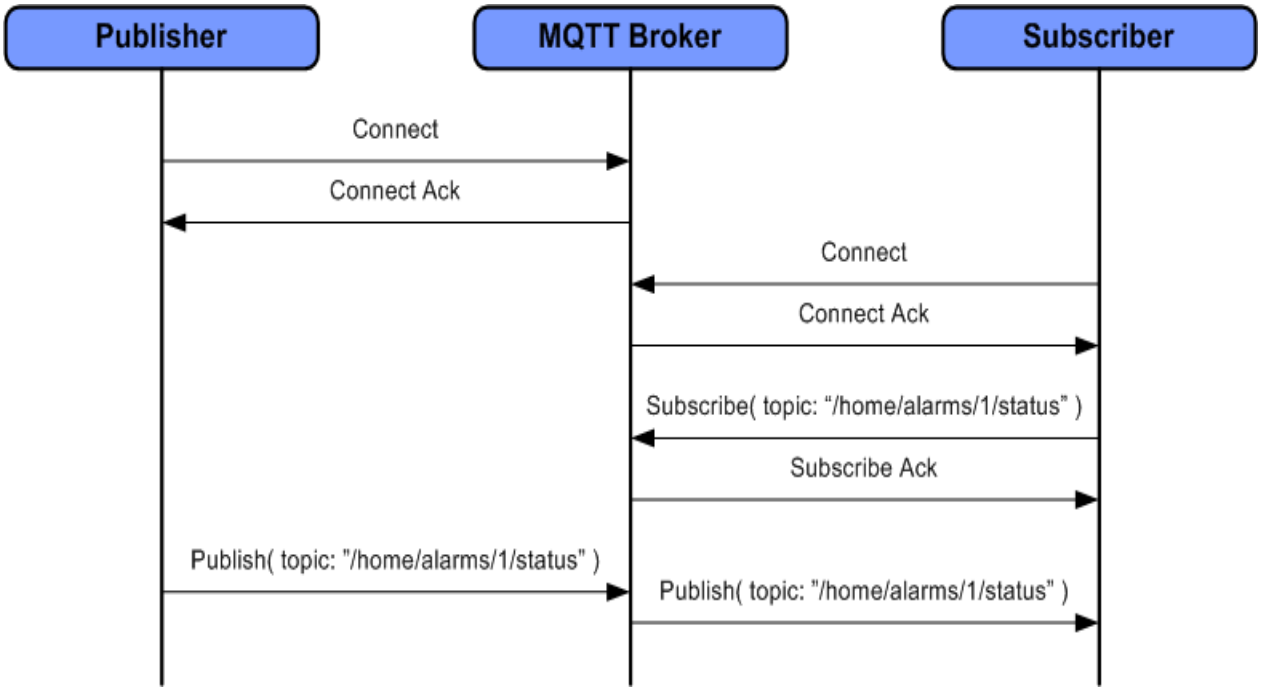



Рис. *12 – Взаємодія підписника і видавця*

Видавець, який є джерелом деяких даних, підключається до брокера. Підписник, споживач інформації, робить те ж саме і підписується на тему, яка представлена тут як «/home/alarms/1/status». В даному прикладі в цій темі публікуються відомості про зміни стану домашньої сигналізації в якійсь «зоні №1». Коли у видавця є нові дані, він звертається до брокера і публікує повідомлення в цій темі. Брокер же, в свою чергу, передає опубліковане повідомлення будь-кому, хто на цю тему підписаний.

Зверніть увагу на ієрархічну структуру імені «/home/alarms/1/status».. Вона схожа на шлях в файлової системі. Це спрощує організацію тем. Крім того, подібні ієрархічні структури іменування ресурсів популярні і в інших архітектурах протоколів, наприклад, в REST.

MQTT, крім того, дозволяє користуватися підстановочними символами, що спрощує процес підписки. Якщо підписник, наприклад, бажає знати про стани всіх датчиків сигналізації, він може підписатися на тему такого вигляду: «/home/alarms/+/status». В результаті, він буде повідомлений про спрацьовуван-

ня сигналізації у всіх зонах, а не тільки в «зоні №1», як у вищенаведеному прикладі. Підписатися на ціле під дерево можна за допомогою шаблону зі знаком «#». Наприклад, підписка на «/home/#» дозволить отримувати повідомлення про все, що відбувається в темах, які перебувають нижче вузла «/home».

**Якість обслуговування.** MQTT підтримує вказівку рівня якості обслуговування (QoS). А саме, існують три таких рівні:

1. QoS 0. Цей рівень задіє стратегію «максимум одноразова доставка повідомлень». Приймач повідомлення не підтверджує їх отримання, відправник, відповідно, передає повідомлення лише раз, не роблячи спроб повторної передачі. Це - метод «відправив і забув».
2. QoS 1. Тут застосовується підхід «мінімум одноразова доставка повідомлень». Гарантується, що приймач отримає повідомлення хоча б один раз. При цьому приймач може отримати одне й те саме повідомлення кілька разів. А відправник буде робити повторні спроби відправки до тих пір, поки не отримає підтвердження про успішну доставку повідомлення.
3. QoS 2. Цьому рівню якості обслуговування відповідає найповільніша процедура доставки повідомлень, але при цьому він - найнадійніший. Його основна особливість - реалізація стратегії «одноразова доставка повідомлень». При його використанні застосовується чотири етапна процедура підтвердження доставки повідомлень.

Вибір конкретного рівня якості обслуговування залежить від особливостей переданих даних і від того, наскільки важливо, щоб вони були доставлені.

## 1.2 Хмарні платформи

Хмарні платформи - платформи для хмарних обчислень, що представляють собою готове програмне і апаратне забезпечення, які здають в оренду через Інтернет для розгортання, розробки, тестування застосунків.

### 1.2.1 Історія створення і розвитку

Ідея створення хмарних платформ була сформована ще за часів появи Інтернету, однак перша хмарна платформа з'явилася тільки в 2006 році під назвою «Amazon Web Services» (хоча надавати доступ до обчислювальних ресурсів через Інтернет компанія почала ще в 2002 році). Саме в 2006 Amazon запропонувала понад 50 різних послуг в 14 географічних регіонах.

Наступною великою платформою стала система Microsoft Azure, що з'явилася в 2010 (зараз посідає друге почесне місце за кількістю користувачів і послуг, що надаються, після Amazon). У 2011 в світі був представлений третій великий гравець на ринку хмарних обчислень - Google Cloud Platform. Через рік після цього почалося активне створення нових платформ і сервісів від різних компаній, і як наслідок зниження цін на оренду платформ. Були створені такі сервіси як: i-Teco OpenStack Cloud, Cloud OS від Azure, Jelastic для PHP і Java додатків та інші.

Сьогодні хмарні платформи як ніколи користуються популярністю. Основними перевагами використання хмарних платформ є швидкість створення нових додатків, гнучкість і масштабованість системи. У топ-10 найбільш відомих і використовуваних хмарних платформ входять: Amazon Web Services, Azure від Microsoft, Google App Engine, Rackspace, Force.com від компанії Salesforce, Intuit Partner Platform, Facebook, IBM Cloud, VMWare vCloud, Sharepoint Online.

### 1.2.2 Технічні характеристики

При використанні хмарних платформ з боку клієнтів необхідна тільки організація робочого місця (комп'ютер / ноутбук / планшет) причому незалежно від місця розташування. Єдиною умовою є наявність виходу в Інтернет. Клієнт (користувач хмарних платформ) займається тільки адмініструванням власних додатків, ПЗ.

За всі інші аспекти, а це: адміністрування баз даних, віртуалізація, мережеве та серверне обладнання, адміністрування операційних систем, обслуговування інфраструктури дата-центру - відповідає постачальник хмарної платформи, часто званий хмарним провайдером. Крім того, окрім платформи для запуску додатків хмарна платформа пропонує ще оренду хмарного сховища і підтримку таких технологій як машинне навчання, штучний інтелект, Big Data.

Для забезпечення безпеки найчастіше використовується окремий VPN і адреса в мережі, власний брандмауер, гнучкі настройки DNS.

Ціна за оренду хмарної платформи зазвичай складається з плати за обчислювальні потужності (включаючи обслуговування обладнання), вартості ліцензії, використовуваного програмного забезпечення/операційної системи/систем віртуалізації і т.д. і плати за послуги хмарного провайдера як постачальника хмарної платформи.

### 1.2.3 Хмарні платформи Інтернету речей

Хмарні платформи IoT являють собою сукупність взаємодіючих між собою хмарних сервісів (хмара управління), яка забезпечує безпосереднє, без участі людини і проміжних АСУ управління підключеними об'єктами. Ця хмара управління володіє всім необхідним функціоналом (програмними алгоритмами обробки даних і управління) як низових систем управління, так і систем управління рівня підприємства. Тобто IoT-платформа одночасно виконує функції уні-

версального засобу інтеграції та реалізує складні і різноманітні алгоритми управління.

Механізм відкритих прикладних інтерфейсів програмування (API) дозволяє підключати до хмари управління будь-які пристрої і будь-які АСУ, не вносячи в них змін, а також обробляти дані, які поставляються в хмару управління, з використанням готових шаблонів, а при їх відсутності - з використанням вбудованих засобів розробки програмних застосунків . Накопичення в платформах IoT історичних даних, що надходять від широкої номенклатури пристроїв і АСУ, і застосування технологій машинного навчання дають можливість автоматизувати процеси вдосконалення алгоритмів, які виконуються хмарою управління, що в принципі неможливо в інформаційно ізольованих АСУ.

# РОЗДІЛ 2 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Була знайдена лише одна детальна порівняльна характеристика IoT хмарних платформ. Але в ній не були представлені деякі платформи, які контролюють найбільший відсоток цього ринку [2], до того ж це не детальне порівняння, а лише класифікація.

## 2.1 Класифікація IoT cloud платформ по галузям застосування та опис

В цій порівняльній характеристиці проаналізовані платформи було розподілено на 10 галузей, які представлені на рис. 13.


Рис. *13 – Галузі застосування IoT cloud платформ.*

Це очевидно, що набагато більше платформ презентовано на ринку, але через технічні особливості та обмеження в часі було обрано 26 з них щоб надати точну картину щодо того як вони працюють, які їх сильні та слабкі сторони, в яких галузях вони доцільні. При вивченні цих IoT платформ, кожна з них була протестована для виявлення слабких та сильних сторін. 10 різних галузей обрано, базуючись на тому, в яких областях більшість IoT cloud платформ зараз прогресує на ринку. Після аналізу було обрано галузі, де ці платформи підходять більше всього, такі як: пристрої, система, різнорідність, дані, розгортання та відстеження. Аналогічно, такі галузі як: аналіз, дослідження та візуалізація обрані, куди можна віднести платформи, які залишилися. Описуються наступні параметри cloud платформ такі як: можливості захоплення даних в реальному часі, візуалізація даних, тип моделі хмари, аналіз даних, налаштування пристроїв, API протоколи та вартість використання, як ключові характеристики. Також наведена Таблиця 1 порівняння IoT clouds, відповідно до того, наскільки вони відповідають вимогам до застосування в розглянутих галузях.

Нижче розглянемо більш детально приклади платформ для кожної галузі застосування.

Таблиця *1*

*Галузі застосування IoT Cloud платформ*

Галузі Застосування IoT Cloud Платформ

| IoT Cloud Платформи | Галузі | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Розробка Застосування | Управління Пристроями | Управління Системою | Управління Неоднорідністю | Управління Даними | Аналіз | Управління Розгортанням | Управління Відслідковуванням | Візуалізація | Дослідження |
| Aer cloud | | | √ | | | √ | | + | | |
| Arkessa | | √ | | | + | | | | | |
| Arrayent Connect | √ | √ | | + | √ | | | | | |
| Axeda | | √ | | | + | | | √ | | |
| Ayla's cloud fabric | | √ | + | | | | | | | |
| Carriots | + | √ | | | | | | √ | | |
| Echelon | √ | √ | | | | | + | | | |
| Etherios | | + | | | | | | √ | | |
| Exosite | | √ | + | | | | | √ | | |
| GroveStreams | √ | | | | | | | √ | | |
| IBM IoT | | √ | | | | | | | | |
| InfoBright | | | | | √ | + | | | | |
| Jasper Control Centre | √ | | | | | + | | √ | | |
| KAA | + | | | | √ | | | | | |
| Microsoft research lab of things | √ | | | | | | | | | |
| Nimbits | | | | | + | √ | | | | |
| Oracle IoT cloud | | | √ | √ | + | | √ | | | |
| OpenRemote | √ | | | + | | | | | | |
| Plotly | | | | | | √ | | √ | | |
| SeeControl IoT | | + | | | | √ | | | | |
| SensorCloud | | + | | | | | | √ | | |
| Temboo | + | | | | | | | | | |
| Thethings.io | √ | | + | | | | | √ | | |
| ThingSpeak | √ | | | | | | | + | | |
| ThingWorkx | √ | | | | + | | | √ | | |
| Xively | √ | + | | | | | | √ | | |

Пояснення: + задовільний, √ відповідний

### 2.1.1 Розробка застосунків

Наступні платформи достатньо ефективні, щоб їх використовувати для розробки додатків, які відсилають команди пристроям на основі отриманих даних.

**KAA** (http://www.kaaproject.org/) — це багатоцільова IoT платформа проміжного рівня з відкритим кодом (Apache ліцензія 2.0) для створення розумних (end-to-end) IoT рішень. Платформа спрощує обмін даними між приєднаними пристроями, механізмами аналізу, механізмами візуалізації та IoT cloud сервісами. зчитування специфікацій пристрою, налаштування, створення зв'язку між пристроями та дозвіл на оновлення прошивки – основна діяльність, яку виконує

KAA. Це надає back end функціональність оперувати великою кількістю IoT рішень включаючи безпеку даних, узгодженість, сумісність та управління даними з набором інструментів розробки програмного забезпечення (SDK) який вбудовують в чіп або пристрій. KAA SDK потребує мало пам'яті, мінімум 10 KB RAM та 40 KB ROM. SDK збирає дані пристроїв, доставляє профілі налаштувань та дозволяє листування між кінцевими точками. Зберігання даних здійснюється двома варіантами NoSQL [3] баз даних такими як: Hadoop [4] та MongoDB.

Плюси: підтримуються застосунки, які використовують NoSQL Big Data бази даних.

Мінуси: підтримується менше модулів апаратного забезпечення.

**Carriots** (https://carriots.com) — це побічний продукт Wairbut (www.wairbut.com), який передбачає допомогу для побудови застосунків для IoT дуже швидко, зберігаючи час та витрати. Платформа як сервіс (PaaS) cloud модель характеризується ключовими технічними характеристиками такими як: управління та контроль віддаленими пристроями, ведення протоколу активності слухачів, запуск користувальницьких аварійних повідомлень та експорт даних. RESTful API дозволяє доставляти отримані дані в  Device, Asset, Group, Service, Project, Stream, Rule, Alarm, Listener, Trigger, Networking, Entity та ConfigTrigger. Електронна пошта, SMS, Twitter, основні класи HTTP використовуються для інформування користувача про поточний стан пристроїв. Дані зберігаються в NoSQL Big Data базі даних, що розширює можливості використання у справжньому сенсі великих даних.

Плюси: підтримується введення в дію застосунків.

Мінуси: менш зручний дизайн.

**Temboo** (https://temboo.com) — це приватна платформа генерації коду застосунків, заснована на хмарі. Це спрощує з'єднання та кодування програмного забезпечення та зменшує витрати на обладнання, що призводить до скорочення часу на розробку та вивід продукту IoT на ринок. Понад 90 вбудованих бібліотек під назвою "Choreos" для сторонніх послуг надають користувачеві можливість користуватися певними службами, що включає в себе погоду Yahoo, Amazon cloud, покупку продуктів через Ebay, керування фотографіями Flickr, Facebook Graph API, аналітика Google, мікроблоги Twitter, Twilio телефонія, PayPal оплата, підтвердження транспортного засобу Uber, трансляція відео в YouTube та багато іншого. Labs [5] — це каталог експериментальних "Choreos", який об'єднує багато "Choreos", створюючи потужні робочі процеси. Це допомагає розробнику візуально налаштовувати апаратне забезпечення для запуску та реагування на онлайн-процеси, зберігати вхідні дані для збереження оперативної пам'яті, віддалено програмувати обладнання, фільтрувати дані відповідно по необхідності.

Плюси: підтримуються за стосунки, основані на "Choreos".

Мінуси: не підходить для ресурсоємних програм.

### 2.1.2 Адміністрування пристроїв

Наступні платформи спеціалізуються на обробці та керуванні пристроями у цифровій формі, наприклад, обробка модулів (Arduino, Raspberry pi і так далі) та аналоговій, наприклад, електричного обладнання (промислових двигунів, роторів та побутових пристроїв).

**SeeControl.** SeeControl **(**http://www.seecontrol.com) є єдиним продуктом змодельованим як сервіс, IoT cloud платформою для підприємств, яка спеціалізується на обміні повідомленнями та управлінням пристроями. Візуалізація даних датчика, аналітика та повний моніторинг робочого потоку виконуються SeeControl. Розширена архітектура push / pull на базі Open API для масово мас-

штабованих продуктів IoT. Моделювання продуктів — це найважливіше функціонал, який моделює фізичні речі, бізнес-групування та події за допомогою Nexus TM Engine щоб перетворити сирі дані у цінну інформацію за допомогою режимів реального часу / пакетної обробки. Інструмент централізованої та складної мережі розподілу вбудованого програмного забезпечення є ще одним життєздатним компонентом, який мотивований нормалізацією даних, що підтримується даними API та пристроями для ERP, CRM та EAM бізнес модулів

Плюси: підтримуються пристрої, основані на Push / Pull запитах.

Мінуси: не достатньо хороша візуалізація.

**SensorCloud.** SensorCloud (http://www.sensorcloud.com) — це приватна хмара IoT, яка надає платформу як сервіс для отримання, візуалізації, моніторингу та аналізу даних, отриманих від дротових або бездротових датчиків Lord Microstrain. В основному SensorCloud — відмінний інструмент, що використовує потужні обчислювальні засоби, такі як масштабованість даних, швидка візуалізація та аналіз користувацьких програм. Аналітика MathEngine® дозволяє розробникам виконувати складні математичні операції з даними. Функції FastGraph та LiveConnect допомагають розробникам запускати графічні функції на збережених даних, завантажених вручну (CSV) або автоматично (OpenData API). SensorCloud надає API RESTful, який дозволяє будь-якому пристрою або додатку завантажувати дані у свою безпечну хмару, яка в даний час побудована на вершині Amazon Web Services (AWS).

Плюси: можна керувати великою базою.

Мінуси: пристрої із відкритим вихідним кодом важко обслуговувати.

**Etherios.** Etherios (http://www.etherios.com) підтримує комплексний набір продуктів та послуг для підприємств. Хмара пристроїв розроблена на основі моделі PaaS, щоб дозволити користувачам груп підключати будь-який продукт і отримувати видимість в реальному часі у своїх активах. Соціальна машина - це ще один інструмент на основі хмар, який забезпечує рішення SaaS при інтегра-

ції машинних даних з відповідним Salesforce.com об’єктом для перетворення його в більш потужний CRM. Etherios об'єднує сучасні підприємства через тисячі бездротових з’єднань і бездротових рішень, призначених для конкретної мети. Він надає індивідуальні рішення для будь-якого пристрою за допомогою Cloud Connector. Крім того, управляє, відслідковує та контролює всі підключені пристрої з одного інтерфейсу в режимі реального часу. Etherios не стягує плату, щоб розробники могли використовувати до 5 пристроїв протягом 30 днів.

Плюси: включені спеціалізовані хмари для пристроїв та сторонні програми.

Мінуси: розробники обмежені обраними пристроями.

**Xively.** Xively (https://xively.com) — це IoT cloud сервіс для підприємств на базі технології Gravity Cloud. Ця платформа, що належить LogMeIn, допомагає компаніям керувати підключеним бізнес продуктом, вирішуючи ряд практичних потреб таких як масштабування, безпека та надійний зв'язок. Вона також забезпечує правильні послуги обробки бізнес-даних у відношенні своїх клієнтів, партнерів та постачальників, що підтримують IoT за допомогою гнучких API-з'єднань. Xively використовує нововведену платформу IoT як службу (IoTaaS), побудовану на її еластичній публічній хмарі. Еластична масштабованість хмар забезпечує інтуїтивно зрозумілі можливості керування життєвим циклом пристроїв, що має на увазі серії дій відповідно часу. Крім того, за допомогою Xively полегшується архівація даних, запуск за станом, активація для надання пристрою в режимі реального часу, керування повідомленнями та маршрутизація. Щоб полегшити контроль над пристроями, Xively має консоль керування робочим середовищем розробника та консоль керування пристроєм, якою може керувати навіть початківець. Він також здатний підтримувати мільйони пристроїв за допомогою API RESTful. Використання форматів даних JSON, XML та CSV покращило свою ефективність з точки зору пристрою асоціативно, що можна відслідкувати за допомогою попередньо оцінених клієнтських бібліотек, побу-

дованих на вершині iOS, Android, JavaScript, а також серверних бібліотек, призначених для додатків на базі високорівневих та веб мов, таких як Ruby, Python і Java.

Плюси: легко інтегруватися з пристроями.

Мінуси: послуги оповіщення мінімально присутні.

### 2.1.3 Управління системою

Представлені платформи в цій галузі відповідають роботам, пов’язаним із управлінням системою, що є невід’ємною складовою формування всієї інфраструктури.

**Ayla's IoT cloud fabric.** Ayla IoT Fabric (https://www.aylanetworks.com) — платформа для підприємств, яка використовує модель PaaS, просте та ефективне рішення для підключення будь-якого пристрою до Інтернету. Ayla Networks надають потужні програмні агенти вбудовані в під’єднані пристрої та застосунки для мобільних пристроїв для підтримки end-to-end. Ayla's Agile Mobile Application Platform (AMAP) побудована на мобільних бібліотеках, які надають оптимізовану мобільну програму для користувачів iOS та Android. Інтеграція даних IoT, візуалізація, аналіз поведінки користувачів та RESTful API надають багато можливостей на інформаційній панелі користувачів.

Плюси:  легка розробка мобільних додатків.

Мінуси: не підходить для розробників малого масштабу.

**Thethings.io.**thethings.io (https://thethings.io) забезпечує повне рішення для зворотного зв'язку для маркерів IoT та розробників IoT застосунків за допомогою простого та гнучкого API. thethings.io — агностичний апарат і дозволяє підключати будь-який пристрій, здатний використовувати протоколи HTTP, Websocket, MQTT або CoAP. Робота в реальному часі, основана на правилах

може легко відстежуватись при розробці end-to-end з'єднання між пристроями, використовуючи керування пристроєм, відстеження та аналітичну підтримку. Об'єкти зберігання даних в режимі реального часу також забезпечують сумісний доступ пристроїв для швидкого та дешевого розроблення продукту. Розгортання хмар в даний час здійснюється на вершині AWS.

Плюси: агностика пристроїв.

Мінуси: бракує самодостатності, залежить від сторонніх веб-служб.

**Exosite.** Exosite (https://exosite.com) — це модульна корпоративна платформа IoT, яка допомагає виробникам випускати на ринок споріднені продукти. Підтримка cloud платформи на базі IoT Software as a Service (SaaS) забезпечує користувачам візуалізацію та аналіз даних в режимі реального часу. Це серверна система, на якій ввімкнено API веб-сервісів, побудований в інфраструктурній структурі, легкий та гнучкий зворотній зв'язок, за допомогою UDP, HTTP і JSON RPC. Різноманітні набори розвитку підтримуються для розробки та розгортання рішень IoT. Прототипні дошки Arduino, Microchip, TI та Renesas добре обмінюються з платформою Exosite. Пристрій клієнтського програмного забезпечення разом із портами загального доступу полегшує приєднані пристроїв для підключення та передачі контекстів через CoAP, а також API інтерфейсів UDP. Управління пристроями, візуалізація даних, моделювання пристроїв, забезпечення поля, інтерфейс для читання та запису даних з клієнта чату, створення віджетів інформаційних панелей порталів - це численні завдання, які контролюються Exosite.

Плюси: розробка системи проста.

Мінуси: відсутнє забезпечення обробки Big data.

### 2.1.4 Управління різнорідностями

Ця галузь застосування використовує IoT cloud платформи, які надають з’єднання та зв'язок між різнорідними системами різних виробників.

**Arrayent Connect TM.** Arrayent (http://www.arrayent.com) — IoT платформа, яка дає змогу з’єднати основні різнорідні бренди такі як Whirlpool, Maytag, та First Alert. Arrayent Connect Cloud являє собою оперативну IoT-систему, яка використовує модель програмного забезпечення, як сервіс (SaaS), допомагає розміщувати всі віртуалізовані пристрої за допомогою оновлень прошивки через Ovi (OTA) з низькою швидкістю затримки даних.

Плюси: гнучка для використання.

Мінуси: служби на основі тригера відстають.

**Open remote.** OpenRemote (http://www.openremote.com) будучи IoT рішенням проміжного рівня з відкритим кодом, дозволяє користувачеві інтегрувати будь-який пристрій, використовуючи доступні ресурси, такі як iOS, Android або веб-браузери. OpenRemote безкоштовний для звичайних дослідників, а з розробників, які розробляють застосунки на продаж, стягується €150–375 за застосунок.

Плюси: підтримується відкритий cloud сервіс.

Мінуси: занадто дорогий для розробників.

### 2.1.5 Управління даними

Наступні IoT Cloud платформи успішно оперують, коли задача управління даними дуже важлива для підтримки нижче розташованих систем. Нижче описані платформи здатні поширювати операції та дії, пов'язані з даними, щоб забезпечити користувачам глибокі знання про поточний сценарій.

**Arkessa.** Arkessa (http://www.arkessa.com) надає з’єднання, моніторинг, контроль та управління між IoT пристроями та підприємствами

Плюси: інтегрування машинних потоків даних з системами аналізу.

Мінуси: програми візуалізації недостатньо відповідні.

**Axeda.** Axeda (http://www.axeda.com) — це платформа, основана на IoT cloud, призначена для управління з’єднаними продуктами та машинами, реалізуя IoT та M2M додатки.

Плюси: керування даними на основі M2M.

Мінуси: бракує самодостатності, залежить від сторонніх веб-служб.

**Oracle IoT cloud.** Oracle IoT (https://cloud.oracle.com/iot) являє собою комбінацію з чотирьох ключових параметрів, таких як: Open — що з'єднує будь-який тип пристрою від датчиків до шлюзів; Insight — що збирає бізнес-вартість значення IoT даних; Secure — що забезпечує нормалізовану безпеку для всіх типів пристроїв, даних та гетерогенних з'єднань; та Accelerate — що допомагає прискорити реалізацію ідеї з мінімальним ризиком.

Плюси: підтримка бази даних.

Мінуси: бракує зв'язку з пристроями з відкритим вихідним кодом.

**Nimbits.** Nimbits (http://www.nimbits.com) гібридний cloud сервер. Підтримує Arduino [6] платформи.

Плюси: легко освоюється розробниками.

Мінуси: обробка запитів у реальному часі.

**ThingWorx.** ThingWorx (https://thingworx.com) популярне, управляюче даними рішення, утворююче приватну cloud платформу.

Плюси: легко створювати додатки, насичені інформацією.

Мінуси: можна прикріпити обмежену кількість пристроїв.

### 2.1.6 Інструменти для статистичного аналізу

Наведені тут IoT Cloud платформи спрямовані на проведення статистичної аналітики, надаючи аналітичні інструменти.

**InfoBright.** InfoBright (https://www.infobright.com/index.php/internet-of-things) залучає підприємства, надаючи свою аналітичну Knowledge Grid архітектуру (основана на IoT аналітична платформа).

Плюси: Knowledge Grid архітектура та обробка big data.

Мінуси: відсутні статистичні послуги.

**Jasper Control Center.** Jasper Control Center (https://www.jasper.com) — це широко гнучка Jasper Control Board платформа, яка може бути налаштована щоб підходити до конкретних операційних потреб, бізнес моделей та потреб серед усіх індустрій.

Плюси: включені шаблони поведінки на основі правил.

Мінуси: не підходить для автоматизації послуг.

### 2.1.7 Управління розгортанням

Включає в себе планування, проектування, побудову, тестування та впровадження нових програмних та апаратних компонентів у live середовищі. Важливо підтримувати цілісність live середовища шляхом правильного розгортання випусків програмного забезпечення [7].

**Echelon.** Echelon (http://www.iiot.echelon.com) — це нова cloud платформа основана на Industrial Internet of Things (IIoT) [8] з усім набором ресурсів, стеком протоколів, апаратними модулями, інтерфейсами комунікації та управлінням пакетами програмного забезпечення для розробки пристроїв в peer-to-peer застосунках.

Плюси: промислова перспектива.

Мінуси: нестача сценарію розробки для початківців.

### 2.1.8 Управління моніторингом

Наступні IoT clouds відіграють важливу роль у запобіганні мережевих катастроф та збереженні мережі. Можливість забезпечення загальної бездоганної інтеграції з системою призводить до ненав'язливої форми онлайн моніторингу.

**AerCloud.** AerCloud (http://www.aeris.com) — це cloud платформа для збору, управління та аналізу даних датчиків для IoT та M2M застосунків. AerCloud — розробка Aeris, надана як Platform as a Service (PaaS).

Плюси: масштабовані M2M сервіси.

Мінуси: не підходить для розробників.

**ThingSpeak.** ThingSpeak [9] (https://thingspeak.com) — це відкрита платформа IoT даних основана на публічній cloud технології. ThingSpeak надає можливість збору, аналізу та активації в реальному часі за допомогою Open API.

Плюси: надання публічної cloud з активаційними об’єктами.

Мінуси: можливість підключення одночасно малої кількості пристроїв.

### 2.1.9 Візуалізація

Наступні IoT clouds є надзвичайно корисними за рахунок великої кількість повноцінних графічних інструментів візуалізації для виведення діяльності систем на екрані в графічному форматі.

**Plotly.** Plotly (https://plot.ly) — це популярний провайдер публічного cloud сервіса візуалізації даних. Plotly надає професійні та комерційні сховища даних, сервіси візуалізації та аналізу для звичайних та IoT застосунків.

Плюси: найкращі інструменти візуалізації, які підтримують IoT.

Мінуси: обмежений обсяг сховища.

**GroveStreams.** GroveStreams (https://thingworx.com) — це популярна публічна cloud хмарна платформа для візуалізації даних. Аналіз даних пристрою відбувається за допомогою RESTful API.

Плюси: включений безперервний моніторинг подій.

Мінуси: відсутні статистичні послуги.

### 2.1.9 Дослідження

Представлені тут IoT clouds допомагають дослідникам оцінювати реалізацію, надаючи можливість підключення пристроїв та отримання від них даних, таким чином знижуючи накладні витрати на фактичну реалізацію. Це мінімізує ризик втрат грошей та час випуску продукту на ринок.

**Microsoft research Lab of Things.** Lab of Things (http://www.lab-of-things.com) — це відкрита IoT платформа, яку розробила компанія Microsoft для експериментальних досліджень, здебільшого для академічних установ. Як правило, вона призначена для взаємодії між пристроями, реалізації різних сценаріїв застосування, розгортання тематичних досліджень, моніторингу приладів та обміну даними за допомогою HomeOS4. Програми можуть потрапляти в такі сфери, як охорона здоров'я, енергоменеджмент, автоматизація дому та багато іншого. Лабораторія речей складається з компонента клієнтської сторони — HomeOS [10] та хмарних сервісів, що розгортаються в Windows Azure. Експериментальні програми, такі як сповіщення про тривогу, виявлення руху через

камеру, збір зразків даних від датчиків та керування приводами з підтримкою Z-Wave [11], знаходяться поверх шару додатків. Публічна система, яка взаємодіє з cloud, розгортається на ПК на базі Windows (HomeHub).

Плюси: підходить для домашньої автоматизації.

Мінуси: бракує API підтримуючих IoT.

**IBM IoT.** IBM IoT cloud platform (https://internetofthings.ibmcloud.com) — це організована архітектура, розроблена щоб надати безпеку та легкість підключення пристроїв, починаючи з чипів до застосунків та складних індустріальних рішень. IDentity as a Service [12] (IDaaS) є основою цієї cloud.

Плюси: надана ідентифікація пристроїв.

Мінуси: складний для протипування застосунків.

### 2.2 Аналіз класифікації галузей застосування IoT cloud платформ

Огляд галузей застосування IoT cloud платформ було проведено на виборці 26 різних IoT clouds, які підпадають під 10 різних галузей застосування. Виявлено, що для таких галузей як управління різнорідними пристроями, аналіз, візуалізація та дослідження, прослідковується нестача в загальному відсотку наявних IoT cloud платформ. В тренді розвитку знаходяться такі галузі застосування: управління пристроями, управління даними та управління застосунками. Також можна побачити, що деякі IoT cloud спроможні задовольняти декілька галузей. IoT clouds орієнтовані на галузь досліджень дуже малочисельні, але дуже необхідні для дослідників IoT. На жаль, не одна з 26 досліджених IoT cloud платформ, не ідеальна в усіх аспектах. Деяким не вистачає можливості візуалізації, іншим — відкритих IoT APIs. Короткий опис плюсів та мінусів повинен допомогти обрати відповідну IoT cloud платформу.

Рис. 14 (дивіться зліва) показує результати отримані з Таблиці 1 де зображено задовільні та відповідні рішення для галузей. З 26 IoT cloud платформ, Платформи галузі управління даними мають найбільший відсоток на ринку — 19.2%. Галузі управління пристроями та застосунками займають по 11.5%. Галузі управління різнорідністю, аналізу, моніторингу, візуалізації та дослідження мають тільки по 7.6%. Галузь управління найменш пріоритетна на даний час — 3.8%.

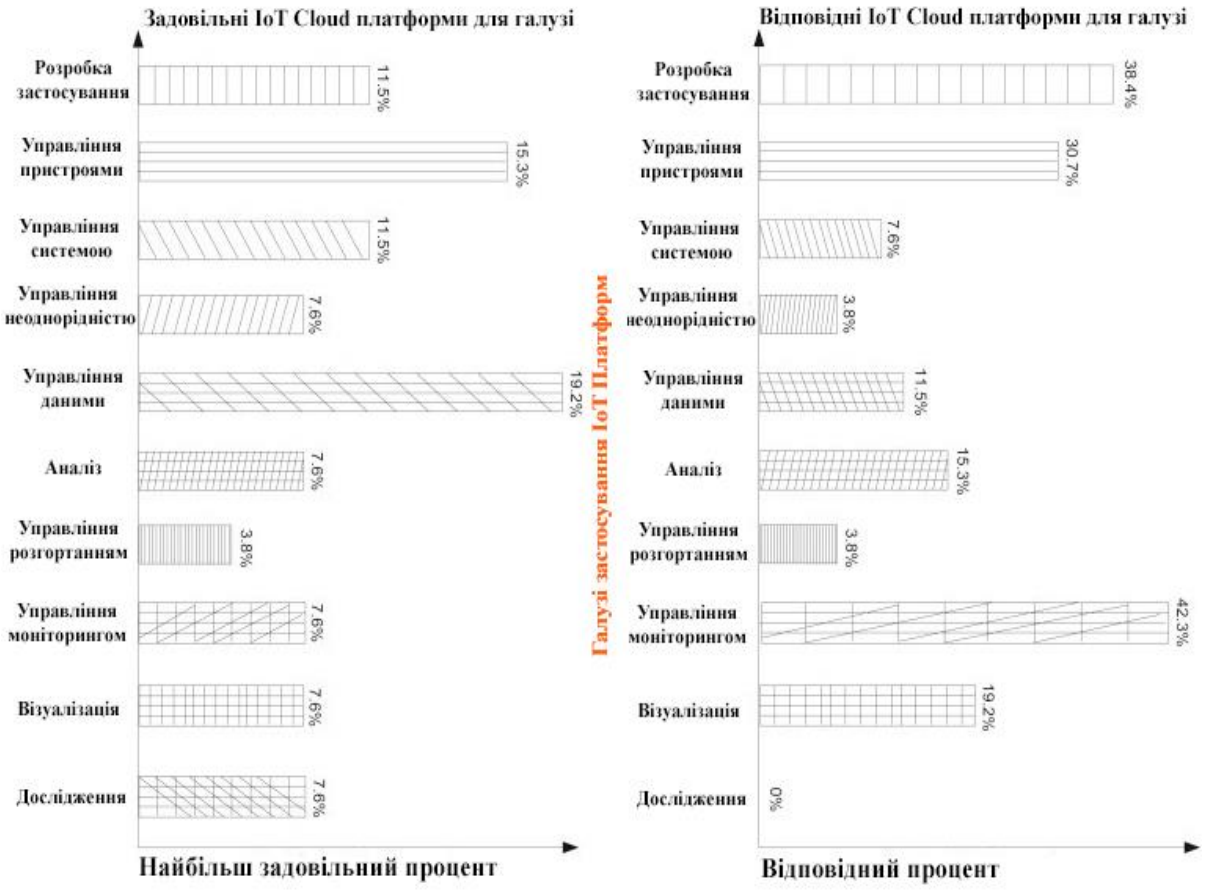



Рис. *14 – Гістограма найбільш задовільних та віповідних IoT Cloud платформ для галузей застосування Таблиця 1.*

## 2.3 Висновки з розділу

1. Виведена класифікація спростить розробникам та дослідникам процес вибору IoT Cloud платформ, відповідних до їх галузі.
2. Характеристика самих платформ та виведені плюси та мінуси допоможуть обрати конкретну IoT Cloud платформу із галузі.

# РОЗДІЛ 3 ПРАКТИЧНІ ДОСЛІДЖЕННЯ ІОТ ХМАРНИХ ПЛАТФОРМ

Для практичних досліджень було обрано такі 4 IoT хмарні платформи: IBM Watson, SAP, AWS IoT Core, Azure IoT Suite. Та подальшого їх порівняння за такими критеріями:

- Ідентифікатор пристрою: керування унікальними ідентифікаторами пристроїв.
- Аутентифікація пристрою: можливість під’єднання тільки за наявності API ключів.
- Управління пристроями: керування метаданими пристроїв та виконання операцій, таких як перезавантаження пристроїв та оновлення програмного забезпечення.
- Команди та керування: щоб пристрій виконував дію, надсилайте повідомлення на пристрій із хмари.
- Правила та дії: можливість робити дії на основі правил, які застосовуються до отриманих платформою даних від пристрою.
- Візуалізація даних: Інструменти для візуалізації даних такі як виведення основних характеристик пристрою, різних типів графіків та діаграм на основі даних.
- Події пристрою: події, які може викликати пристрій, такі як підключення пристрою та відключення.
- Інтелектуальна аналітика: аналіз даних пристрою на хмарі дає змогу прогнозувати, коли мають відбуватися певні дії. Наприклад, аналізуючи телеметричний двигун повітряного судна, потрібно визначити, коли потрібно забезпечити технічне обслуговування двигуна.

## 3.1 IBM Watson IoT хмарна платформа

Безпечна, розумна, масштабована платформа як центр IoT. Платформа Watson IoT дає вам змогу повністю керувати вашим IoT, а також робити кращі, ділові рішення [30]. Надає такі можливості:

**Управління пристроєм.** Використовуючи послугу керування пристроєм, ви можете посилати команди для виконання дії пристрою, такі як перезавантаження або оновлення вбудованого програмного забезпечення, отримання діагностики пристрою та метаданих, а також додавання та видалення пристроїв.

**Масштабований зв'язок з хорошим відкликом.** Використовується промисловий стандарт MQTT (OASIS затверджений) для підключення пристроїв та програм. MQTT призначений для ефективного обміну даними з пристроями у режимі реального часу.

**Безпечне спілкування.** Надає можливість безпечно отримувати дані і відправляти команди на свої пристрої. Робиться це за допомогою MQTT з TLS, щоб захистити все спілкування між вашими пристроями та сервісом.

**Зберігання та доступ до даних.** Окрім доступу до даних у режимі реального часу, що надходять із ваших пристроїв, ви можете вибрати дані для зберігання на певний період, що дозволить вам мати доступ до збережених даних та даних у реальному часі з ваших пристроїв.

**Потужна веб-панель інструментів.** Надає гнучку, масштабовану і просту у використанні потужну веб-панель інструментів.

### 3.1.1 Схема роботи платформи

Схема роботи платформи IBM Watson IoT представлена на рис. 15

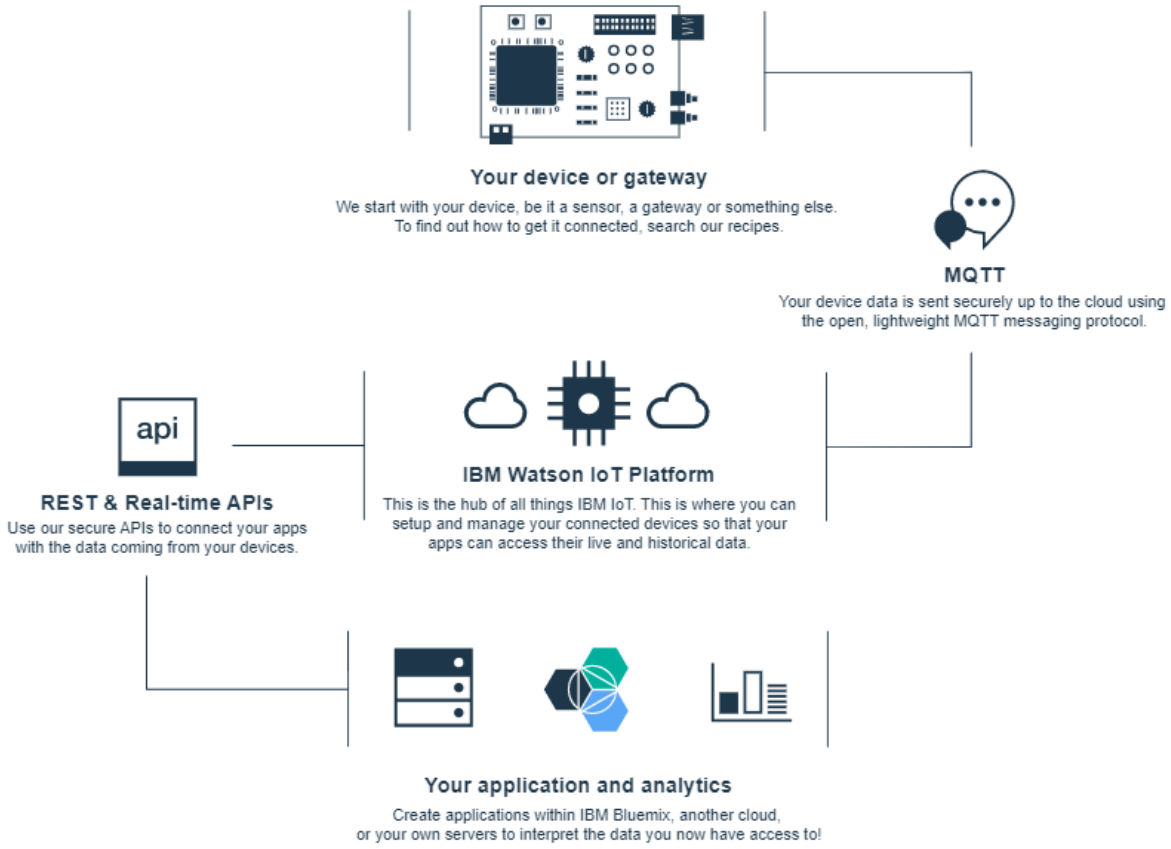



Рис. *15 – Схема роботи платформи IBM Watson IoT*

Як представлено на рис. 15, дані з пристроїв безпечно відсилаються за допомогою MQTT протоколу до IBM Watson IoT платформи, де ви можете додавати та налаштовувати ваші пристрої, щоб застосунки могли отримати доступ до їх даних. Потім можна підключати ваші застосунки за допомогою REST & Real-time APIs.

### 3.1.2 Потужна веб-панель інструментів

Платформа надає гнучку, масштабовану і просту у використанні потужну веб-панель інструментів, яку представлено на рис. 16.

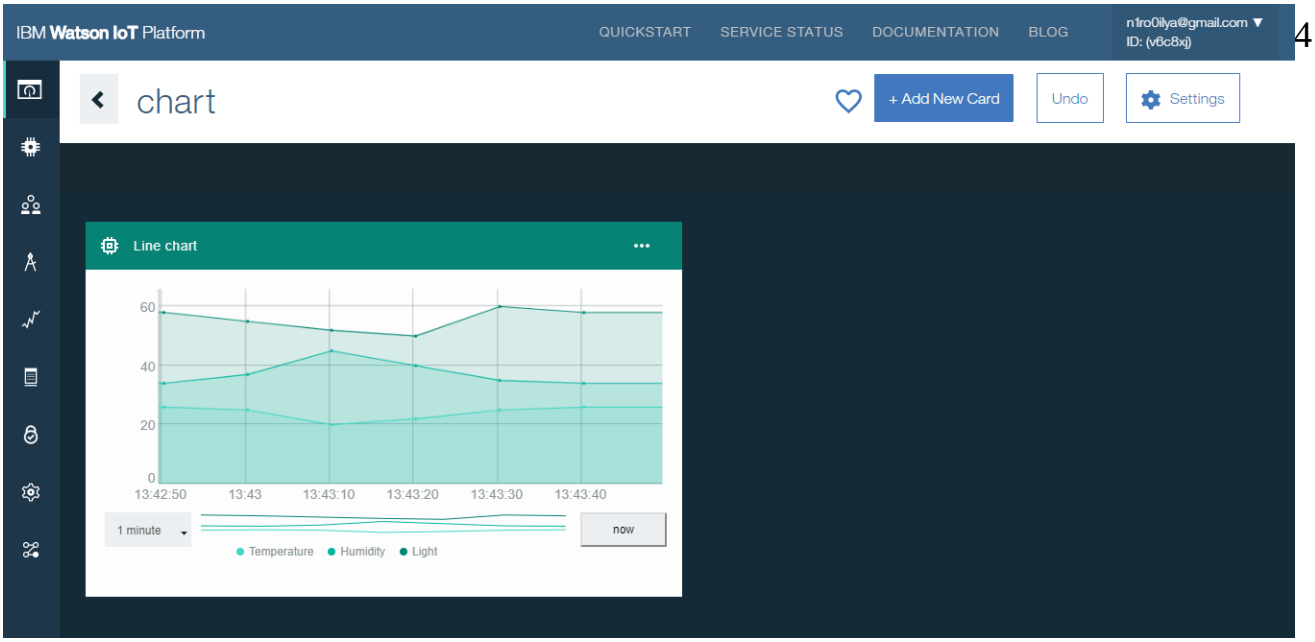



Рис. *16 – Веб-панель інструментів*

В веб-панелі було створено панелі та карти на цих панелях для відстеження даних пристрою. На рис. 16 зображено картку з лінійним графіком по даним отриманим з віртуального пристрою weather_device1024, який передає Temperature, Humidity та Light. Підтримуються наступні картки:

- Загальна візуалізація — значення одного або кількох наборів даних.
- Лінійний графік — один або декілька наборів даних в діаграмі прокрутки в реальному часі.
- Гістограма — значення параметрів даних в стовпчиках з мітками. Використовуючи меню "Налаштування", можна обрати горизонтальний або вертикальний напрям стовпчиків.
- Кругова діаграма - Два або більше наборів даних в круговому представленні.
- Значення — значення одного або кількох наборів даних.
- Вимірювання — Значення набору даних, показаного як вимірювання. Використовуйте меню «Налаштування», щоб за бажанням встановити порогові значення для нижнього, середнього та верхнього діапазонів значень.
- Властивості пристрою — конкретні властивості для одного або декількох пристроїв.

- Всі властивості пристрою — всі властивості для одного або декількох пристроїв.
- Список пристроїв — список для відстеження кількох пристроїв. Список може бути використаний як джерело даних для інших карток.
- Інформація про пристрій — основна інформація для одного пристрою.
- Карта пристрою — розташування пристроїв у списку пристроїв.

Є можливість створювати схему типу пристрою та зв’язати властивості пристрою, щоб потім створити правила, які ініціюються на основі значень даних з ваших пов'язаних властивостей пристрою.

Також є функціонал створення правил та дій. Правила — це умови, засновані на стані, які відповідають показникам пристрою у режимі реального часу з попередньо визначеними пороговими значеннями або іншими даними про властивості для активації сповіщення, якщо умова виконується. На додаток до сповіщення, яке відображається на інформаційній панелі Watson IoT платформи, додається одна чи кілька дій для запуску бізнес-логіки, коли спрацьовує правило.

Умови додаються в паралельні рядки, щоб застосувати їх як умови OR, або їх можна додати в послідовні стовпці, щоб застосувати їх як умови AND. Погляньте на наступні приклади:

- Просте правило може викликати попередження, якщо значення параметра перевищує вказане значення: Condition = temp_cpu>80
- Більш складне правило може спрацьовувати, коли виконується поєднання порогових значень: Condition = temp_cpu>60 AND cpu_load>90

Щоб викликати умову, яка порівнює дві або більше властивостей, або викликати дві або більше умови порівняння властивостей, які об'єднані послідовно, використовуючи AND, дані запуску спрацьовування передаються в одному

повідомленні пристрою. Якщо дані отримані в кількох повідомленнях, умови або послідовні умови не спрацьовують.

### 3.1.3 Підключення пристроїв за допомогою Python

Було використано Python для підключення пристроїв, які взаємодіють із створеною організацією на IBM Watson IoT Platform, приклад можно побачити на Лістингу 1. Клієнт Python для платформи Watson IoT пропонує API, що полегшує просту взаємодію з функціями платформи Watson IoT, абстрагувавшись від нижче лежачих базових протоколів, таких як MQTT та HTTP.

Лістинг *1*
*Підключення пристрою*

```
import time
import sys
try:
    import ibmiotf.device
except ImportError:
    # This part is only required to run the sample from within
the samples
    # directory when the module itself is not installed.
    #
    # If you have the module installed, just use "import
ibmiotf.device"
    import os
    import inspect
    cmd_subfolder = os.path.realpath(

os.path.abspath(os.path.join(os.path.split(inspect.getfile(ins
pect.currentframe()))[0], "../../src")))
    if cmd_subfolder not in sys.path:
        sys.path.insert(0, cmd_subfolder)
    import ibmiotf.application
    import ibmiotf.device

organization = "v6c8xj"
deviceType = "weather_device"
deviceId =  'weather_device1024'
authMethod = 'token'
authToken = 'qwerty1234'
```

```
try:
    deviceOptions = {"org": organization, "type": deviceType,
"id": deviceId, "auth-method": authMethod,
                     "auth-token": authToken}
    deviceCli = ibmiotf.device.Client(deviceOptions)
except Exception as e:
    print("Caught exception connecting device: %s" % str(e))
    sys.exit()
# Connect and send a datapoints: "Temperature", "Light", "Hu-
midity"  into the cloud as an event of type "weather_info" 10
times
deviceCli.connect()

for x in range(0, 12):
    data = {'Temperature': x,
          'Light': x,
          'Humidity': x
         }

    def myOnPublishCallback():
        print("Confirmed event %s received by IoTF\n" % x)

    success = deviceCli.publishEvent("weather_info ", "json",
data, qos=0, on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")
    time.sleep(1)

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```

### 3.1.4 Підключення застосунків за допомогою Python

Було використано Python для створення та розробки додатків, які взаємодіють із організацією на IBM Watson IoT Platform. Клієнт Python для платформи Watson IoT пропонує API, що полегшує просту взаємодію з функціями платформи Watson IoT, абстрагувавшись від нижче розташованих базових протоколів, таких як MQTT та HTTP.

**Конструктор та аутентифікація.** Словник параметрів для конструктора, який використовувався при підключенні, включає в себе параметри, які використовуються для взаємодії з платформою Watson IoT Platform. Конструктор створює клієнтський екземпляр і приймає словник параметрів, який містить наступні визначення:

- orgId — ідентифікатор організації.
- appId — унікальний ідентифікатор застосунку в організації.
- auth-method — метод аутентифікації. Єдиний метод, який підтримується — apikey.
- auth-key — опціональний API ключ, який було вказано, через обраний метод аутентифікації apikey.
- auth-token — токен API ключу, який також необхідний, через встановлення методу аутентифікації apikey.
- clean-session — "True" або "False" значення, яке потрібно лише, якщо потрібно підключити програму в режимі тривалої підписки. За замовчуванням встановлено значення "True".

Якщо б не було надано словника параметрів, то клієнт підключався б до послуги Watson IoT Platform Quickstart як незареєстрований пристрій. Приклад використання конструктора, який використовувався для підключення під час досліджень, наведено в Лістингу 2.

Лістинг *2*

*Використання конструктора*

```
import ibmiotf.application
try:
  options = {
    "org": organization,
    "id": appId,
```

```
        "auth-method": authMethod,
        "auth-key": authKey,
        "auth-token": authToken,
        "clean-session": true
      }
      client = ibmiotf.application.Client(options)
    except ibmiotf.ConnectionException  as e:
     …
```

Альтернативою до словника параметрів, під час підключення, був файл налаштування, який містить словник параметрів у форматі коду наведеного в Лістингу 3.

Лістинг *3*

*Використання файлу налаштувань*

```
import ibmiotf.application
try:
  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)
except ibmiotf.ConnectionException as e:
...
```

Файл конфігурації програми повинен мати формат представлений у Лістингу 4.

Лістинг *4*

*Файл конфігурації програми*

```
[application]
org=orgId
```

```
id=myApplication
auth-method=apikey
auth-key=key
auth-token=token
clean-session=true/false
```

**Підписка на події пристрою.** Події - це механізм, за допомогою якого пристрої публікують дані до екземпляру платформи Watson IoT. Пристрій керує вмістом події та присвоює ім'я кожної події, яку він надсилає.

Коли подію отримує екземпляр Платформи Watson IoT, перевіряються дані аутентифікації передаючого пристрою, що робить неможливим, підміну справжнього пристрою.

За замовчуванням програми підписуються на всі події з усіх під'єднаних пристроїв. Під час підключення використовувались параметри deviceType, DeviceId, event та msgFormat, щоб контролювати обсяг підписки. Один клієнт може підтримувати кілька підписок. Наведені нижче приклади коду показують, як використовувались параметри deviceType, DeviceId, event та msgFormat для визначення обсягу підписки під час дослідження:

Підписка на всі події з усіх пристроїв представлена в Лістингу 5.

Лістинг *5*

*Підписка на всі події з усіх пристроїв*

```
import ibmiotf.application
  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)

client.connect()
client.subscribeToDeviceEvents()
```

Підписка на всі події з усіх пристроїв певного типу представлена в Лістингу 6.

Лістинг 6

*Підписка на всі події з усіх пристроїв певного типу*

```
import ibmiotf.application

  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)

client.connect()
client.subscribeToDeviceEvents(deviceType=myDeviceType)
```

Підписка на певну подію з усіх пристроїв представлена в Лістингу 7.

Лістинг 7

*Підписка на певну подію з усіх пристроїв*

```
import ibmiotf.application

  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)

client.connect()
client.subscribeToDeviceEvents(event=myEvent)
```

Підписка на певну подію з двох або більше різних пристроїв представлена в Лістингу 8.

Лістинг *8*

*Підписка на певну подію з двох або більше різних пристроїв*

```
import ibmiotf.application

  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)

client.connect()
client.subscribeToDeviceEvents(deviceType=myDeviceType,
deviceId=myDeviceId, event=myEvent)
client.subscribeToDeviceEvents(deviceType=myOtherDeviceTy
pe, deviceId=myOtherDeviceId, event=myEvent)
```

Підписка на всі події, які публікуються у форматі JSON представлена в Лістингу 9.

Лістинг *9*

*Підписка на всі події, які публікуються у форматі JSON*

```
import ibmiotf.application

  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)

client.connect()
```

```
    client.subscribeToDeviceEvents(deviceType=myDeviceType,
deviceId=myDeviceId, msgFormat="json")
```

**Обробка подій із пристроїв.** Для обробки подій, отриманих по підпискам, було зареєстровано метод зворотного виклику подією як наведено у Лістингу 10. Повідомлення поверталися як екземпляр класу Event:

- String event.device — унікально ідентифікує пристрій серед усіх типів пристроїв в організації.
- String event.deviceType — визначає тип пристрою. Як правило, тип пристрою — це група для пристроїв, які виконують певне завдання, наприклад "weather_info".
- String event.deviceId — представляє ідентифікатор пристрою. Як правило, для данного типу пристрою deviceId є унікальним ідентифікатором цього пристрою, наприклад, серійним номером або MAC-адресою.
- String event.event — зазвичай використовується для групування конкретних подій, наприклад "статус", "попередження" та "дані".
- String event.format — формат може бути будь-яким рядком, наприклад JSON.
- Dictionary event.data — дані повідомлення. Максимальна довжина 131072 байт.
- Date and time event.timestamp — дата та час події.

Лістинг *10*

*Обробка подій, отриманих по підпискам*

```
        print(str % (event.format, event.event,
event.device, json.dumps(event.data)))
```

```
...
client.connect()
client.deviceEventCallback = myEventCallback
client.subscribeToDeviceEvents()
```

**Підписка на статус пристрою.** За замовчуванням, коли була зроблена підписка на статус пристрою, то було отримано оновлення статусу для всіх підключених пристроїв. Було використано параметри типу та ідентифікатора, щоб контролювати обсяг підписки. Один клієнт може підтримувати кілька підписок.

Зроблена підписка на оновлення статусу всіх пристроїв, яка представлена в Лістингу 11.

Лістинг *11*

*Підписка на оновлення статусу всіх пристроїв*

```
import ibmiotf.application

  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)

client.connect()
client.subscribeToDeviceStatus()
```

Підписка на оновлення статусу всіх пристроїв певного типу представлена в Лістингу 12.

Лістинг *12*

*Підписка на оновлення статусу всіх пристроїв певного типу*

```
import ibmiotf.application
```

```
      options =
ibmiotf.application.ParseConfigFile(configFilePath)
      client = ibmiotf.application.Client(options)

    client.connect()
    client.subscribeToDeviceStatus(deviceType=myDeviceType)
```

Підписка на оновлення статусу двох різних пристроїв представлена в Лістингу 13.

Лістинг *13*

*Підписка на оновлення статусу двох різних пристроїв*

```
    import ibmiotf.application

      options =
ibmiotf.application.ParseConfigFile(configFilePath)
      client = ibmiotf.application.Client(options)

    client.connect()
    client.subscribeToDeviceStatus(deviceType=WeatherDevi
ceType, deviceId=WeatherDevice1)
    client.subscribeToDeviceStatus(deviceType=WeatherDevi
ceType, deviceId= WeatherDevice2)
```

**Обробка оновлень статусу з пристроїв**

Щоб обробляти оновлення статусу, отримані підписками, було зареєстровано метод виклику подією як наведено в Лістингу 14. Повідомлення повертались як екземпляр класу Status.

Існує два типи подій: події підключення та відключення. Усі події статусу включають такі властивості:

- String status.clientAddr
- String status.protocol
- String  status.clientId
- String  status.user
- Date and time status.time
- String status.action
- Date and time status.connectTime
- Integer status.port

Властивість status.action визначає тип події статусу: Connect чи Disconnect. Події типу Disconnect містять такі додаткові властивості:

- Integer status.writeMsg
- Integer status.readMsg
- String  status.reason
- Integer status.readBytes
- Integer status.writeBytes

Лістинг *14*

*Обробка оновленого статусу, отриманого підписниками*

```
import ibmiotf.application

  options =
ibmiotf.application.ParseConfigFile(configFilePath)
  client = ibmiotf.application.Client(options)

def myStatusCallback(status):
```

```
      if status.action == "Disconnect":
        str = "%s - device %s - %s (%s)"
        print(str % (status.time.isoformat(), status.device,
status.action, status.reason))
        else:
          print("%s - %s - %s" % (status.time.isoformat(),
status.device, status.action))
      ...

    client.connect()
    client.deviceStatusCallback = myStatusCallback
    client.subscribeToDeviceStstus()
```

**Публікація подій з пристроїв**

Також було використано можливість публікації події так, наче вони походять з будь-якого пристрою організації як наведено в Лістингу 15.

Лістинг *15*

*Публікація подій з пристроїв*

```
    import ibmiotf.application

      options =
ibmiotf.application.ParseConfigFile(configFilePath)
      client = ibmiotf.application.Client(options)

    client.connect()
    myData={'Temperature' : 12, 'Humidity' : 40, 'Light' :
50}
```

```
    client.publishEvent(myDeviceType, myDeviceId, "status",
"json", myData)
```

**Публікація команд на пристрої**

Публікація команд на підключені пристрої виконувалась як представлено в Лістингу 16.

Лістинг *16*

*Публікація команд на підключені пристрої*

```
    import ibmiotf.application

      options =
ibmiotf.application.ParseConfigFile(configFilePath)
      client = ibmiotf.application.Client(options)

    client.connect()
    commandData={'TurnOff' : 50}
    client.publishCommand(myDeviceType, myDeviceId,
"reboot", "json", commandData)
```

## 3.2 SAP хмарна платформа

Сервіси SAP Cloud Platform (IoT) надають взаємодію операційних та інформаційних технологій, щоб зробити машини розумнішими та вести end-to-end цифрові передачі. SAP Cloud Platform Internet of Things надає можливість вбудовувати, налаштовувати та керувати будь-якими видами віддалених пристроїв. Крім того, SAP Cloud Platform IoT забезпечує можливості прийняття рішень та

інструменти, що веде до значної оптимізації ваших процесів в основі вашого бізнесу [31].

**Великомасштабне керування пристроями.** За допомогою IoT SAP Cloud Platform можна безпечно з'єднатися з великою кількістю пристроїв за допомогою різноманітних протоколів, щоб отримувати дані. Служба IoT надає вам повну гнучкість щодо того, де і як обробляти дані IoT - або в вашій мережі, або на платформі SAP Cloud Platform. Використовуйте додаткові служби SAP Cloud Platform, щоб аналізувати IoT в режимі реального часу або взаємодіяти з основними бізнес-процесами.

### 3.2.1 Схема роботи платформи

На рис. 17 зображене схематичне представлення  роботи платформи.

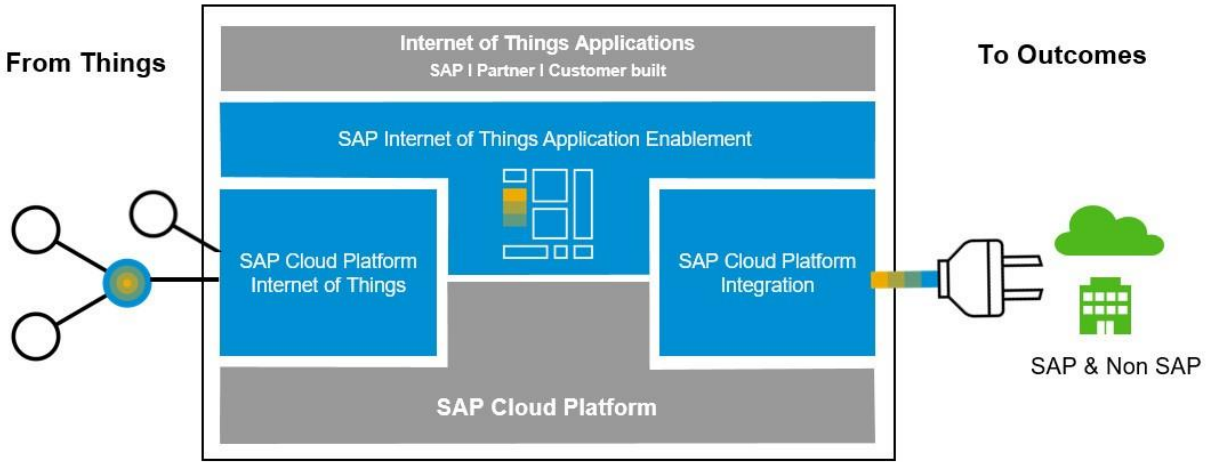



Рис. *17 – Схема роботи платформи*

### 3.2.2 Підключення пристроїв за допомогою Python

Через те що для Sap IoT хмарної платформи не має бібліотеки обгортки над протоколами прикладного рівня. То для обміну даними з платформою можна було обрати 3 варіанти протоколів: HTTP/MQTT/WEBSOCKET. Був обраний протокол MQTT через мінімальні обчислювальні вимоги в порівняння з HTTP

та WEBSOCKET, як зазначено в розділі 1.1.6, де це детально описано. В лістингу 17 наведено приклад підключення пристрою.

Лістинг *17*

*Приклад підключення пристрою*

```
    import time
    import paho.mqtt.client as paho
    broker="broker.hivemq.com"
    port=1883
    def on_publish(client,userdata,result): #create function
for callback
    print("data published \n")
    client1= paho.Client("weatherdevice1") #create client
object
    client1.on_publish = on_publish #assign function to
callback
    client1.connect(broker, port) #establish connection
    while True:
        data = {"Temperature": 12, "Humidity": 55, "Light":
40}
        res = client1.publish("sapiot/weatherdevice1","data")
#publish
        time.sleep(3)
```

### 3.2.3 Підключення застосунків за допомогою Python

На відміну від IoT  хмарної платформи IBM Watson MQTT брокера необхідно надавати самому. Був обраний безкоштовний брокер "broker.hivemq.com". В лістингу 18 приведено код застосунку, який використовувався для отримання даних, які публікують пристрої.

Лістинг *18*

```
import paho.mqtt.client as paho
broker="broker.hivemq.com"
port=1883
def on_message(mosq, obj, msg):
    global message
    print(msg.topic + " " + str(msg.qos) + " " +
str(msg.payload))
    message = msg.payload
client1= paho.Client("weather_app") #create client object
client1.connect(broker, port) #establish connection
client1.on_message = on_message
client1.subscribe("sapiot/weatherdevice1", 0)
client1.loop_start()
```

### 3.3 Хмарна платформа AWS IoT Core

AWS IoT Core - це керована хмарна платформа, яка дозволяє підключеним пристроям просто та безпечно взаємодіяти з хмарними додатками та іншими пристроями. AWS IoT Core підтримує роботу з мільярдами пристроїв і трильйонів повідомлень, що дозволяє надійно та безпечно обробляти і спрямовувати ці повідомлення до адрес AWS та інших пристроїв. З AWS IoT Core ваші програми зможуть постійно стежити за всіма використовуваними пристроями і взаємодіяти з ними, навіть коли вони знаходяться в автономному режимі [32].

AWS IoT Core спрощує роботу з такими сервісами AWS, як AWS Lambda, Amazon Kinesis, Amazon S3, Amazon Machine Learning, Amazon DynamoDB, Amazon CloudWatch, AWS CloudTrail та Amazon Elasticsearch Сервіс з вбудованими можливостями інтеграції з платформою Kibana, що дозволяє створювати

застосунки Інтернету речей для збирання, обробки, аналізу даних, що генеруються підключеними пристроями, а також виконання дій на основі цих даних без необхідності керувати будь-якою інфраструктурою.

Надає такі можливості:

**Підключення пристроїв та керування ними.** AWS IoT Core дозволяє легко підключати пристрої до хмари або один до одного, як представлено на рис. 18. AWS IoT Core підтримує протоколи HTTP, WebSockets і спрощений протокол зв'язку MQTT, спеціально спроектований для підтримки нестабільного підключення, скорочення обсягу коду, який передається пристрою, і роботи в мережах з низькою пропускною здатністю. AWS IoT Core також підтримує інші стандартні і призначені для користувача протоколи, забезпечуючи взаємодію між пристроями навіть в разі використання ними різних протоколів.

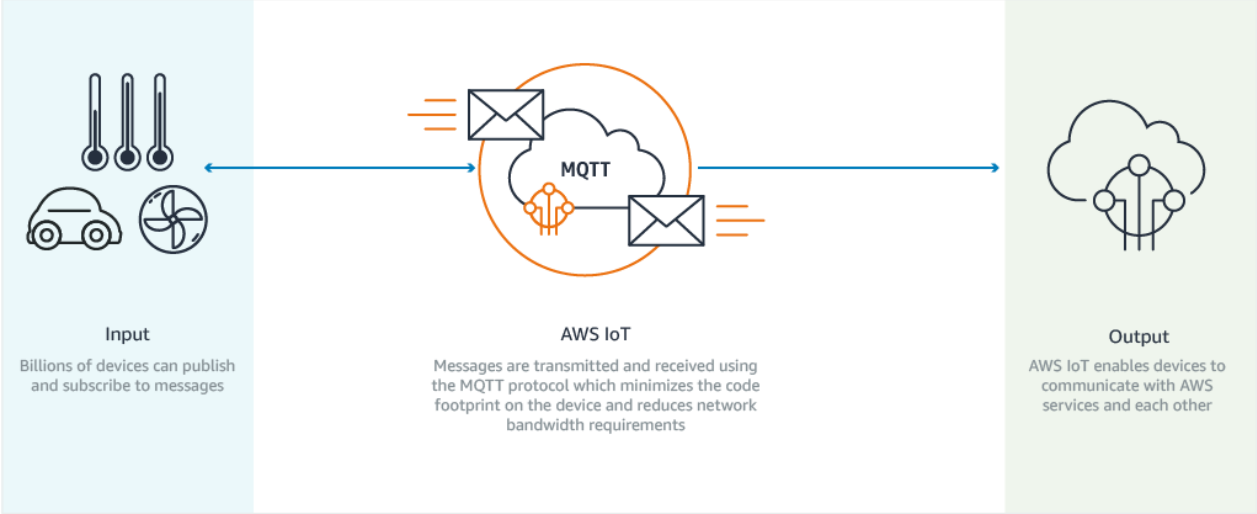



Рис. *18 – Взаємодія пристроїв з хмарою*

**Безпека підключень та даних.** AWS IoT Core забезпечує аутентифікацію і наскрізне шифрування між усіма точками підключення, дивіться схематичне представлення на рис. 19. Таким чином, будь-який обмін даними між пристроями і AWS IoT Core відбувається тільки після перевірки ідентифікації. Крім того,

можна забезпечити безпечний доступ до своїх пристроїв і додатків, застосовуючи політики з точним налаштуванням дозволів.

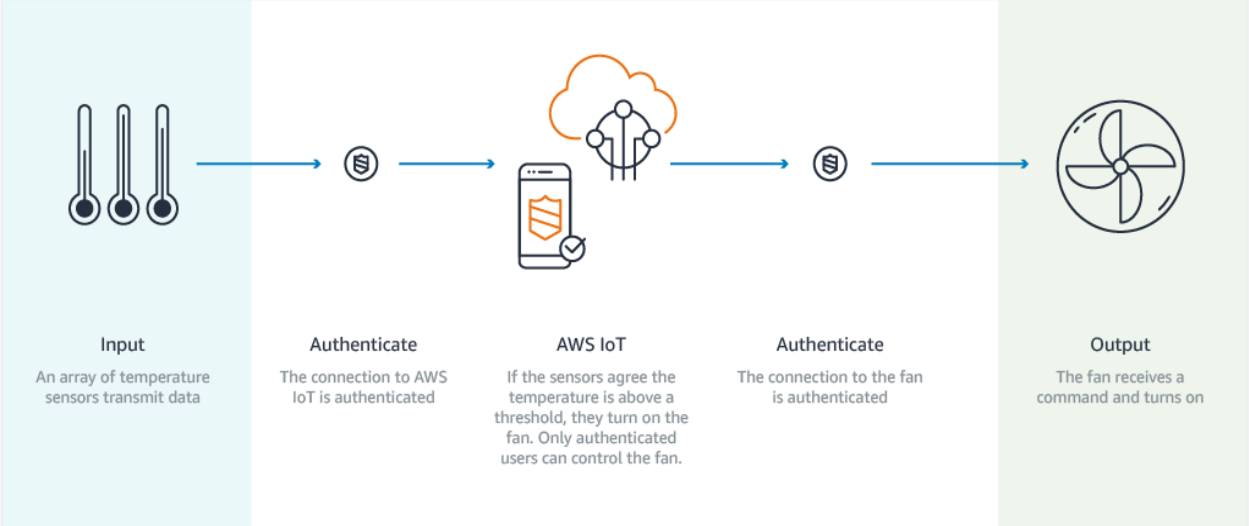



Рис. *19 – Аутентифікація між усіма точками підключення*

**Зчитування і настройка стану пристроїв в будь-який момент.** AWS IoT Core зберігає останній стан пристрою, що дозволяє в будь-який момент зчитати або задати цей стан. Завдяки цьому застосунки визначають пристрій так, як ніби пристрій постійно підключено. Це означає, що додаток може зчитувати стан пристрою, навіть коли пристрій працює автономно, а також дозволяє задати стан пристрою, який буде застосовано при наступному його підключенні, приклад можна побачити на рис. 20.


Рис. *20 – Зчитування і настройка стану пристроїв*

### 3.4 Azure IoT Suite

Нажаль, для хмарної платформи Azure IoT Suite були проведені лише теоретичні дослідження через відсутній безкоштовний план для опробування функціоналу та можливостей платформи.

**Швидке з’єднання та масштабування.** Надає змогу використовувати попередньо налаштовані рішення, додавати нові пристрої та з'єднувати існуючі за допомогою SDK пристроїв для декількох платформ, включаючи Linux, Windows і системи працюючі в реальному часі. Легко масштабується від декількох сенсорів до мільйонів одночасно під'єднаних пристроїв [33].

**Аналіз та дії на основі невикористаних даних.** Збирайте раніше невикористані дані з пристроїв і датчиків, а також надає можливість використовувати вбудовані інструменти для візуалізації та обробки цих даних. Налаштування аналітики в режимі реального часу за допомогою синтаксису на основі SQL за допомогою масштабованого, високопродуктивного та еластичного способу, без необхідності керувати складною інфраструктурою та програмним забезпечен-

ням. Присутня велика бібліотека алгоритмів для того щоб розширити аналітику передбачення, аналітичні та машино навчальні рішення в режимі реального часу, інтегруючи код з таких мов, як R та Python безпосередньо у робочий простір.

**Інтеграція Azure IoT Suite.** Легко інтегруйте з системами та додатками, включаючи Salesforce, SAP, Oracle Database та Microsoft Dynamics, що полегшує доступ до даних і підтримує різнорідні системи. Витримує мільйони повідомлень на неоднорідні пристрої за допомогою мобільного механізму сповіщення із меншою кількістю зусиль для розробки. Підтримує створення мобільних та вебпрограм, які інтегруються з сторонніми та Microsoft веб APIs, і надає змогу використання OAuth 2.0 для створення власних безпечних веб APIs.

**Надає можливість аутентифікації кожного пристрою.** Надає змогу налаштування індивідуальних ідентифікаційних даних та облікових даних для кожного з підключених пристроїв, а також зберігає конфіденційність повідомлень від хмари до пристрою та від пристрою до хмари. Також надає можливість вибірково скасовувати права доступу певних пристроїв для забезпечення цілісності системи.

# РОЗДІЛ 4 ПОРІВНЯННЯ ДОСЛІДЖЕНИХ ІОТ ПЛАТФОРМ

## 4.1 Порівняння функціоналу

На основі виконаних досліджень, проаналізовані платформи можна порівняти на основі функціоналу який вони надають, дивіться табл. 2. Якщо функціонал відсутній то ставиться –, а у випадку наявності +. Функціонал, який надається:

- Ідентифікатор пристрою: керування унікальними ідентифікаторами пристроїв.
- Аутентифікація пристрою: можливість під’єднання тільки за наявності API ключів.
- Управління пристроями: керування метаданими пристроїв та виконання операцій, таких як перезавантаження пристроїв та оновлення програмного забезпечення.
- Команди та керування: щоб пристрій виконував дію, надсилайте повідомлення на пристрій із хмари.
- Правила та дії: можливість робити дії на основі правил, які застосовуються до отриманих платформою даних від пристрою.
- Візуалізація даних: Інструменти для візуалізації даних такі як виведення основних характеристик пристрою, різних типів графіків та діаграм на основі даних.
- Події пристрою: події, які може викликати пристрій, такі як підключення пристрою та відключення.
- Інтелектуальна аналітика: аналіз даних пристрою на хмарі дає змогу прогнозувати, коли мають відбуватися певні дії. Наприклад, аналізуючи телеметричний двигун повітряного судна, потрібно визначити, коли потрібно забезпечити технічне обслуговування двигуна.

Таблиця *2*

*Порівняння функціоналу IoT хмарних платформ*

| Функціонал | IBM Watson IoT | SAP IoT | AWS IoT Core | Azure IoT Suite |
|---|---|---|---|---|
| Ідентифікатор пристрою | + | + | + | + |
| Аутентифікація пристрою | + | +/- | + | + |
| Управління пристроями | + | + | + | + |
| Команди та керування | + | + | + | + |
| Правила та дії | + | + | + | + |
| Візуалізація даних | + | + | + | + |
| Події пристрою | + | + | + | + |
| Інтелектуальна аналітика | + | +/- | + | + |

Всі 4 платформи мають можливість передачі даних через MQTT, HTTP, WEBSockets, але тільки IBM Azure IoT Suite підтримує AMQP, а в Sap IoT взагалі треба надавати свого MQTT брокера та налаштовувати для нього аутентифікацію. Найбільшу кількість сервісів для інтеграції має AWS IoT Core, а IBM Watson IoT найбільша кількість сервісів одразу інтегрована. В кількості варіан-

тів наявних мов програмування для спрощеного створення застосунків на платформі найкращим рішенням є IBM Azure IoT Suite з додатковими .Net, UWP (Universal Windows Platform).

### 4.2 Порівняння простоти використання

За рахунок одразу вбудованої потужної веб-панелі інструментів, перевагу отримує IBM Watson IoT, так як збереження даних, візуалізація даних, велика кількість різних типів графіків та діаграм, відображення статусу пристроїв, введення правил для оповіщень та виконання дій вже налаштовані в панелі. Все що необхідно від користувача це обрати, за допомогою інтуїтивно зрозумілого інтерфейсу, що необхідно відображати як зображено на рис. 21. Немає потреби витрачати час на підключення цих сервісів які наявні в 3 IoT хмарних платформах: SAP, AWS IoT Core, Azure IoT Suite, але потребують налаштування. Тому ця платформа найкраще підходить для початківців. На відміну від Sap IoT онлайн брокер також створюється платформою з вже налаштованою аутентифікацією як в AWS IoT Core та Azure IoT Suite.

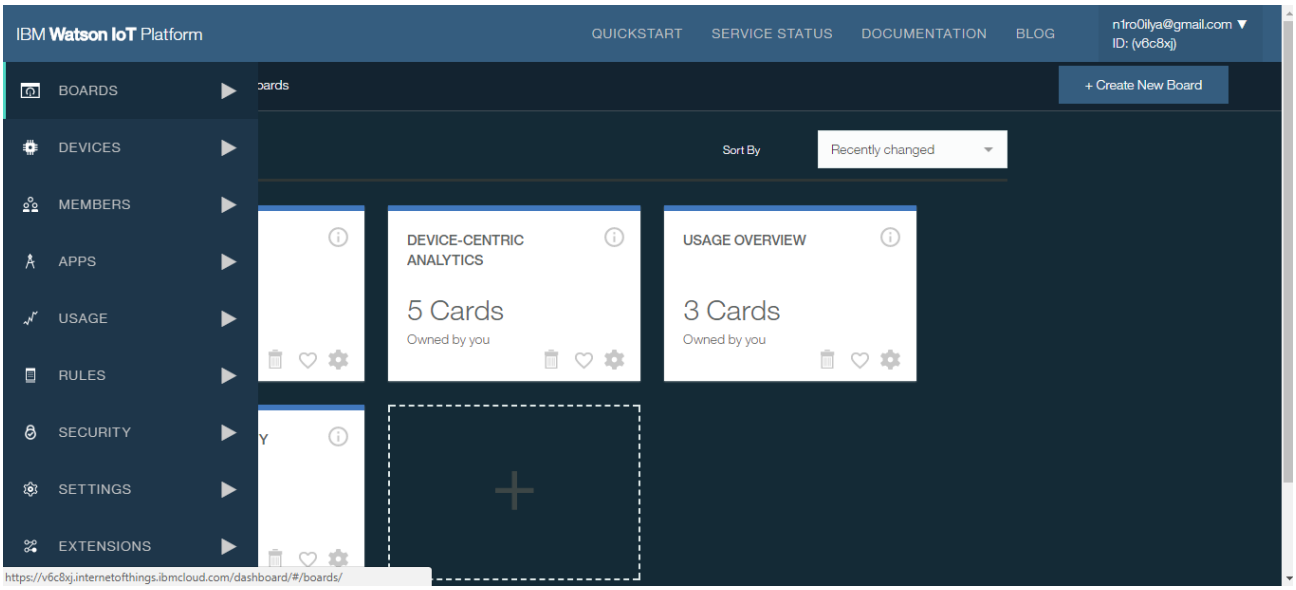



Рис. *21 – Зручна веб-панель інструментів*

# РОЗДІЛ 5 ОХОРОНА ПРАЦІ ТА ТЕХНОГЕННА БЕЗПЕКА

## 5.1 Аналіз небезпечних і шкідливих факторів, що впливають на програміста при розробці програмного комплексу

Питання охорони праці людини необхідно вирішувати на всіх стадіях трудового процесу незалежно від виду професійної діяльності [35].

Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, наприклад, шум і вібрація на робочому місці, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

У даному розділі вирішується питання охорони праці програміста на стадії розробки ним програмного комплексу та розрахунок середнього рівня шуму у робочому приміщенні програміста.

Приміщення, в якому працює програміст знаходиться в учбовому корпусі на кафедрі програмного забезпечення автоматизованих систем.

Приміщення має загальну площу 54 м$^2$, висоту стелі 3,2 м. У приміщенні знаходиться 9 робочих місць з ПК. Кожне робоче місце обладнане робочим столом площею 6м$^2$, стільцем та персональним комп'ютером, що складається з монітора, системного блоку, клавіатури та миші. Слід відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6м$^2$, а об'єм не менший за 20м$^3$, тобто об'єму даного приміщення не вистачає для розташування 9 робочих місць операторів ПК (19,2м$^3$<20м$^3$).

Аналіз умов праці показує, що у приміщенні лабораторії на програміста можуть негативно впливати наступні фізичні та психофізиологічні фактори:

- підвищена або знижена температура повітря робочої зони;

- підвищена або знижена вологість повітря;
- недостатня освітленість робочого місця;
- підвищений рівень шуму на робочому місці;
- підвищена іонізація повітря;
- підвищений рівень електромагнітних випромінювань;
- нервово-психічні перевантаження (розумова перенапруга);
- фізичні перевантаження (одноманітна поза викликає статичну втому);

В табл. 3 наведена карта умов праці програміста на робочому місці.

Таблиця *3*

*Карта умов праці програміста на робочому місці*

| Чинники виробничого середовища і процесу праці | Значення чинника (ГДК, ГДР) | | Тривалість дії чинника, в % за зміну | Оцінка факторів умов праці, бали |
|---|---|---|---|---|
| | Норма | Факт | | |
| Температура повітря на робочих місцях в теплий період року Т, С | 23-28 | 25 | 100 | 3 |
| Відносна вологість повітря ф, % | 61..75 | 66 | 100 | 3 |
| Рівень шуму L, дБ А | 60 | 58,7 | 100 | 3 |
| Освітленість приміщення Е, лк | 300 | 330 | 50 | 1 |
| Категорія зорових робіт, Зор. | Е | Д | 100 | 2 |
| Тривалість застережного спостереження, Дл., % | 25..50 | 25 | 100 | 2 |
| Кількість важливих об’єктів спостереження, Об, од. | 5 | 24 | | 4 |
| Список порушень вимог безпеки | 3 | | | |

**Заходи з поліпшення умов праці**

У результаті аналізу були виявлені порушення в організації безпосередньо самого робочого місця програміста. У зв'язку з цим пропонується організувати робоче місце програміста наступним способом:

- висота над рівнем підлоги робочої поверхні, на якій працює програміст, повинна складати 720 мм. Бажано, щоб робочий стіл при необхідності можна було регулювати по висоті в межах 680-780мм;
- оптимальний розмір поверхні столу 1600 х 1000 мм. Під столом повинен бути простір для ніг з розмірами по глибині 650 мм. Робочий стіл оператора повинен також мати підставку для ніг, розташовану під кутом 15° до поверхні столу. Довжина підставки - 400 мм, ширина - 350 мм. Відстань клавіатури від краю столу повинна бути не більш 300 мм, що забезпечить програмісту зручну опору для передплічь. Відстань між очима й екраном монітору повинна складати 40-80 см;
- робочий стілець програміста повинен бути оснащений підйомно-поворотним механізмом. Висота сидіння повинна регулюватися в межах 400-500мм. Глибина сидіння повинна складати не менш 380 мм, а ширина не менш 400 мм. Висота опорної поверхні спинки не менш 300 мм, ширина не менш 380 мм. Кут нахилу спинки стільця до площини сидіння повинен змінюватися в межах 90 - 110.

Виходячи з результатів аналізу важкості та напруженості праці пропоную скоротити час роботи за комп'ютером, робити перерви, сумарний час яких повинен складати 50 хвилин при 8-ми годинній зміні.

**Виробнича санітарія**

У даному розділі розглянемо повітряне середовище робочої зони програміста — температуру та вологість повітря, швидкість руху повітря, інтенсивність теплового випромінювання. Робота програміста за енерговитратами відноситься до категорії легких робіт Іа, Іб, тому повинні дотримуватися наступні вимоги згідно ДСН 3.3.6.042-99:

- оптимальна температура повітря — 22°С (допустима — 23-28°С);
- оптимальна відносна вологість — 40-60% (допустима — не більш 75%);
- швидкість руху повітря не більш 0,1 м/с.

Виміряні за допомогою приладів (психрометр Августа) температура та вологість у лабораторії майже відповідають вказаним у таблиці для теплого періоду року.

Розташовані у приміщенні 9 ПК являються джерелами тепловиділень, крім того для підтримання у приміщенні в холодний період року оптимальних параметрів мікроклімату використовуються нагріті поверхні опалювальної системи. Нормованим показником ІЧВ являється гранично допустима густина потоку енергії Іг.д. Вт/м$^2$, яка встановлюється в залежності від площі опромінюваної поверхні тіла людини (Sопр).

Нормовані рівні складають:

- Іг.д = 35 Вт/м$^2$ при Sопр > 50%;
- Іг.д = 70 Вт/м$^2$ при Sопр ~ 25-50%;
- Іг.д = 100 Вт/м$^2$ при Sопр < 25%.

Для створення й автоматичної підтримки в приміщенні незалежно від зовнішніх умов оптимальних значень температури, вологості, чистоти і швидкості руху повітря, у холодний час року використовується водяне опалення, у теплий час року застосовується кондиціонування повітря.

Нормованим параметром природного освітлення згідно ДБН В.2.5-28 — 2006 являється коефіцієнт природного освітлення (КПО). КПО встановлюється в за-

лежності від розряду виконуваних зорових робіт. Робота програміста відноситься до робіт середньої точності (IV розряд зорових робіт, мінімальний розмір об'єкту розрізнення складає 0,5 -1,0мм), для яких при використанні бокового освітлення КПО=1,5%. Для штучного освітлення нормованим параметром виступає мінімальний рівень освітленості та коефіцієнт пульсації світлового потоку, який не повинний бути більшим ніж 20%. Мінімальна освітленість встановлюється в залежності від розряду виконуваних зорових робіт. Для IV розряду зорових робіт вона складає 300-500 лк.

За даними вимірювань (люксметр Ю-116) рівень природної освітленості поверхні, де розташований ПК програміста, складає 350 лк при освітленості тієї же поверхні відкритим небосхилом в 35000 лк, тобто КПО = 1%, що не відповідає нормативному КПО.

Для штучного освітлення у приміщенні використовуються люмінесцентні лампи, які в порівнянні з лампами розжарювання мають ряд істотних переваг:

- за спектральним складом світла вони близькі до природного світла;
- мають підвищену світлову віддачу (у 2-5 разів вищу, ніж у ламп розжарювання);
- мають триваліший термін служби (до 10 тис. годин).

Допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітору комп'ютера представлені в таблиці 3. Нормованим параметром невикористаного рентгенівського виромінювання виступає потужність експозиції дози. На відстані 5 см від поверхні екрану монітору її рівень не повинен перевищувати 100 мкР/год. Максимальний рівень рентгенівського випромінювання на робочому місці зазвичай не перевищує 20 мкР/год.

Допустимі значення параметрів неіонізуючих електромагнітне випромінювання представлені в табл. 4:

Таблиця *4*

*Допустимі значення параметрів*

| Найменування параметра | | Допустимі значення |
|---|---|---|
| Напруженість електричної складової електромагнітного поля на відстані 50 см від поверхні відео монітора | | 10 В/м |
| Напруженість магнітної складової електромагнітного поля на відстані 50 см від поверхні відео монітора | | 0,3 Ам |
| Напруженість електростатичного поля не повинна перевищувати | Для дорослих користувачів | 20 кВ/м |
| | Для дітей | 15 кВ/м |

На відстані 5-10 см від екрана і корпуса монітора рівні напруженості можуть досягати 140 В/м по електричній складовій, що значно перевищує допустимі значення.

Для попередження впровадження небезпечної техніки всі дисплеї повинні бути сертифіковані.

Шум погіршує умови праці здійснюючи шкідливу дію на організм людини. Працюючі в умовах тривалої шумової дії відчувають дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену стомлюваність, зниження апетиту, біль у вухах і т.д. Такі порушення в роботі ряду органів і систем організму людини можуть викликати негативні зміни в емоційному стані людини аж до стресових. Під впливом шуму знижується концентрація, порушуються фізіологічні функції, з'являється втома у зв'язку з підвищеними енергетичними витратами і нервово-психічним напруженням, погіршується мовна комутація. Все це знижує працездатність людини і його продуктивність, якість і безпеку праці.

Як було вказано вище, у приміщенні знаходиться дев’ять робочих місць з ПК, кожне з яких обладнане монітором, вінчестером в системному блоці, трьома вентиляторами системи охолоджування ПК та клавіатурою. Крім того поряд працює периферійна техніка. Таким чином у приміщенні мають місце шуми механічного і аеродинамічного походження, широкосмугові із аперіодичним підсиленням при роботі принтерів. Допустимий еквівалентний рівень шуму для робочого місця програміста складає 60 дБА. Після проведення аналізу робочого місця програміста в лабораторії було з’ясовано, що воно відповідає встановленим вимогам.

**Електробезпека**

Приміщення за небезпекою ураження електричним струмом можна віднести до 1 класу, тобто це приміщення без підвищеної небезпеки (сухе, без пилу, з нормальною температурою повітря, ізольованими підлогами і малим числом заземлених приладів).

На робочому місці програміста з всього устаткування металевим є лише корпус системного блоку комп'ютера, але тут використовуються системні блоки, що відповідають стандартові фірми IBM, у яких крім робочої ізоляції передбачений елемент для заземлення і провід з жилою, що заземлює, для приєднання до джерела живлення.

Основні причини ураження людини електричним струмом на робочому місці:

- дотик до металевих неструмоведучих частин (корпусу, периферії комп'ютера), що можуть виявитися під напругою в результаті ушкодження ізоляції;
- нерегламентоване використання електричних приладів;
- відсутність інструктажу співробітників з правил електробезпеки.

На протязі роботи на корпусі комп'ютера накопичується статична електрика. На відстані 5-10 см від екрана напруженість електростатичного поля складає 60-280 кВ/м, тобто в 10 разів перевищує норму 20 кВ/м. Електробезпечність у приміщенні лабораторії пропоную забезпечити наступними технічними способами і засобами захисту:

- для зменшення накопичення статичної електрики застосовувати зволожувачі і нейтралізатори, антистатичне покриття підлоги;
- забезпечити приєднання металевих корпусів устаткування до жили, що заземлює. Заземлення корпуса ПК забезпечити підведенням жили, що заземлює, до розеток. Опір заземлення 4 Ом, згідно (ПУЄ) для електроустановок з напругою до 1000 В.

Також організаційними заходами:

- своєчасне проведення інструктажів з техніки безпеки;
- заборонена використання непередбачених у лабораторії електричних приладів, таких як електричні чайники, обігрівачі.

**Пожежна безпека**

Ступінь вогнестійкості будинків приймається в залежності від їхнього призначення, категорії по вибухопожежній і пожежній небезпеці, по поверховості, площі поверху в межах пожежного відсіку згідно НАПБ Б.03.002- 2007.

Будинок, у якому знаходиться приміщення по пожежній небезпеці будівельних конструкцій відноситься до категорії Д, оскільки тут присутні займисті (книги, меблі, оргтехніка і т.д.) і важко горючі речовини (сейфи, різне устаткування і т.д.), що при взаємодії з вогнем можуть горіти без вибуху.

По конструктивних характеристиках будинок можна віднести до будинків з несучими і огороджуючими конструкціями із природних або штучних кам'яних матеріалів, бетону або залізобетону, де для перекриттів допускається використання дерев'яних конструкцій, захищених штукатуркою або важкогорючими

листовими, а також плитними матеріалами. Отже, ступінь вогнестійкості будинку можна визначити як другу (II). Приміщення лабораторії по функціональній пожежній небезпеці відноситься до категорії В. Пожежа в приміщенні, може привести до дуже несприятливих наслідків (втрата коштовної інформації, псування майна, загибель людей і т.д.), тому необхідно:

- виявити й усунути всі причини виникнення пожежі;
- розробити план заходів для ліквідації пожежі в будинку;
- розробити план евакуації людей з будинку.

Причинами виникнення пожежі можуть бути:

- несправності електропроводки, розеток і вимикачів які можуть привести до короткого замикання або пробою ізоляції;
- використання ушкоджених (несправних) електроприладів;
- використання в приміщенні електронагрівальних приладів з відкритими нагрівальними елементами;
- виникнення пожежі внаслідок влучення блискавки в будинок;
- загоряння будинку внаслідок зовнішніх впливів;
- неакуратне поводження з вогнем і недотримання мір пожежної безпеки.

Для профілактики пожежі надзвичайно важлива правильна оцінка пожежонебезпеки будинку, визначення небезпечних факторів і обґрунтування способів і засобів пожежопопередження і захисту. Одне з умов забезпечення пожежобезпеки — ліквідація можливих джерел займання. У приміщенні джерелами займання можуть бути:

несправне електроустаткування, несправності в електропроводці, електричних розетках і вимикачах. Для виключення виникнення пожежі з цих причин необхідно вчасно виявляти й усувати несправності, проводити плановий огляд і вчасно усувати всі несправності;

- несправні електроприлади. Необхідні міри для запобігання пожежі містять у собі своєчасний ремонт електроприладів, якісне виправлення поломок, не використання несправних електроприладів
- обігрівання приміщення електронагрівальними приладами з відкритими нагрівальними елементами. Відкриті нагрівальні поверхні можуть спричинити пожежу, тому що в приміщенні знаходяться паперові документи і довідкова література у виді книг, посібників, а папір — легкозаймистий предмет. З метою профілактики пожежі пропонується не використовувати відкриті обігрівальні прилади в приміщенні лабораторії;
- коротке замикання в електропроводці. З метою зменшення імовірності виникнення пожежі внаслідок короткого замикання необхідно, щоб електропроводка була схованою;
- влучення в будинок блискавки. У літній період під час грози можливе влучення блискавки внаслідок чого можлива пожежа. Щоб уникнути цього я рекомендую установити на даху будинку блискавковідвід;
- для усунення загоряння в результаті паління в приміщенні пропоную категорично заборонити паління, а дозволити тільки в строго відведеному для цього місці.

З метою запобігання пожежі пропоную проводити з інженерами, що працюють у приміщенні, протипожежний інструктаж, на якому ознайомити працівників із правилами протипожежної безпеки, а також навчити використанню первинних засобів пожежогасіння.

У випадку виникнення пожежі необхідно відключити електроживлення, викликати по телефону пожежну команду, евакуювати людей із приміщення відповідно до плану евакуації і приступити до ліквідації пожежі вогнегасниками. При наявності невеликого вогнища полум'я, можна скористатися підручними засобами з метою припинення доступу повітря до об'єкта загоряння.

**Розрахунок середнього шуму на робочому місці програміста**

Розрахунок середнього рівня шуму на робочому місці програміста виконуємо з урахуванням рівнів звукового тиску від різних джерел, які приведені в табл. 5.

Таблиця *5*

*Рівні звукового тиску від різних джерел*

| Джерело шуму | Рівень шуму, дБА |
|---|---|
| Жорсткий диск | 40 |
| Вентилятор | 40 |
| Принтер | 45 |
| Сканер | 45 |

Рівень шуму, що виникає від декількох некогерентних джерел, що працюють одночасно, підраховується на підставі принципу енергетичного підсумовування рівня інтенсивності окремих джерел.

$$L_{\Sigma} = 101lg \sum 10^{0,1L_i} \quad (1)$$

де, $L_i$ – рівень звукового тиску і-го джерела шуму; n – кількість джерел шуму. Підставивши значення рівняння звукового тиску для кожного виду устаткування у формулу (1), отримаємо:

$$L = 10 \lg \ (10^{4,0} + \ 10^{4,0} + 10^{5,0} + \ 10^{4,5}) = 49{,}2 \text{ дБА}.$$

За наявності декількох джерел шуму з однаковим рівнем інтенсивності $L_i$ загальний рівень шуму визначають за формулою:

$$L = Li + 10lgn$$

У нашому випадку таких джерел дев’ять, отже загальний рівень шуму буде визначатися так:

$$L = 49{,}2 + 10 \lg 9 = 58{,}7 \text{ дБА}.$$

Розраховане значення середнього рівня шуму не перевищує гранично допустимий рівень шуму для робочого місця програміста, тобто не треба передбачати заходи по зниженню рівня шуму.

**Висновки до розділу охорони праці**

Детально проаналізовано небезпечні і шкідливі фактори, які можуть діяти у робочому приміщенні. Виконано порівняння нормативних показників з фактичними значеннями величин, що нормуються.

Зроблено опис повітряного середовища робочої зони – температури, вологості повітря та швидкості руху повітря.

Розглянуто основні причини ураження людини електричним струмом на робочому місці та розроблено заходи щодо їх усунення.

Проаналізована ступінь вогнестійкості будинку щодо відповідності вимогам пожежної безпеки. З метою запобігання пожежі визначено ймовірні причини пожеж та розроблено заходи щодо їх усунення. На підставі властивостей та кількості пожежонебезпечних речовин і матеріалів, що знаходяться на об’єкті, обґрунтовується категорія вибухопожежонебезпечності приміщення.

Виконано розрахунок середнього шуму на робочому місці програміста, який не перевищує гранично допустимий рівень, з чого зроблено висновок про те, що не треба передбачати заходи по зниженню рівня шуму.

# ВИСНОВКИ

У результаті виконання даної магістерської роботи було отримано наступні результати:

1. Досліджено Інтернет речей.
2. Досліджена архітектура та реалізації Інтернету речей.
3. Досліджені протоколи для передачі даних в Інтернеті речей.
4. Проаналізовані та протестовані наявні IoT хмарні платформи.
5. Виведена порівняльна характеристика досліджених IoT хмарних платформ.
6. Порівняльна характеристика та детальній опис представлених інструментів спростить процес вибору необхідної платформи для розробки IoT рішення.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Росляков А. В. Интернет вещей / А. В. Росляков, С. В. Ваняшин, А. Ю. Гребешков. – Самара, 2015.
2. The Internet of Things (IoT) cloud platform market [Електронний ресурс] – Режим доступу до ресурсу: https://www.marketsandmarkets.com/Market-Reports/iot-cloud-platform-market-195182.html?gclid=EAIaIQobChMIv_-s-LTD2AIVRYwZCh3rOg0DEAAYASAAEgJ9D_D_BwE.
3. NoSQL [Електронний ресурс] – Режим доступу до ресурсу: http://nosql-database.org.
4. Apache Hadoop [Електронний ресурс] – Режим доступу до ресурсу: https://hadoop.apache.org/.
5. IoT Cloud Platform - temboo [Електронний ресурс] – Режим доступу до ресурсу: https://temboo.com/library/Library/Labs.
6. Arduino [Електронний ресурс] – Режим доступу до ресурсу: http://arduino.cc.
7. Release and deployment management [Електронний ресурс] – Режим доступу до ресурсу: www.tutorialspoint.com/itil/release_and_deployment_management.htm.
8. L. Xu, W. He, S. Li
   Internet of things in industries: a survey
   IEEE Trans Ind Inf, 10 (4) (2014), pp. 2233-2243
9. IoT Cloud Platform - thingspeak [Електронний ресурс] – Режим доступу до ресурсу: https://thingspeak.com/docs/tutorials#analyze.
10. HomeOS [Електронний ресурс] – Режим доступу до ресурсу: http://research.microsoft.com/en-us/projects/homeos/.
11. Z-Wave [Електронний ресурс] – Режим доступу до ресурсу: www.z-wave.com.

12. Identity-as-a-Service-IDaaS [Електронний ресурс] - ежим доступу до ресурсу: http://searchmobilecomputing.techtarget.com/definition/identity-as-a-Service-IDaaS
13. LoRa [Електронний ресурс] – Режим доступу до ресурсу: https://www.lora-alliance.org/.
14. D. Bandyopadhyay, J. Sen: Internet of things: applications and challenges in technology and standardization, Wireless Personal Communications, 58 (1) (2011), pp. 49-69
15. Homeos [Електронний ресурс] – Режим доступу до ресурсу: http://research.microsoft.com/en-us/projects/homeos/.
16. Babu SM, Lakshmi AJ, Rao BT. A study on cloud based internet of things: CloudIoT. In: Proceedings of 2015 global conference on communication technologies, 2015, pp. 60–65.
17. Emeakaroha VC, Cafferkey N, Healy P, Morrison JP. A cloud-based IoT data gathering and processing platform. In: Proceedings of 3rd international conference on future internet of things and cloud, 2015, pp.50–57.
18. B. Kantarci, H.T. Mouftah
Sensing services in cloud-centric internet of things: a survey, taxonomy and challenges, Proceedings of IEEE ICC (2015), pp. 1865-1870
19. Alessio Botta, Walter de Donato, Valerio Persico, Antonio Pescapè. On the integration of cloud computing and internet of things. In: Proceedings of international conference on future internet of things and cloud, pp. 23–30, 2014.
20 G. Suciu, A. Vulpe, O. Fratu, V. Suciu
M2M remote telemetry and cloud IoT big data processing in viticulture
Proceedings of IEEE IWCMC (2015), pp. 1121–1171
21. L. Wang, R. Ranjan, Processing distributed internet of things data in clouds, IEEE Cloud Computing (2015), pp. 76-80

22. Nastic S, Sehic S, Le DH, Truong H, Dustdar S. Provisioning Software-defined IoT Cloud Systems. In: Proceedings of international conference on future internet of things and cloud, 2014, pp. 288–295.
23. Nastic S, Copil G, Truong H, Dustdar S. Governing elastic IoT cloud systems under uncertainty. In: Proceedings of IEEE 7th international conference on cloud computing technology and science, 2015, pp. 131–138.
24. H. Truong, S. Dustdar, Principles for engineering IoT cloud systems IEEE Cloud Computing (2015), pp. 68-76
25. S.W. Kum, J. Moon, T. Lim, J. Park, A novel design of IoT cloud delegate framework to harmonize cloud-scale IoT services, Proceedings of IEEE ICCE (2015), pp. 247-248
26. Celesti A, Fazio M, Giacobbe M, Puliafito A, Villari M. Characterizing cloud federation in IoT. In: 30th international conference on advanced information networking and applications workshops, 2016, pp. 93–98.
27. A. Taherkordi, F. Eliassen
Scalable modeling of cloud-based IoT services for smart cities
Proceedings of first IEEE international workshop on context-aware smart cities and intelligent transport systems (2016)
28. Fortino G, Guerrieri A, Russo W, Savaglio C. Integration of agent-based and cloud computing for the smart objects-oriented IoT. In: Proceedings of IEEE 18th international conference on computer supported cooperative work in design, 2014, pp. 493–498.
29. Survey of IoT Cloud Platforms market [Електронний ресурс] – Режим доступу до ресурсу: https://www.profitbricks.com.
30. Watson Internet of Things Platform [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/internet-of-things/spotlight/watson-iot-platform.

31. SAP Cloud Platform Internet of Things [Електронний ресурс] – Режим доступу до ресурсу: https://cloudplatform.sap.com/capabilities/iot.html.
32. AWS IoT Core [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/iot-core/.
33. Azure IoT Suite [Електронний ресурс] – Режим доступу до ресурсу: https://www.microsoft.com/en-us/internet-of-things/azure-iot-suite.
34. Безверхий А.І., Полякова Н.П., Попівщий В.І., Скрипник І.А., Оформлення і захист курсових, дипломних та кваліфікаційних робіт./ А. Безверхий, Н. Полякова, В. Попівщий, І. Скрипник – Запоріжжя ЗДІА 2007. – 51.с.
35. Кожемякін Г. Б. ОХОРОНА ПРАЦІ ТА ТЕХНОГЕННА БЕЗПЕКА Методичні вказівки до виконання розділу магістерських робіт для студентів ЗДІА всіх спеціальностей денної та заочної форм навчання / Г. Б. Кожемякін, В. Г. Рижков, К. В. Бєлоконь. – Запоріжжя: ЗДІА, 2012. – 49 с.