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

Радіотехнічний факультет
(повна назва інституту/факультету)

Кафедра радіоконструювання та виробництва радіоапаратури
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ С. А. Ільєнко
(підпис) (ініціали, прізвище)
"16" червня 2018р.

Магістерська дисертація

за спеціальністю 172 Телекомунікації та радіотехніка
(код і назва спеціальності)

за спеціалізацією Інтелектуальні технології мікросистемної радіоелектронної техніки

на тему: Цифрового синтезу для інтернету речей

Виконав (-ла): студент (-ка) 6_ курсу, групи РТ-371мп
(шифр групи)

Рибалко Владислав Петрович
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к.т.н., доц. Новіков Ю. Ф.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з охорони праці к.т.н., доцент Каштанов С.Ф.
(назва розділу) (посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент інженер, ТОВ "Інтелектуальні комунікації Західного" Є. В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

Київ – 2018 року

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

Факультет радіотехнічний

Кафедра радіоконструювання та виробництва радіоапаратури

Рівень вищої освіти – другий (магістерський)

Спеціальність 172 – телекомунікації та радіотехніка

Спеціалізація інтелектуальні технології мікросистемної радіоелектронної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Є.А. Нелін
(підпис) (ініціали, прізвище)
«__» _____ 20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Рибалка Владислав Ігорович
(прізвище, ім'я, по батькові)

1. Тема дисертації Цифрова система для інтернету речей,

науковий керівник дисертації к.т.н. доц. Адаменко Ю.Ф.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «06» 11 2018 р. №4094-с

2. Термін подання студентом дисертації 30.11.2018

3. Об'єкт дослідження архітектура рішення для інтернету речей

4. Предмет дослідження цифрова система для інтернету речей

5. Перелік завдань, які потрібно розробити аналіз архітектури існуючих систем; аналіз протоколів передачі даних та їх порівняння; вибір мови програмування; створення концепції цифрової архітектури інтернету речей.

6. Орієнтовний перелік ілюстративного матеріалу презентація орієнтована на слайдів

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст.в.дос. Іванов С.В.		

9. Дата видачі завдання 03.09.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Аналіз джерел інформації за напрямком дослідження	03.09.18 – 21.09.18	
2.	Теоретичні обґрунтування, розробка архітектурної характеристики проектів-аналогів	24.09.18 – 12.10.18	
3.	Охорона праці та розробка сценарію проекту	15.10.18 – 02.11.18	
4.	Розробка на основі розробленої архітектури	05.11.18 – 23.11.18	
5.	Представлення результатів магістерської дисертації	26.11.18 – 18.12.18	
6.	Захист МД	21.12.2018	

Студент (підпис) В.Р. Рибалка (ініціали, прізвище)

Науковий керівник дисертації (підпис) І.Г. Арапенко (ініціали, прізвище)

РЕФЕРАТ

Обсяг пояснювальної записки магістерської дисертації (МД) становить 81 сторінок, які включають в себе 5 розділів, 18 ілюстрацій, 14 таблиць, 1 додаток і 34 бібліографічних найменувань за переліком джерел посилань.

Ключові слова: інтернет речей, архітектура інтернету речей, mqtt, coap, nb-iot, docker.

Актуальність теми дослідження. Протягом останніх десятиліть інформаційні та комунікаційні технології намагалися постійно збільшувати кількість інтернет-пристроїв. Окрім традиційних комп'ютерів та мобільних пристроїв, — це пристрої, що варіюються від домашньої або побутової техніки, промислового обладнання та автоматики, охорони здоров'я, транспорту, енергії, будівель, міст та людей. Додавання інших девайсів, які традиційно були поза мережею інтернет, стало можливим завдяки технологічному прогресу за допомогою обладнання та нового програмного забезпечення. Розроблюване архітектурне рішення полегшить розробку нових систем та мереж інтернету речей.

Мета та завдання. Метою МД є уніфікація архітектурного рішення для інтернету речей, що буде використано кінцевим користувачем.

Об’єкт та предмет роботи. Об’єктом робти є архітектурне рішення.

Предметом виступає уніфікація системи інтернету речей.

Наукова новизна одержаних результатів: проведена уніфікація архітектурного рішення, що дозволяє використання такого для побудови проектів різного призначення та інтеграції з існуючими.

Практичне значення одержаних результатів: уніфікована система спрощує створення систем інтернету речей для розробників та дозволяє просту інтеграцію з існучими такими системами.

ABSTRACT

The master thesis consists of 81 pages, include 5 sections, 18 illustrations, 14 tables, 3 annex and 21 bibliographic titles.

Key Words: internet of things, architecture for internet of things, mqtt, coap, nb-iot, docker.

Relevance of the topic. Over the past decades, information and communication technologies have been trying to steadily increase the number of Internet devices. In addition to traditional computers and mobile devices, these are devices that range from home or home appliances, industrial equipment and automation, healthcare, transport, energy, buildings, cities, and people. Adding other devices that were traditionally out of the Internet has become possible thanks to technological progress with the help of hardware and new software. An architectural solution has been developed that facilitates the development of new systems and Internet things.

The purpose of research. The purpose is to unify the architectural solution for the Internet of things, which will be used by the end user.

Object of research: the architecture solution for internet of things.

Subject of research: unified architecture for internet of things.

Scientific novelty of the obtained results. Unification of the architectural solution has been made, which allows the use of such for construction projects of different purposes and integration with existing ones.

The practical value of the results obtained. The unified architecture simplifies the development of systems for the Internet of things for developers and allows for easy integration with the existing similar systems.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до магістерської дисертації

на тему: Уніфікована система для інтернету речей

Київ — 2018 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

CoAP — Constrained Application Protocol

D2D — Device-to-Device

DL — Downlink

DTLS — Datagram Transport Layer Security

IoT — Internet of Things

IP — Internet protocol

H2H — Human-to-Human

M2M — Machine-to-Machine

MQTT — Message Queue Telemetry Transport

NB-IoT — Narrowband-IoT

PaaS — Planform as a Service

REST — Representational State Transfer

SaaS — Software as a Service

SMS — Short Message Service

SOA — Service orienter architecture

SSL — Secure Sockets Layer

TSL — Transport Layer Security

UL — Uplink

WSN — Wireless Sensor Network

ВСТУП

Фраза інтернет речей (далі IoT) вперше була використана Кевіном Ештоном у 1999 році (Ashton, 2009), коли він виступив з доповіддю на Procter & Gamble. У своєму виступі він стверджував, що не тільки люди повинні генерувати та створювати дані, але комп'ютери та інші вбудовані пристрої повинні мати можливість збирати власну інформацію шляхом відстеження або взаємодії з їх внутрішніми станами або зовнішніми оточеннями. По суті, введення інтернету речей поширюватиме традиційний інтернет, зробивши мережеві з'єднання більш релевантними та цінними, ніж будь-коли раніше, а також додати нове значення в інформаційно-комунікаційну техніку. Коротко кажучи, IoT можна описати більш трансформаційно, ніж традиційний інтернет, який вплине на спосіб життя людей.

Інтернет речей — це ідея зробити фізичні об'єкти інтелектуальними, обмінюватися інформацією та приймати рішення в режимі реального часу. IoT забезпечує автоматичну взаємодію між інтелектуальними системами для обміну даними для значущої аналізу часу. Інтернет речей — це одна з найбільших технологій, що розвивається, IoT дасть нове напрям світового інтернету та непередбачуваний розвиток бізнесу та прибуток. Наприклад, галузі промисловості та промислової автоматизації додають розширений сенсор для декількох об'єктів для надсилання та отримання даних, що призводить до величезної кількості корисної інформації.

До сих пір багато хто плутає архітектуру IoT з архітектурою автоматизації, де головною завданням є отримання інформації з датчиків, а на їх основі здійснюється управління виконавчими механізмами.

IT-архітектура включає в себе дві, на перший погляд, несумісні речі. З одного боку — це велика кількість периферійних пристроїв з малими обчислювальних потужностями, низьким енергоспоживанням, високою швидкістю реакції на події. З іншого боку — хмарні сервери з високою обчислювальною потужністю для обробки великого масивів даних, їх

зберігання та класифікації, часто з елементами машинного інтелекту та аналітиків.

В силу різноманітності технологій, стандартизованого архітектурного рішення поки що не існує. Виробники таких пристроїв використовують різні апаратні платформи та мережеві протоколи, що робить мережу інтернет речей неоднорідною. Метою МД є уніфікація архітектурного рішення для інтернету речей, що буде використано кінцевим користувачем.

Для досягнення поставленої мети необхідно виконати такі завдань:

– аналіз архітектури існуючих систем;
– аналіз протоколів передачі даних та порівняння їх;
– вибір мови програмування;
– створення концепції уніфікованої архітектури інтеренету речей.

1 ЗАГАЛЬНІ ВІДОМОСТІ

1.1 Визначення інтернету речей

IoT — це мережа фізичних об'єктів або «речей», які взаємодіють один з одним. Це є розподілена електроніка, датчики та виконавчі пристрої з обчислювальною потужністю, програмним забезпеченням та мережевим з'єднанням, що дозволють користувачам стати її невід'ємною частиною. IoT пройшло багато розробок та міркувань з різними визначеннями на основі Інтернету. У своїй роботі Dr. Ovidiu Vermesan and Co. (д-р Вермезан та інші, 2011) описали цей термін, розглянувши інтернет яким він є зараз, як Інтернет Енергетики (IoE), Інтернет Медіа (IoM), Інтернет Людей (IoP) та Інтернет Сервісів (IoS). Cisco (Evans, 2012) вирішила монетизувати цей термін як Інтернет Всього (Internet of Everything), де його розглядали як систему, що складається з речей, де процес, дані та люди разом утворювали «мережу мереж». З точки зору Cisco — IoE з'єднує людей, процес, дані та «речі» разом, щоб сформувати мережу, придатну та корисну для допомоги у відстеженні «речей», а також для вирішення деяких глобальних проблем, таких як посуха, кліматичні зміни, джерела або питна вода, і голод.

IoT швидко розширюється, а області застосування включають розумні міста, інтелектуальну воду, інтелектуальне вимірювання, безпеку та надзвичайні ситуації, роздрібну торгівлю, логістику, промисловий контроль, інтелектуальне сільське господарство, розумне тваринництво, домашні та домашні господарства автоматизація та електронна охорона здоров'я. Однак, оскільки області застосування охоплюють різні середовища та пов'язані пристрої різноманітні, це робить IoT дуже неоднорідним, а отже, труднощами та бар'єрами, такими як зв'язність, керування живленням, складність, швидкий розвиток, безпека та якість обслуговування, які завжди пов'язані з стандартними викликами мережі бездротових сенсорних мереж (WSN), були перелічені Chase (Chase, 2013) як перешкода для розвитку IoT. Інші проблеми це конфіденційність, чутливість до участі, аналізі даних, візуалізація на основі

геоінформаційної системи (ГІС) та хмарних обчислень. Крім того, проблеми з підключенням до IoT також пов'язані з архітектурними і протокольними проблемами, які (Gubbi, Buyya, Marusic, & Palaniswami, 2013) вважають у своїй роботі відкритою проблемою.

Сьогодні виробники промислового обладнання стикаються, якщо не з усіма, але з більшістю вищезгаданих проблем. Тому для того, щоб IoT працював успішно та задовольняв прогнозований обсяг пристроїв, підключених до Інтернету до 2020 року, він повинен будуватися на відкритій, гнучкій технічній, програмній та мережевій платформах, здатних розвиватися і адаптуватися.

Отже, IoT включає в себе одночасно декілька явищ. Це безпосередньо прилади, які під'єднані до мережі та взаємодіють між собою. Також, це ще й спосіб підключення – M2M – машина-до-машини, без прямої участі людини/оператора. І найважливіше – дані, які можна збирати, зберігати, аналізувати та, в подальшому, використовувати для підвищення комфорту чи прийняття бізнес-рішень.

За думкою Роба Ван Краненбурга, інтернет речей являє собою «чотири-шаровий пиріг». Перший рівень пов'язаний з ідентифікацією кожного об'єкта. Другий рівень представляє разом із сервісом по обслуговуванню потреб споживача (можна розглянути як мережу власних «речей», наприклад, «розумний дім»). 3-й рівень пов'язаний з урбанізацією міського життя. Тобто, це концепція «розумного міста», де вся інформація, що стосується мешканців даного міста, зосереджується в конкретному житловому кварталі, в Вашому будинку та в сусідніх будинках. Нарешті 4-й рівень – сенсорна планета.

Іншими словами, інтернет речей можемо розглядати як мережу мереж, в якій невеликі мало суміжні мережі створюють більш масштабні мережі.

Звичайно, для взаємодії приладів необхідний канал зв'язку. Компанія Cisco провела детальний технічний аналіз, який показав, що IP цілком може бути адаптований до потреб мереж нового типу. В такому випадку IoT отримає наступні переваги: сумісність, масштабування і, найголовніше, єдину мову

спілкування. Дані переваги у свій час перетворили складну структуру окремих та загальнодоступних мереж в об’єднану глобальну комунікаційну систему, відому як Інтернет.

З тих пір, як промислові підприємства почали з'єднувати практично кожен пристрій та «річ», від сміттєвих баків до термостатів, у випадку збору даних у режимі реального часу, сьогодні бізнесу стало відомо, що реальне призначення IoT — це не просто збір та обробка даних, а це аналіз даних для розуміння того ж самого бізнесу.

Деякі визначення чи назви вживаються англійською мовою. Це пов’язано зі специфікою теми.

1.2 Галузі використання IoT технологій

В будь-якій сфері життя щось можливо оптимізувати, автоматизувати. Особливо активно IoT розвивається в аграрному секторі, логістиці, Smart City. Тобто, там де є потреба у віддаленому моніторингу стану об’єктів або при збору великих об’ємів даних з метою наступного аналізу. IoT дає можливість економити на обслуговуванні обладнання: датчики збирають інформацію про його стан, тому технічне обслуговування та ремонт відбувається саме тоді, коли це дійсно потрібно. Профілактика — завжи дешевше ремонту.

Інтернет речей також допомагає рятувати життя. Наприклад, нові Apple Watch спостерігають за серцевим ритмом у режимі реального часу. Вони допоможуть виявити аритмію та інші серцево-судинні захворювання.

Україна не стоїть в стороні від світових брендів. Великі міста поступово стають «розумними», флагмани цього руху — Київ та Львів. Хоча про комплексний підхід та масштабному запуску технологій Smart City говорити рано, аграрний сектор та логістика вже використовують рішення для IoT.

Міста. Міський транспорт з датчиками переміщень, мусорні баки з датчиками заповнення, планування маршрутів транспорту на основі даних про переміщення людей по місту, відеоспостереження, контроль за рівнем води у водоймах, датчики шуму та забруднення дають змогу зробити місто більш

комфортним та безпечнішим. А масиви даних, які збираються в результаті роботи датчиків, дають можливість владі міст краще розуміти потреби мешканців.

Аграрний сектор. В аграрному секторі інтернет речей знімає головну біль у агрономів відносно стану ґрунту. Датчики в землі фіксують показники: чи достатньо вологи, чи не пора удобрити рослини. Дрони фіксують ситуацію з висоти пташиного польоту та передають її інженерам. В оцінці стану ґрунтів, інженерам допоможуть нейронні мережі. Тепер не потрібно обходити всі поля на своїх двох аби проконтролювати врожай. Ось Нідерланди, будучи невеликою країною з високою густотою населення, являються досі одними із світових лідерів по вирощуванню продуктів харчування. Все це стало можливим дякуючи IoT технологіям.

Логістика. Доставка. Саме так, доставка будь-яких товарів з виробництва чи складу в магазини стала більш передбачуваною. Це важливо як для кінцевого користувача, так і для бізнесу. Логістичні компанії можуть відслідкувати де знаходиться автомобіль, або в який момент йому потрібно під'їхати до вантажу. Окрім цього, подібні системи активно застосовуються на водному транспорті, на вантажних судах.

Будівництво. «Розумні» лічильники самі фіксують, скільки енергії було затрачено в цьому місяці — не потрібно більше знімати показання й передавати їх по телефону. Деякі рішення для «розумного» дому навіть показують скільки кВт потрібно для живлення кожної лампочки чи побутового пристрою. Вони можуть бути увімкнені/вимкнені через смартфон. Наприклад, якщо встановити таку систему на дачі, то увімкнути опалення можна заздалегідь, клацнувши декільки кнопок у своєму мобільному, а приїхати у теплий дім. Подібні технології вже присутні на ринку України, але ще не є масовими.

Медицина. Медичні прибори, підключені до інтернету, дозволяють не тільки зекономити на лікуванні, попереджуючи серйозні ускладненні, але й рятувати життя. Дані збираються й поступають до лікаря практично в

автоматичному режимі, по ним можливо виявити причини хвороби. Система сповіщує лікаря, якщо аналізи пацієнта відхилені від норми тощо. Медичний інтернет речей в деяких країнах підтримується на державному рівні. Наприклад, державні органи Кореї намагаються зробити доступними прилади для людей похилого віку. А в Туречині інтергували програми співпраці між державою та бізнесом для боротьби з діабетом і його ускладненнями.

Системи безпеки. Системи відеоспостереження та охорони стають частиною життя як окремих бізнес одиниць, приватних будинків, так і цілих міст. Камери відеоспостереження з розпізнанням облич в метро — це не переказ фільму, це є реальність. І також є інтернетом речей.

Транспорт. Якщо розумні безпілотні автомобілі — це все ще технологія майбутнього, яка готується до масового штурму міст, то сучасний керований автомобіль з датчиками для аналізу стану системи і швидкої діагностики вже став реальністю. За словами Gartner, до 2020 року на дорогах світу буде більш ніж 250 млн авто, підключених до мережі. Тобто, кожна п’ята, Карл.

Рітейл. Маркети без касирів, камери, що розпізнають емоції покупців. Віртуальна та доповнена реальність, яка дозволяє дізнатися про продукт більше. Всі ці технології вже існують та рано чи пізно вони дійдуть і до нас. А поки що — кас самообслуговування все більше у сучасних супермаркетах.

1.3 Проблеми сучасного інтернету речей

1.3.1 Стандартизація

Насьогодні, галузі IoT не вистачає стандартизації, і для підключення одного «розумного» пристрою може використовуватися одна технологія, для іншого — інша. Це як ніби купуєш техніку, призначену для США або Китаю, до себе додому в Україну. Начебто все добре, але в розетку її підключити не зможеш — вилки різні. Або як старі мобільні телефони, де були абсолютно різні роз'єми для зарядки і навушників. До ери microUSB при зміні бренду телефону доводилося міняти всю периферію, наприклад, купувати ще одну зарядку, щоб вона була в офісі.

Виробники пристроїв для інтернету речей використовують різні стандарти підключення, що може стати проблемою, якщо купувати їх несистемно. Для одних тільки пристроїв «розумного будинку» є свій стандарт в Європі (Z-wave) і США (Zigbee). Нідерланди і Сінгапур побудували національні мережі для інтернету речей в стандарті LoRa, який працює на низькій частоті і вимагає мінімуму базових станцій для покриття великих територій.

Передбачається, що ідеальним стандартом зв'язку для інтернету речей стане 5G, але про реальні комерційних продуктах на базі цієї технології можна буде говорити тільки через кілька років. Поки що пристрої погано сумісні між собою, і до впровадження рішень потрібно підходити комплексно, щоб не вийшло, що були викинуті величезні гроші, а результатом стала тільки головний біль і ніякої оптимізації.

Стандартизація в IoT покликана знизити бар'єри для входу нових постачальників сервісів і користувачів, служить для поліпшення взаємодії різних додатків і сервісів, а також для забезпечення кращої якості продуктів або сервісів вищого рівня. Достатня координація зусиль в процесі стандартизації забезпечить пристроїв і додатків з різних країн можливість обмінюватися інформацією.

Крім того, рекомендується розробити і галузеві стандарти або інструкції для спрощення інтеграції різних сервісів при впровадженні інтернету речей в промисловість.

1.3.2 Безпека

Дійсно, питання безпеки в сфері інтернету речей стоїть гостро. Розрізненість і відсутність стандартів грає на руку і кіберзлочинцям. Близько року тому ботнет Mirai показав, на що здатні роутери, камери відеоспостереження та навіть «розумні» няньки, об'єднані для проведення DDoS-атак. Пристрої IoT уразливі, до їх захисту потрібно підходити з тією ж серйозністю, як і звичайного комп'ютерного обладнання.

Як відзначають IT-експерти в DEAC, при будь-якій роботі з великими даними виникає питання їх безпеки і збереження, тому DDoS та інші навмисні вірусні атаки не є єдиним ризиком. Розміщення, зберігання і подальша обробка даних з розумних пристроїв вимагає великих обчислювальних ресурсів і підлаштовуватися потужностей «заліза».

Багато з існуючих сьогодні технологій доступні для побутового використання, але не підходять для промислового застосування, в яких пред'являються підвищені вимоги з безпеки. Існуючі технології шифрування, запозичені з WSN (бездротової сенсорної мережі) або інших мереж, повинні бути ретельно перевірені перед їх використанням для захисту інформації при реалізації інтернету речей. Так як IoT дозволяє багато повсякденні речі відслідковувати, моніторити і пов'язувати, значна кількість особистої та персональної інформації може збиратися автоматично.

Захист приватності в середовищі інтернету речей стане більш серйозною, ніж в традиційному середовищі ІКТ, так як кількість векторів атак на «речі» IoT, мабуть, буде набагато більше. Наприклад, монітор здоров'я буде збирати дані пацієнта, такі як частота серцевих скорочень і рівень цукру в крові, а потім відправляти інформацію безпосередньо в кабінет лікаря по мережі. При цьому вона може бути вкрадена або зламана.

Інший приклад — біодатчик, який використовується в харчовій промисловості. Він може застосовуватися для моніторингу температури і бактеріального складу продуктів харчування, що зберігаються в холодильнику. Коли щось псується, дані про це відправляються в компанію через мережу. Однак така інформація повинна бути строго конфіденційною, щоб захистити репутацію харчової компанії.

Слід зазначити, що деякі питання, такі як визначення конфіденційності в IoT її юридичне тлумачення, як і раніше чітко не визначені. Незважаючи на те, що вже існують мережеві технології безпеки, для забезпечення основ конфіденційності та безпеки в IoT належить виконати ще багато роботи.

2 ТЕХНОЛОГІЇ ІНТЕРНЕТУ РЕЧЕЙ

2.1 Мови програмування

Як вже зазначалось у попередньому розділі, IoT оточує нас майже всюди, терпляче очікуючи команд. За результатами досліджень Ian Skerrett — лідерами являються Java, C, JavaScript та Python. І це не дивно, адже вони є

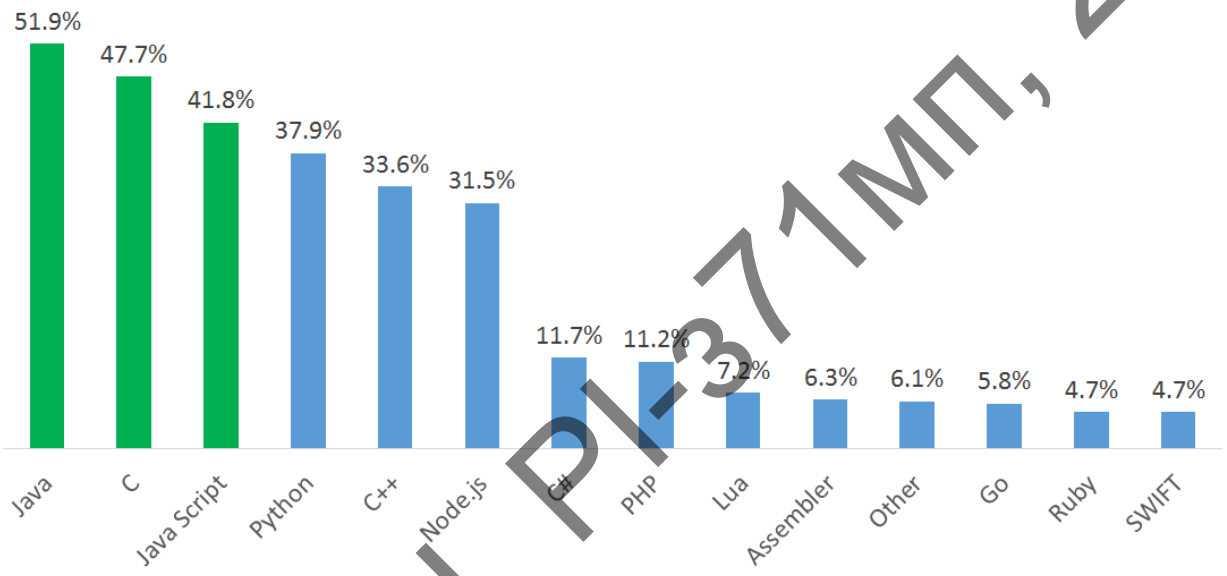

Рисунок 2.1 — Рейтинг мов програмування за 2015-2017 роки [10]

найпопулярнішими мовами програмування на сьогодні, як показано на рис 2.1.

То яку ж мову використовути для створення IoT рішення? Проста відповідь у тому, що «розумні речі» розуміють ті самі мови програмування, що і настольний комп'ютер. «Raspberry Pi — це фактикно повноцінний комп'ютер», — сказав Ian Skerrett, віце-президент маркетингу в Eclipse Foundation [10].

Немає жодної причини, чому ми не можемо вибрати мову програмування для IoT тим самим шляхом, як і для будь-якого іншого проекту. Коли прийде час, код буде відправлений для компіляції на сервер, немає різниці на який. Сервер, в свою чергу, буде підключатись до сенсорів та хабів, зазвичай це якась мікросервісна архітектура, яку описано в наступному розділі. Потім, отримані дані зберігаються до стандартних баз даних.

Перша в рейтингу — Java. І це не дивно, так як Java відома як «пиши раз, запускай будь-де», одна з перших мов програмування для нестандартних обчислень.

Переваги Java добре відомі. Розробники можуть створювати та налагоджувати код на своєму робочому місці, а потім перенести його на будь-який чіп за допомогою віртуальної машини Java. Це означає, що код може працювати не тільки на тих місцях, де JVM є загальними (сервери та смартфони), але і на найменших машинах.

Java ME або мікро-видання було доступно на телефонах та інших вбудованих пристроях, оскільки специфікація була затверджена в 2000 році. Це дозволило зберегти простір з дуже обмеженим набором бібліотек класів та іншими інструментами. Сьогодні основна увага приділяється Java SE Embedded, яка набагато ближче до можливостей стандартної версії. Розробники можуть використовувати останні функції платформи Java 8, а потім перенести свій код на менший, вбудований пристрій.

Більша частина економії обчислювальних ресурсів за допомогою Java SE Embedded є наслідком вилучення класів, необхідних для відображення інформації, коли машини можуть бути налаштовані без проблем, без монітора чи клавіатури. Весь зв'язок проходить через мережу.

Другою в рейтингу найпопулярніших є C. Синтаксис забарвлений пунктуацією, і існує мільйони різних помилок, які ви можете зробити, але мова все ще є першочерговим вибором для багатьох програмістів, які пишуть для найнижчого рівня програмного забезпечення, найближчого до апаратного забезпечення. Мова не приховує від вас нічого, а це означає, що ви можете грати з кожною частиною коду, щоб мати найкращу продуктивність з пристрою. Кожен біт можна перевернути. Кожне значення в стекі доступне. Просто не робіть помилки, тому що є декілька мереж безпеки.

Victor Berrios є головним технічним співробітником в ZigBee Alliance, групою, яка підтримує стандарт ZigBee, який зв'язує невеликі пристрої. «З того, що ми бачимо на ринку, C залишається вибором для обмежених

пристроїв», — сказав він. Вони, як правило, не включають в себе комерційну ОС, а базовий тип планувальника завдань управління ресурсами, який також кодується в C.

Більш продвинуті або більші пристрої з повноцінними операційними системами все ще використовують велику кількість коду C, але Berrios сказав, що інші мови, такі як Java, починають використовуватися так само часто. Коли смартфон постачається з Apple, більша частина програмування все ще виконується в Objective C, але це, мабуть, буде поступово замінено на Swift.

Python починав свою історію як скриптова мова, щоб склеювати один з одним справжній код. Але він все більше використовується як основна мова для багатьох розробників. Коли невеликі пристрої мають достатньо пам'яті та обчислювальної потужності, розробники можуть вільно вибирати мову, що полегшує їх життя, і це все частіше стає Python.

Kinman Covey, розробник мікроконтролерів, стверджує, що Python легко вивчати і підтримує цю мову велика, корисна спільнота. Синтаксис чистий і простий, що привертає більшу кількість програмістів. Мова часто є першим вибором для соціологів та біологів, наприклад. Коли вони потребують розумного пристрою в лабораторії, вони раді використати мову, яку вони знають, Python.

«Python є вибором для одного з найпопулярніших мікроконтролерів на ринку, Raspberry Pi», — сказав Covey. Більша частина навчальної літератури написана на Python, і багато шкіл використовують платформу для навчання комп'ютерних програм. Якщо проект є відносно простим і немає великих обчислювальних вимог, можна створити ефективні інструменти з тих самих плат і бібліотек, які використовуються в початкових школах.

Є також версії, розроблені, щоб бути ще меншими. Плата MicroPython і пакет програмного забезпечення — це невеликий мікроконтроллер, оптимізований для запуску Python на невеликій платі, що становить лише кілька квадратних дюймів [10,13].

2.2 HTTP протокол

HTTP (hypertext transfer protocol) — це найпоширеніший протокол передачі даних по всій мережі інтернет. HTTP був стандартизований Internet Engineering Task Force (IETF) в співпраці з World Wide Web Consortium (W3C) (MIT). HTTP працює за технологією клієнт-серверній передачі повідомлень, де клієнт запитує Hypertext Markup Language (HTML) сторінку з серверу, а сервер, в свою чергу, відповідає саме HTML сторінкою.

Як проілюстровано в таблиці 2.1, HTTP базується на TCP, який може використовувати сокети для передачі даних. З'єднання між клієнтом та сервером розпочинається з сокет з'єднання на порту 80, котрий є зарезервованим портом для HTTP з'єднання з сервером. Коли з'єднання встановлене, це означає що сервер приймає запити від клієнта у формі HTML сторінки чи інших об'єктів.

Таблиця 2.1 Програмні протоколи IoT

Протокол	**Транспортний протокол**	**Передача повідомлень**	**WAN (2G, 3G, 4G)**	**Живлення**	**Ресурс**	**Безпека**
HTTP/REST	TCP	Request/Responce	OK	Високе	100Ks/RAM Flash	Низька
MQTT	TCP	Pub/Sub Request/Response	OK	Середнє	10Ks/RAM Flash	Середня
CoAP	UDP	Request/Responce	OK	Низьке	10Ks/RAM Flash	Середня

Коли з'єднання встановлене, HTML сторінки та об'єкти почитають пересилатись між клієнтом та сервером. Після того як запиз завершився, TCP завершує з'єднання між клієнтом і сервером, також очищає пам'ять таким чином, що попередній запит до сервера тепер видалений. Найбільш широко використовувані чотири HTTP запити, це — GET, POST, PUT та DELETE. GET запит відображає веб-сторінку та об'єкти кожного разу як проходить

такий запит до серверу. POST та PUT методи використовуються для модифікації серверного ресурсу, а DELETE — для видалення ресурсів, які не є потрібними [14].

2.3 REST

REST — це мова та незалежна архітектура програмного забезпечення для операційної системи для розробки мережевих додатків та розповсюдження HTTP-системи для з'єднання машин разом. REST — клієнт-сервер який не має постійного стану, кешований, точка-точка інтерфейс і розроблений як легка система. Зв’язок розпочинається, коли клієнт надсилає повідомлення у формі запиту на сервер, а сервер відправляє назад клієнту у формі відповіді, вказуючи, чи надійшов запит, надісланий клієнтом, чи була помилка. З REST зв'язок між пристроями в хмарі можливий через TCP / IP, де HTTP використовується для підключення до всесвітньої мережі (World Wide Web).

2.4 CoAP

CoAP — це протокол передачі даних, що базується на Client-to-Server моделі зв’язку. Він подібний до HTTP та був стандартизований в межах IETF, Constrained RESTful Environments (CoRE) робочої групи. Він призначений для пристроїв, обмежених за ресурсами потужності, пам’яті чи за передачею даних. HTTP є основним протоколом, оскільки зв’язок між клієнтом і сервером надто «важкий» для подібних пристроїв [15]. CoAP був розроблений для вирішення обмежень, що накладає HTTP на датчики, серсори та інші пристрої з низькою потужністю.

Дизайн моделі CoAP еквівалентний до HTTP клієнт-серверної моделі, але більша частина реалізації призначена для M2M (машина до машини) з’єднання. CoAP не підтримує передачу через TCP, але він знаходиться над рівнем User Datagram Protocol (UDP). Він використовує UDP трансляцію та мультитрансляцію для передачі пакетів та взаємодії між клієнтом та сервером асинхронно. Оскільки UDP встановлює бездротове з’єднання, зв’язок CoAP

також є бездротовим, та може бути використаний разом з різними пакетними протоколами комуцікаціями, такими як SMS.

Більше того, пристрої пов'язані через CoAP мають змогу дізнатись одне про одного, передавати одне одному дані. CoAP також має підтримує методологію спостереження за змінами стану. Це така модель передачі стану, яка дозволяє безперервно отримувати дані від сервера. Це може бути корисно наприклад в медичній сфері, де дані з сенсорів напряму надходять на загальний монітор про пацієнта у режимі реального часу. CoAP — це асинхронний обмінник повідомленнями, працюючи як модель спостереження-сповіщення. Подібний до HTTP, клієнт надсилає запити типу GET в режимі спостереження (observable mode) аби дізнатись що ж там на сервері. Сервер відправляє у відповідь повідомлення кожного разу, коли стан даних змінився.

CoAP також відомий як Conditional Observer (умовний спостерігач, мається на увазі «спостерігає за умов») або модель на основі подій. Це означає, що клієнт буде повідомлений від сервера тільки тоді, коли певні дії відбудуться. Оскільки повідомлення не будут отримані за будь-яких змін, а тільки за тих, на які потрібні нам — буде збережена енергія батарейки чи будь-якого іншого джерела живлення. Наприклад, датчик температури оновлюється кожну секунду, навіть якщо значення температури не змінилось. За допомогою CoAP клієнт отримає тільки повідомлення про зміну значення, а не кожну секунду одну й ту ж саму температуру. Ще одна особливість CoAP — це підтримка проксі, тобто клієнт може запитати дані з CoAP серверу через HTTP.

2.4.1 Типи повідомлень CoAP

Для CoAP визначено 4 типи повідомлень: Confirmable (потребує підтвердження), Non-Confirmable (не потребує підтвердження), Acknowledgement (підтвердження), Reset (скинути) [15].

Коли клієнт надсилає запит на сервер з підтверджуваними повідомленнями (CON), він вимагає, щоб кінцевий прийом підтверджував (ACK) повідомлення з тим самим ідентифікатором повідомлення. Ця передача між клієнтом і сервером зазвичай використовується, коли потрібна надійна доставка даних. Повторна передача даних відбувається після закінчення часу очікування для закінчення ACK, і він повторить кола до отримання ACK з ідентифікатором повідомлення. На рис 2.1 показана надійна передача повідомлень між клієнтом і сервером.

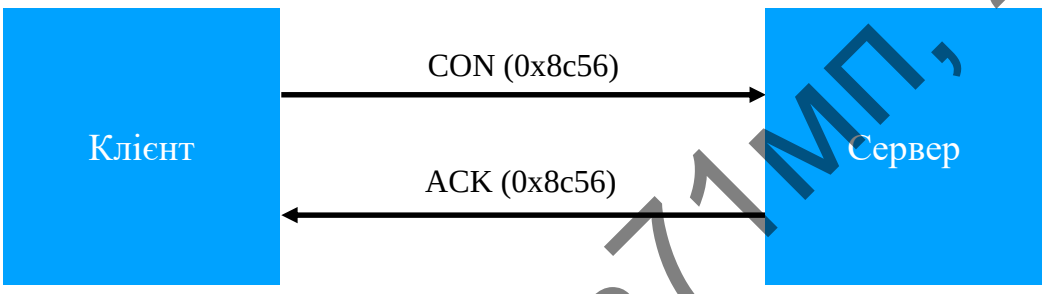

Рисунок 2.1 — Передача підтверджуваних повідомлень CoAP

Техніка передачі даних, що не підлягає підтвердженню, не вимагає ACK, і вона є ненадійною. На рис 2.2 цей тип технології обміну даними використовує тип NON-повідомлення, які містять ідентифікатор повідомлення для контролю передачі. Це найбільш поширений обмін даними, але існує можливість втрачених даних або отриманих не в тому порядку.

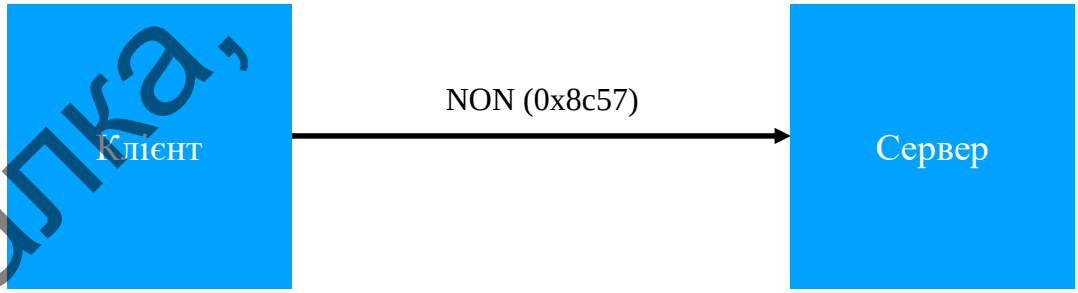

Рисунок 2.2 — Передача не підтверджуваних повідомлень CoAP

Крім того, як показано на рис 2.1, повідомлення ACK з ідентифікатором повідомлення надсилається клієнту (відправнику) з сервера (одержувача), до якого приходить певне підтверджене повідомлення (CON).

CoAP підтримує також повідомлення про помилки. Коли клієнт надсилає запит, використовуючи тип CON або NON-тип повідомлення, він одразу отримує повідомлення ACK, якщо це повідомлення, що підтверджується. ACK містить відповідне повідомлення про успішне або невдале виконання доставленого повідомлення.Знову ж таки, коли сервер отримує запит на повідомлення CON, і він не може негайно відповісти на запит, він надсилає порожній ACK, щоб клієнт повторно відправив повідомлення після закінчення певного проміжку часу. Однак, нова конфігурація надсилається клієнту кожного разу, коли сервер готовий відповісти на повідомлення, а клієнт відповідає за запитом ACK, щоб підтвердити повідомлення CON з сервера. На рис 2.3 показані окремі відповіді, коли клієнт використовував запит GET для запиту температури від клієнта.

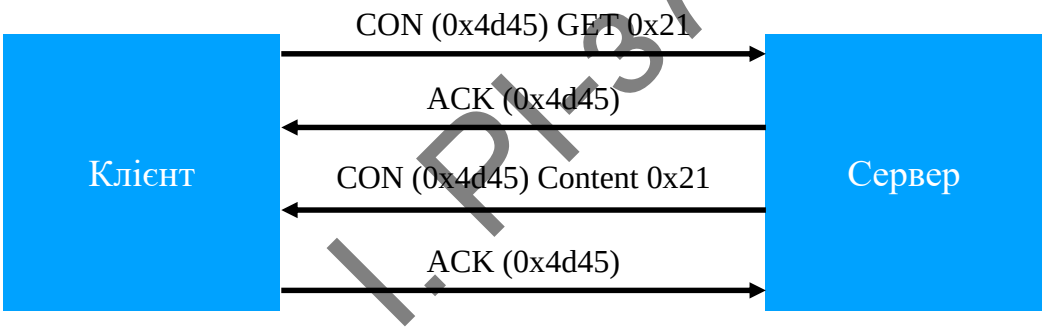

Рисунок 2.3 — CoAP CON повідомлення запиту з різними відповідями

Повідомлення ACK з підтверджуваних повідомлень не вказують на успіх або невдачу будь-якого запиту, але повідомлення ACK може містити негативну відповідь.

Reset, яке також називається «Негативне підтвердження» (NACK), — це повідомлення про помилку, яке надсилається з приймаючого кінця (сервер), щоб повідомити відправника (клієнта) про те, що певне повідомлення втрачено або приймач не зміг обробити повідомлення. Іншими словами, для відхилення помилок та невідомих повідомлень, коли отримано певні повідомлення (Confirmable або Non-Confirmable), але для відсутності певного контексту для правильної обробки, надсилається повідомлення про скидання [15].

2.4.2 Формат повідомлень CoAP

CoAP базується на обміні компактними повідомленнями, які за замовчуванням передаються через UDP. Повідомлення CoAP кодуються у простій двійковій формі, і це 4-байтовий заголовок з фіксованим розміром, після чого можливі додаткові розширення, такі як значення параметра Token Value для змінної-довжини, послідовність з нульовим або більше параметрами CoAP у параметрі Type-Length-Value (TLV) і необов'язкове корисне навантаження, яке займає решту датаграми. У таблиці 2.2 показана структура формату повідомлень CoAP. 4-байтовий заголовок складається з Version (Ver, 2-bit), Type (T, 2-біт), Length Token (TKL, 4-біт), Code (8-bit) та ідентифікатора повідомлення (8-біт).

Таблиця 2.2 Формат CoAP повідомлення

Version (2 біт)	Type (2 біт)	Code (8 біт)	Message ID (16 біт)
Token 0-8 байт (якщо є)			
Options (якщо є)			
Payload (якщо є)			

2-бітовий непідписана числовий файл версії вказує на CoAP номер версії та для RFC-7252 CoAP специфікації приймає значення 1 (01 в бінарному форматі). Це означає, що кожне повідомлення повинно мати номер версії. В іншому випадку повідомлення з невідомим номером версії ігноруються. Всі інші значення зарезервовані для майбутніх версій.

Type (тип) також є 2-бітним цілим непідписаним заголовком, яке вказує, чи є повідомлення типу Confirmable, Non-Confirmable, Acknowledgment або Reset. Token Length — це 4-бітне ціле число без знаку в заголовку, яке вказує довжину поля токена змінної довжини, що становить від 0 до 8 байтів. Якщо число встановлено на 0, це означає, що немає варіантів, а корисне навантаження (якщо такий є) негайно слідує за заголовком. Однак, якщо число

більше 0, поле вказує кількість варіантів, які слід негайно слідувати за заголовком.

Код також є 8-бітовим цілим непідписаним числом в заголовку і розділений на підполя: 3-розрядний клас (найважливіші біти) і 5-бітні деталі (найменший значнимий біт). Клас може вказати запит, успішну відповідь, відповідь клієнтської помилки або відповідь на помилку сервера. Ідентифікатор повідомлення — це 16-бітне ціле непідписане значення в заголовку, і воно використовується для виявлення дублювання повідомлень та для співставлення повідомлень типу Acknowledgement/RESET до повідомлень типу Confirmable/Non-Confirmable.

Значення Token знаходиться поруч із заголовком і становить від 0 до 8 байт, як вказано полем Token Length у заголовку. Він використовується для співставлення запитів та відповідей. Однак 8-байтовий заголовок допомагає захистити атаки, такі як підроблення, і це правило, що всі повідомлення CoAP мають токени, навіть якщо вони мають нульову довжину.

Як зазначалось раніше, параметри CoAP можуть бути присутніми лише тоді, коли значення поля Token Length не є нулем. У полі опцій зберігається інформація, яка впливає на продуктивність та функціональність CoAP. Крім того, CoAP визначає кількість опцій, які можуть бути включені в повідомлення, і кожен параметр у повідомленні вказує номер опції, довжину опції та значення параметра. Деталі та вичерпний список опцій розробляються в документації RFC-7252. Корисне навантаження також є необов'язковим, і може бути доступним лише тоді, коли воно не має нульової довжини, і попередньо встановлено однобайтовим маркером корисного навантаження (0xFF), що вказує на кінець параметрів і початок корисної завантаження.

Дані про корисне навантаження розширюються від маркера до кінця датаграми UDP. Відсутність маркера корисного навантаження представляє собою корисне навантаження з нульовою довжиною, а наявність маркера, з іншого боку, з наступним корисним завантаженням нульової довжини, повинна оброблятися як помилка формату повідомлення. Крім того,

повідомлення запитів та відповідей від клієнта та сервера, відповідно, можуть містити дані про корисне навантаження. Вони також можуть бути передані разом з повідомленням, що підтверджується, і повідомлення, яке не підлягає підтвердженню.

2.4.3 Обмін повідомленнями між сервером та клієнтом

Обмін повідомленнями між кінцевими точками CoAP (Client-to-Server) виконується асинхронно. CoAP використовує протокол UDP для транспортування повідомлень. Він може бути ненадійним, що означає — повідомлення можуть прийти в неправильному порядку, можуть з’явитись дублікати або взагалі не дійти.

Однак, CoAP представляє простий надійний механізм, подібний до протоколу TCP, з наступними особливостями:

– проста зупинка та очікування надійності ретрансляції з експоненціальним відключенням для підтверджуваних повідомлень;

– детекція дублікатів для обох підтверджуваних та не підтверджуваних повідомлень.

Для передачі повідомлень через CoAP протокол використовується техніка запиту-відповіді.

2.4.4 Запити

Метод запиту для CoAP схожий на HTTP запит, має методи GET, POST, PUT, DELETE. GET метод використовується для отримання стану інформаційного ресурсу, який предоставлений як Uniform Resource Identifier (URI, однорідний ідентифікатор ресурсу). Інформація, така як значення сенсорів, наприклад температури, імена приладів або стан приладу — можуть бути отримані за допомогою GET запиту. POST та PUT методи є подібними, вони використовуються для створення нового ресурсу або коли ресурс

оновився. Запит DELETE методом використовується для видалення ресурсу по вказаному URI.

Як говорилося раніше, CoAP підтримує спостереження за змінами ресурсів, і це інакша форма запиту яка дозволяє клієнту спостерігати за станом ресурсу через певний період часу. Цей метод був спроектований тому, що методи GET, POST, PUT та DELETE не працювали належним чином, коли клієнт мав вести спостереження за ресурсом на сервері. Метод спостереження дозволяє вузлу сепаратора CoAP постійно надсилати сповіщення після того, як він отримав реєстраційне повідомлення від клієнта. Мета серверу полягає у тому, аби тримати спостерігача (клієнта) в курсі (оновлювати), повідомляючи йому останні значення ресурсів.

Коли спостерігач (клієнт) зацікавлений у оновленні даних з серверу, він надсилає реєстраційне повідомлення до серверу. Воно надсилається GET запитом зі значенням опції спостереження «0». Сервер, в свою чергу, додає клієнта до списку спостерігачів даного ресурсу і розпочинає розсилку повідомлень. Дані повідомлення містять значення у полі спостереження та використовуються для перевірки вимірювань (показань датчиків). У випадку, коли сервер не має змоги додати нового спостерігача до списку, від відправляє відповідь без значення у полі спостереження.

На рис 2.4, взятого із специфікації RFC7641, показано як клієнт реєструється на певному ресурсі та отримує повідомлення. В даному прикладі, клієнт є зацікавлений у спостереженні за значенням температури на сервері, та відправляє повідомлення реєстраційне повідомлення. Сервер додає клієнта

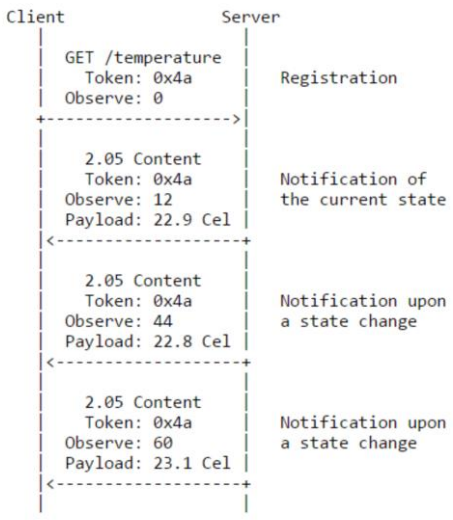

Рисунок 2.4 — Спостереження клієнтом ресурсу в CoAP [15]

(спостерігача) до своєї бази даних та розпочинає надсилати повідомлення до спостерігача. Коли клієнт більш не зацікавлений у значеннях температури, від надсидає повідомлення про зняття реєстрації зі значенням поля спостереження «1».

Ще один спосіб прининити спостереження — відхилити повідомлення, надіславши повідомлення скидання (Reset). Також, обмін повідомленнями між сервером та клієнтом може бути підтверджувальним та непідтверджувальним. У випадку підтверджувального (Confirmable) повідомлення, сервер буде очікувати значення підтвердження (Asknowledgement) від клієнта. Якщо після певного періоду часу, з декількома повторними передачами повідомлення, сервер не отримав підтвердження (Acknowledgement) від клієнта — сервер вирішить, що клієнт більш не зацікавлений у спостереженні за ресурсом та видалить його з списку спостерігачів.

2.4.5 Відповіді

Коли запит надісланий від клієнта до серверу, останній відповідає у відповідності до згенерованого токену клієнта (Generated Token). Відповідь буде ідентифікована полем Code в заголовку CoAP і матиме значення відповідно коду відповіді (Response Code, так званий статус-код). Розрізняють наступні статус-коди:

– Success 2.x.x — індикує про те, що клієнт успішно отримав повідомлення, зрозумів та прийняв.

– Client error 4.x.x — призначений для випадків, коли клієнт, має помилку. Цей код відповіді поширюється на будь-який метод запиту.

– Server error 5.x.x — відповідає випадкам, коли сервер усвідомлює, що він має помилку або не може виконувати запит. Ці коди відповіді можуть бути застосовані до будь-якого методу запиту.

2.4.6 Безпека CoAP

Як і в будь-якому сполученні між пристроями, безпека важлива, і вона не є виключенням для протоколу CoAP, який є стандартизований (ISO / IEC 20922) [16]. Однак, при розгляді безпеки будь-яких систем зв'язку існує три елементи, які слід розглянути. Це цілісність системи, аутентифікація та конфіденційність. Транспортний рівень захищеності Datagram (DTLS) RFC 6347 був розроблений як протокол безпеки для CoAP. Перш за все, CoAP використовує передачу датаграми, і DTLS може досягти вищезгаданих елементів захисту. Він добре підходить для захисту додатків та пристроїв, що чутливі до затримок, і має механізм переупорядкування повідомлень, які надходять не в тому порядку, повторну передачу втрачених повідомлень під час першого з'єднання (handshake) та розмірів повідомлень. Враховуюься і помилок під час дешифрування, але немає повідомлень про помилки та про завершення сеансу. Він також додає три реалізації: 1 пакет ретрансляції, два послідовних номерів присвоєння при handshake та виявлення трьох повторень.

DTLS має композицію із 2 шарів. Нижній шар, відомий як DTLS Record Protocol, забезпечує безпеку з'єднання та має 2 базові властивості:

– використовуючи симетричне шифрування, з'єднання є приватним;

– з'єднання є надійним через перевідку цілісності повідомлення.

Ці властивості або опції можуть використовуватись як поодинці, так і разом, або взагалі не використовуватись.

Верхній шар компонується з 3-х протоколів Alert, Handshake та Data. Протокол DTLS Handshake використовується для узгодження параметрів безпеки сеансу, використовуваного пізніше для захищеного зв'язку. Протокол попередження (Alert) DTLS можна використовувати в будь-який час під час Handshake процедури і до закриття сеансу, повідомляючи про фатальні помилки або попередження. Протокол даних (Data) DTLS-додатків складається з даних програм, які виконуються на рівні запису, фрагментовані, стиснуті та зашифровані на основі поточного стану з'єднання.

Більш того, за певних умов, Change Cipher Spec Protocol може замінити один із вище згаданих протоколів захисту DTLS. Протокол повідомлень Change Cipher Spec Protocol використовується аби повідомити Record Protocol, щоб захистити наступні записи, використовуючи з'єднувальний шифр та ключі. Рис 2.5 демонструє процедуру Handshake.

Рисунок 2.5 — Процедура Handshake DTLS [17]

CoAP призначений для простої взаємодії з протоколом HTTP для інтеграції з мережею інтернет під час виконання специфічних вимог, таких як підтримка багатоадресної передачі, дуже низькі накладні витрати та простота для обмежених середовищ. Таким чином, CoAP повністю сумісна з HTTP, але має нижчий надмірний рівень, підтримує багатоадресне та інше. Впровадження CoAP набагато простіше, ніж HTTP, оскільки він працює над UDP, а не використовує складний контроль над TCP. Таким чином, максимальний розмір пакету CoAP становить лише 1152 байти.

2.5 Протокол MQTT

Все ж таки, із розвитком промисловості збільшилась кількість приладів, які потрібно контролювати та отримувати від них різноманітні дані. Для вирішення проблем взаємодії великої кількості приладів та проблем об’єднання їх в одну мережу, на допомомогу ZigBee приходить протокол MQTT.

MQTT — це легкий, компактний та відкритий протокол обміну даними. Він створений для передачі даних на віддалених локаціяї, де потрібний невеликий розмір коду та є обмеження пропускних властивостях каналу.

Можна мовити, що MQTT був спеціально винайдений для концепції IoT. Тому розглянемо його детальніше. Також, існує версія протоколу MQTT-SN для сенсорних мереж. Дана версія призначення для вбудованих бездротових приладів без підтримки TCP/IP мереж, таких як ZigBee.

Основні особливості протоколу MQTT:

– асинхронний протокол;

– компактні повідомлення;

– робота в умовах нестабільного зв’язку на лінії передачі даних;

– підтримка декількох рівнів якості обслуговування (QoS);

– легка інтеграція нових приладів.

Протокол MQTT працює на програмному рівні поверх TCP/IP та використовує по замовчуванню порт 1883 (8883 при підключенні через SSL) [16].

2.5.1 Принцип роботи

Обмін повідомленнями в протоколі MQTT здійснюється між клієнтом (client), який може бути видавцем або підписником (publisher / subscriber) повідомлень, і брокером (broker) повідомлень. Видавець відправляє дані на MQTT брокер, вказуючи в повідомленні певну тему, топік (topic). Підписники можуть отримувати різні дані від безлічі видавців залежно від підписки на відповідні топіки.

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

– Connect — встановити з'єднання з брокером;

– Disconnect — розірвати з'єднання з брокером;

– Publish — опублікувати дані в топік на брокера;

– Subscribe — підписатися на топік на брокера;

– Unsubscribe — відписатися від топіка.

Через MQTT зручно обмінюватись повідомленнями, так як їх топіки мають певну семантику, зрозумілу більшості людей (human readable format). Топіки представляють собою символи з кодуванням UTF-8. Ієрархічна структура топіків має формат «дерева», що спрощує їх організацію та доступ до даних. Топіки складаються з одного або декількох рівнів, які розділені між собою символом «/» [16].

Приклад топіка в який датчик температури, розташований в спальній кімнаті публікує дані брокеру:

– «/home/living-space/living-room1/temperature».

Підписник може так само отримувати дані відразу з декількох топіків, для цього існують «wildcard». Вони бувають двох типів: однорівневі і багаторівневі. Для більш простого розуміння розглянемо в прикладах кожен з них:

– «/home/living-space/+/temperature».

Для його використання однорівневий wildcard застосовується символ «+». Наприклад, нам потрібно отримани значення з усіх спалень. Тоді «+» може набувати наступних значень:

– «/home/living-space/living-room1/temperature»;

– «/home/living-space/living-room2/temperature»;

– «/home/living-space/living-room3/temperature».

Для використовування багаторівневого wildcard використовують «#»:

– « /home/living-space/#».

Тоді, для отримання даних із датчиків по всьому дому, потрібно відправити повідомлення з наступними топіками:

– «/home/living-space/living-room1/temperature»;
– «/home/living-space/living-room1/light1»;
– «/home/living-space/living-room1/light2»;
– «/home/living-space/living-room1/humidity»;
– «/home/living-space/living-room2/temperature»;
– «/home/living-space/living-room2/light1».

2.5.2 Формат повідомлення

MQTT повідомлення складається з декількох частин: фіксований заголовок (присутній на всіх повідомленнях), змінний заголовок (присутній тільки в певних повідомленнях), дані або “навантаження” (payload, присутня тільки в певних повідомленнях). Струкрура MQTT повідомлення зображена на рис 2.6.

Рисунок 2.6 — Структура MQTT повідомлення [16]

Message Type — це тип повідомлення, наприклад: connect, subscribe, publish та інші.

Flags specific to each MQTT packet — ці 4 біти відведені під допоміжні маркери, наявність та значення яких залежать від типу повідомлення.

Remaining Length — являє собою довжину даного повідомлення (змінних заголовок + дані), може займати від 1 до 1 байта.

Усього в протоколі MQTT існує 15 типів повідомлень, які приведені у таблиці 2.3.

Таблиця 2.3 Список типів повідомлень у протоколі MQTT

Тип	**Значення**	**Напрямок**	**Опис**
Reserved	0000 (0)	-	Зарезервований
CONNECT	0001 (1)	К* -> С**	Запит клієнта на підключення до серверу
CONNACK	0010 (2)	К <- С	Підтвердження успішного підключення
PUBLISH	0011 (3)	К <- С, К -> С	Публікація повідомлення
PUBACK	0100 (4)	К <- С, К -> С	Підтвердження публікації
PUBREC	0101 (5)	К <- С, К -> С	Публикація отримана
PUBREL	0110 (6)	К <- С, К -> С	Дозвіл на видалення повідомлення
PUBCOMP	0111 (7)	К <- С, К -> С	Публікація закінчена
SUBSCRIBE	1000 (8)	К -> С	Запит на підписку
SUBACK	1001 (9)	К <- С	Запит на підписку прийнятий
UNSUBSCRIBE	1010 (10)	К -> С	Запит на відписку
UNSUBACK	1011 (11)	К <- С	Запит на відписку прийнятий
PINGREQ	1100 (12)	К -> С	PING запит
PINGRESP	1101 (13)	К <- С	PING відповідь
DISCONNECT	1110 (14)	К -> С	Повідомлення про відключення від сервера
Reserved	1111 (15)	-	Зарезервований

де К — клієнт, С — сервер.

Чотири старших біти першого байта фіксованого заголовку відведені під спеціальні маркери, зображені на рис 2.7.

Рисунок 2.7 — Маркери MQTT повідомлення

DUP — маркер дублікату, встановлюється тоді, коли клієнт чи MQTT брокер здійснює повторту відправку пакету (використовується в типах PUBLISH, SUBSCRIBE, UNSUBSCRIBE, PUBREL). При даному маркерові змінний заголовок повинен містити в собі MessageID (ідентифікатор повідомлення).

QoS — якість обслуговування (0, 1, 2).

RETAIN — при публікації даних з встановленим даним маркером, брокер зберігає його. При наступній підписці на цей топік, брокер в той же час відправляє повідомлення з цим маркером. Використовується тільки в повідомленнях з типом PUBLISH.

Заголовок може бути змінним. Він може бути присутнім тільки в деяких заголовках. У ньому передаються наступні дані:

– *Packet identifier* — ідентифікатор пакета, присутній у всіх типах повідомлень (крім CONNECT, CONNACK, PUBLISH (з QoS <1), PINGREQ, PINGRESP, DISCONNECT);

– *Protocol name* — назва протоколу (тільки в повідомленнях типу CONNECT);

– *Protocol version* — версія протоколу (тільки в повідомленнях типу CONNECT);

– *Connect flags* — прапори вказують на поведінку клієнта при підключенні [16].

2.5.3 Структура навантаження

Корисне навантаження може бути зчитане з повідомлення як на сервері, так і на клієнті. Формат навантаження зображений на рис 2.8.

Bit	7	6	5	4	3	2	1	0
Byte 8	User name	Password	Will Retain	Will QoS		Will Flag	Clean Session	Reserved

Рисунок 2.8 — Формат навантаження MQTT повідомлення

User name — при наявності цього маркера в «навантаження» має бути вказано ім'я користувача (використовується для аутентифікації клієнта).

Password — при наявності цього маркера в «навантаження» повинен бути вказаний пароль (використовується для аутентифікації клієнта).

Will Retain — при значенні «1», брокер зберігає у себе маркер Will Message.

Will QoS — якість обслуговування для Will Message, при встановленому прапорі Will Flag, Will QoS і Will retain є обов'язковими.

Will Flag — при встановленому цьому маркері, після того, як клієнт відключиться від брокера без відправлення команди DISCONNECT (у випадках непередбачуваного обриву зв'язку і т.д.), брокер сповістить про це всіх підключених до нього клієнтів через так званий маркер Will Message.

Clean Session — очистити сесію. При встановленому «0», брокер збереже сесію, всі підписки клієнта, а також передасть йому всі повідомлення з QoS1 і QoS2, які були отримані брокером під час відключення клієнта при його наступному підключенні. Відповідно при встановленій «1», під час наступного з'єднання клієнту буде необхідно заново підписуватися на топіки.

Session Present — застосовується в повідомленні з типом CONNACK. Якщо брокер приймає підключення Clean Session зі значенням «1» — він повинен встановити «0» в біт Session Present (SP). Якщо брокер приймає підключення Clean Session зі значенням «0», то значення біта SP залежить від того, зберігав чи брокер раніше сесію з цим клієнтом (якщо так, то в SP

виставляється «1» і навпаки). Тобто цей параметр дозволяє клієнту визначити чи була збережена брокером попередня сесія.

Connect Return code — якщо брокер з якихось причин не може прийняти правильно сформований CONNECT пакет від клієнта, то в другому байті CONNACK пакета він повинен встановити відповідне значення з нижчезазначених списку (таблиця 2.4).

Таблиця 2.4. Список повернених кодів при з'єднанні

Значення	**Повернене значення**	**Описание**
0	0x00 Connection Accepted	Підключення прийняте
1	0x01 Connection Refused, unacceptable protocol version	Брокер не підтримує версію протокола, що використовує клієнт
2	0x02 Connection Refused, identifier rejected	Client ID підключеного клієнта немає в списку дозволених
3	0x03 Connection Refused, Server unavailable	З'єднання встановлено, але MQTT клієнт недоступний
4	0x04 Connection Refused, bad user name or password	Неправильний логін чи пароль
5	0x05 Connection Refused, not authorized	Доступ з'єднання заборонено
6-255		Зарезервовано

Дані навантаження, що передаються в повідомленнях MQTT, визначаються в підключеному додатку. Розмір даних може бути вирахуваний шляхом зчитування *Remaining Length*, довжини змінного заголовку [16].

2.5.4 Рівні обслуговування

Як було зазначено вище, MQTT протокол підтримує три рівні якості обслуговування (QoS) при передачі повідомлення.

QoS 0 At most once. На цьому рівні видавець один раз відправляє повідомлення брокеру і не чекає підтвердження від нього, тобто відправив і забув (рис 2.9).

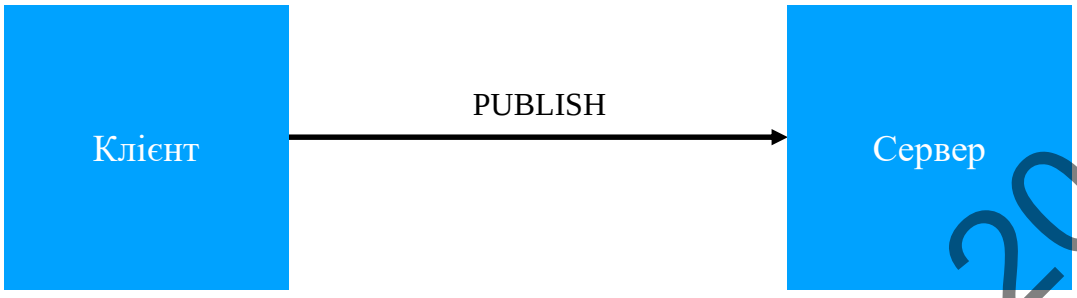

Рисунок 2.9 — Схема передачі повідомлення при QoS 0

QoS 1 At least once. Цей рівень гарантує, що повідомлення точно буде доставлено брокеру, але є ймовірність дублювання повідомлень від видавця. Після отримання дубліката повідомлення, брокер знову розсилає це хто підписався, а видавцеві знову відправляє підтвердження про отримання повідомлення. Якщо видавець не отримав PUBACK повідомлення від брокера, він повторно відправляє цей пакет, при цьому в DUP встановлюється «1». Рівень QoS 1 зображений на рис 2.10.

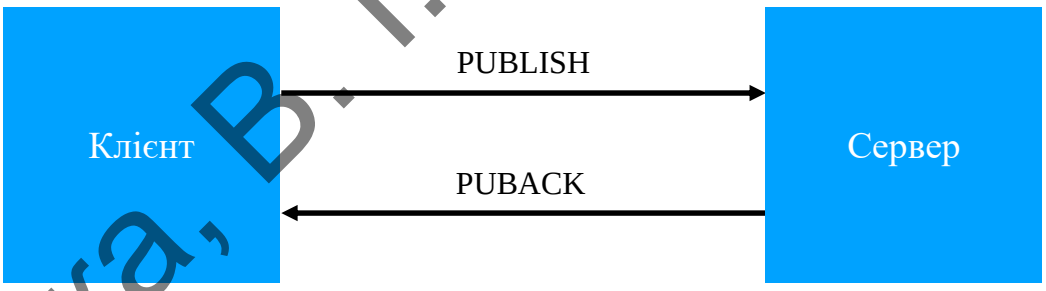

Рисунок 2.10 — Схема передачі повідомлення при QoS 1

QoS 2 Exactly once. На цьому рівні гарантується доставка повідомлень підписнику і виключається можливе дублювання відправлених повідомлень

(рис 2.11).

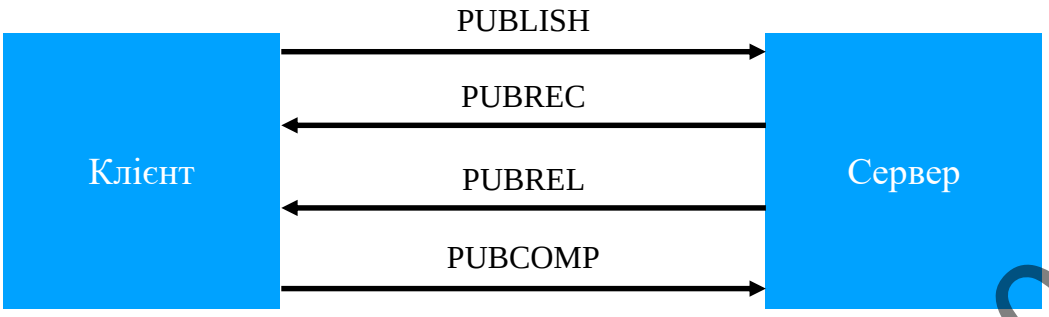

Рисунок 2.11 —Схема передачі повідомлення при QoS 2

Видавець відправляє повідомлення брокеру. У цьому повідомленні вказується унікальний Packet ID, QoS — 2 і DUP — 0. Видавець зберігає повідомлення непідтвердженими поки не отримає від брокера відповідь PUBREC. Брокер відповідає повідомленням PUBREC в якому міститься той же Packet ID. Після його отримання видавець відправляє PUBREL з тим же Packet ID. До того, як брокер отримає PUBREL він повинен зберігати копію повідомлення у себе. Після отримання PUBREL він видаляє копію повідомлення і відправляє видавцеві повідомлення PUBCOMP про те, що транзакція завершена.

Для забезпечення захисту передачі даних, в протоколі MQTT реалізовано наступні методи безпеки:

– Аутентифікація клієнтів. Пакет CONNECT може містити в собі поля USERNAME і PASSWORD. При реалізації брокера можна використовувати ці поля для аутентифікації клієнта;

– Контроль доступу клієнтів через Client ID;

– Підключення до брокера через TLS / SSL [16].

Переваги MQTT полягають у тому, що його легко реалізувати, гарантує доставку даних, агностик до переданих даних, виробляє низький рівень надлишкового часу роботи і працює протягом тривалих сесійних періодів. Поки що MQTT використовується для взаємодії з датчиками дистанційного керування та пристроями керування через низьку пропускну здатність та ненадійні канали зв'язку. MQTT також ідеально підходить для мобільних

додатків завдяки своїм маленьким розмірам, низьким споживанням енергії, накладними витратами та зручною багатоадресною передачею. Протокол MQTT підтримує різні платформи та різні популярні мови програмування. Це ідеальне рішення для Push повідомлень. Багато вітчизняних підприємства широко використовують протокол MQTT як протокол передачі повідомлень на телефоні та сервері. Завдяки простоті та масштабованості, він також широко використовується для відправлення повідомлень мобільним абонентам.

2.6 Сенсорні мережі та ZigBee

Бездротові сенсорні мережі (WSN) — це розподілена мережа багатьох датчиків (сенсорів) та виконавчих приладів, об'єднаних між собою через радіоканал (на основі стандартів 802.15.4/ZigBee). При цьому, зона покриття подібної мережі може складати від декількох метрів, до декількох кілометрів за рахунок властивості ретрансляції повідомлень від одного елемента до іншого. Застосовується дана технологія для вирішення багатьох практичних задач, пов'язаних з моніторингом, управління, логістикою, тощо.

Об'єднані в бездротову сенсорну мережу, датчики створюють розподілену, самоорганізовану систему збору, обробки та передачі інформації. Застосування інтелектуальних бездротових систем промислового моніторингу дає можливості, такі як:

– заміна кабелів в системах;

– своєчасне виявлення можливих відмов виконавчих механізмів по контролю таких параметрів, як вібрація, температура, тиск, тощо;

– контроль доступу до системи безпеки;

– контроль периметру об'єктів;

– контроль за переміщенням персоналу по території підприємства;

– автоматизація контролю проведення інспекцій та технічного обслуговування;

– контроль екологічних параметрів навколишнього середовища.

Перечислені вище можливості доступні вже сьогодні завдяки специфікації мережевих прокотолів верхнього рівня ZigBee, що регламентовано стандартом IEEE 802.15.4. Специфікація ZigBee орієнтована на додатки, які потребують гарантованої безпеки передачі даних при відносно невеликих швидкостях та можливості довготривалої роботи мережевих приладів від автономних джерел живлення. Так, один датчик може працювати від батарейки типу AA чи AAA близько 3-х років.

Основна особливість технології ZigBee полягає в тому, що вона при невеликому енергоспоживанні підтримує не просто прості мережеві топології («точка-точка», «дерево» і «зірка»), але і самоорганізуюча і самовідновлювальна ячейка (mesh) топологія з ретрансляцією і маршрутизацією повідомлень.

Крім того, специфікація ZigBee містить можливість вибору алгоритму маршрутизації, залежно від вимог програми та стан мережі, механізму стандартизації додатків — прикладних програм, стандартів кластерів, кінцевих точок, прив'язків, гнучких механізмів безпеки, а також забезпечує простоту розгортання, обслуговування та модернізації.

Мережі ZigBee будуються з базових станцій трьох основних типів: координаторів, маршрутизаторів і кінцевих пристроїв.

Координатор запускає мережу і керує нею. Він формує мережу, виконує функції центру управління мережею і довірчого центру (trust-центру) — встановлює політику безпеки, задає налаштування в процесі приєднання пристроїв до мережі, відає ключами безпеки.

Маршрутизатор транслює пакети, здійснює динамічну маршрутизацію, відновлює маршрути при перевантаженнях в мережі або відмові будь-якого пристрою. При формуванні мережі маршрутизатори приєднуються до координатора або іншим маршрутизаторів, і можуть приєднувати дочірні пристрою — маршрутизатори і кінцеві пристрої.

Маршрутизатор працюють в безперервному режимі, мають стаціонарне живлення і можуть обслуговувати «сплячі» пристрою. Маршрутизатор може обслуговувати до 32 сплячих пристроїв.

Кінцевий пристрій може приймати і відправляти пакети, але не займається їх трансляцією і маршрутизацією. Кінцеві пристрої можуть підключатися до координатора або маршрутизатора, але не можуть мати дочірніх пристроїв.

Кінцеві пристрої можуть переводитися в сплячий режим для економії заряду акумуляторів. Саме кінцеві пристрої мають справу з датчиками, локальними контролерами і виконавчими механізмами[9].

2.7 NB-IoT

Narrowband-IoT (NB-IoT) — це новий стільниковий стандарт, призначений для обмежених пристроїв на основі модифікованого повітряного інтерфейсу LTE. Іншими словами, даний стандарт розширює можливості використання стільникового зв'язку для різних обмежених пристроїв [8].

Крім того, NB-IoT пропонує гнучкість розгортання, що дозволяє операторам мобільних мереж реалізувати NB-IoT, використовуючи дуже вузьку смугу спектру. Ця функція дозволяє спростити розгортання програм з низьким енергоспоживанням. Технічно, NB-IoT слідує за дизайном MAC звичайних мереж LTE. Істотні відмінності стосуються лише ширини і пропускної здатності. Ключові відмінності між останніми специфікаціями 3GPP та NB-IoT пояснюються в таблиці 2.5 [8].

Таблиця 2.5 Порівняння 3GPP стандартів мобільного зв’язку [8]

LTE	**Rel. 8 Cat 4**	**Rel. 11 Cat 1**	**Rel. 12 Cat 0**	**Rel. 13 (LTE-M) Cat 1.4 MHz**	**Rel. 13 (NB-IoT) Cat 200 kHz**
Downlink	150 Mbps	10 Mbps	1 Mbps	1 Mbps	200 kbps
Uplink	50 Mbps	5 Mbps	1 Mbps	1 Mbps	144 kbps

Таблиця 2.5 Порівняння 3GPP стандартів мобільного зв’язку [8]

LTE	**Rel. 8 Cat 4**	**Rel. 11 Cat 1**	**Rel. 12 Cat 0**	**Rel. 13 (LTE-M) Cat 1.4 MHz**	**Rel. 13 (NB-IoT) Cat 200 kHz**
Duplex	Full	Full	Half	Half	Half
Bandwidth	20 MHz	20 MHz	20 MHz	1.4 MHz	200 kHz
Tx power	23 dBm	23 dBm	23 dBm	20 dBm	23 dBm

У DL (передача даних від мобільної станції до базової станції, DL) NB-IoT використовує OFDMA (ортогональний частотний розподіл багаторазового доступу) з інтервалом піднесущих 15 кГц. Як і в LTE, передача організована за часом 10 мс. Кожен кадр складається з 10 субкадрів, які є найменшим можливим розподілом каналів у часовій області. Кожна підкамера складається з двох слотів (кожні 0,5 мс), яка точно така ж, як у звичайних LTE [8].

Однак у частотній області NB-IoT використовується лише один фізичний ресурсний блок (LTB) LTE, тобто дванадцять піднесучих 15 кГц із загальною смугою пропускання 180 кГц, що в сім разів менше, ніж у стільниковому LTE. Ця модифікація зменшує пропускну здатність каналу до вимог пристроїв IoT. Через нижчу пропускну здатність, обробка сигналу в NB-IoT набагато менш складна і може оброблятися навіть обмеженими пристроями IoT. Незважаючи на це, NB-IoT використовує таку ж структуру ресурсної сітки, що забезпечує хороше співіснування з регулярними мережами LTE.

У UL (передача даних від базової станції до мобільної станції, UL) NB-IoT також використовує загальну смугу пропускання 180 кГц. Передача Multicarrier базується на SC-FDMA, з інтервалом піднесущих 15 кГц, часовими інтервалами 0,5 мс і субкадрами 1 мс, як у звичайних мережах LTE. Таким чином, структура блоків ресурсів така сама, як у низхідній лінії зв'язку. Проте для сигналізації використовуються різні фізичні канали, а загальна кількість ресурсів для корисного навантаження відрізняється [8].

NB-IoT підтримує три різних варіанти розгортання: діапазон, захисний діапазон та автономний режим. Різниця між цими випадками розгортання показана на рис 2.12. Внутрішнє розгортання використовує блоки ресурсів NB-IoT у старому спектрі LTE, які вимагають додаткової інформації про сигнали, щоб уникнути конфліктів з LTE UE. Розгортання в охоронних смугах не вимагає додаткового контролю, оскільки LTE UE не використовує ці смуги. Третя опція — це автономне розгортання, що також не заважає UE LTE, але вимагає придбання додаткових діапазонів спектрів, що збільшує витрати на розгортання.

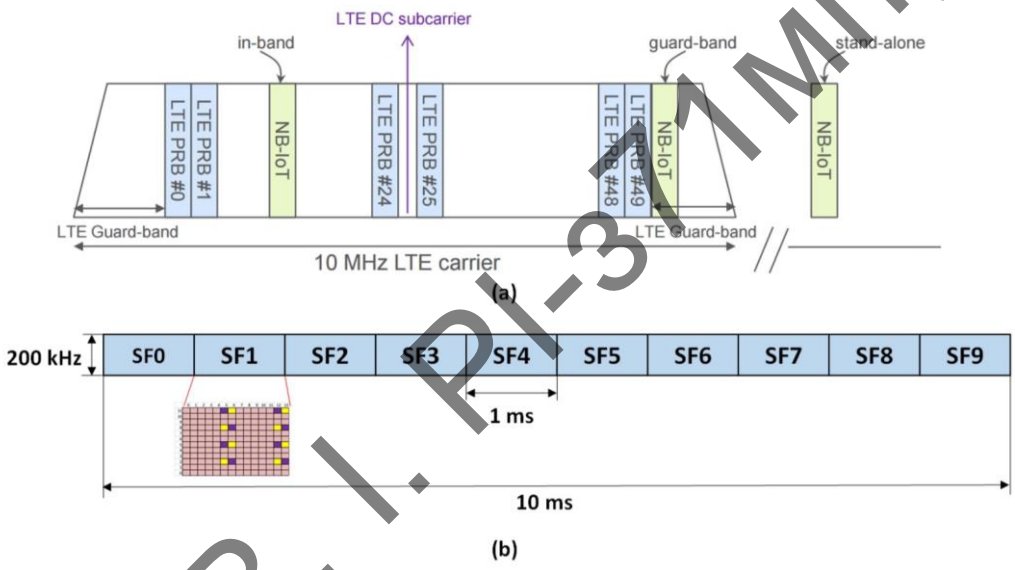

Рисунок 2.12 — NB-IoT/LTE в частотному (a) та часовому (b) діапазонах [8]

Ресурсний блок NB-IoT складається з 12 піднесучих та 7 символів часу, розподілених в межах 180 кГц та 0,5 мс відповідно. Кожен пристрій IoT використовує один ресурсний блок у частотній області та два послідовні блоки ресурсів (1 мсек субкадр) у часовому домені. Таким чином, загальна кількість елементів ресурсу для одного пристрою:

$$N_{RES} = 12 * 14 = 168 \qquad (2.1)\ [8]$$

Залежно від сценарію розгортання, кількість елементів ресурсу для сигналізації відрізняється. Для розгортання автономної і захисної смуги використовуються лише 16 пілотних контрольних символів для оцінки якості

каналу низхідної лінії зв'язку над спеціальними піднесучими (рис. 2.13). Таким чином, кількість елементів для корисної навантаження становить:

$$N_{payload} = 168 - 16 = 152 (2.2) \quad [8]$$

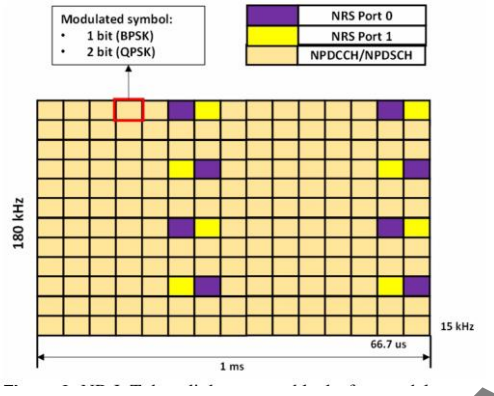

Рисунок 2.13 — NB-IoT DL ресурс для автономного розгортання [8]

Для розгортання в межах діапазону для LTE UE використовуються додаткові 16 спеціальних лінійних символів для посилання, а 28 каналів фізичного контролю використовуються для передачі інформації про планування та розподілу ресурсів для UE LT (рис 2.14). Загальна кількість елементів для корисного завантаження IoT буде:

$$N_{payload} = 168 - 16 - 16 - 28 = 108 \quad (2.3) [8]$$

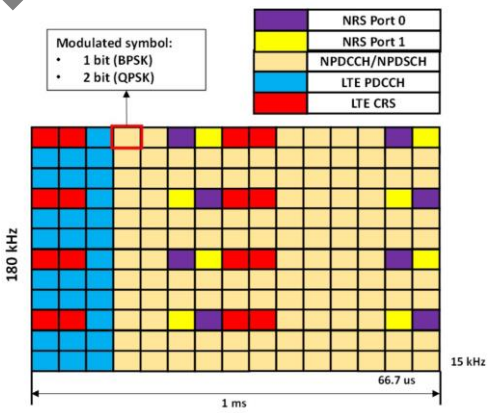

Рисунок 2.14 — Розгортання NB-IoT в межах діапазону

У UL для будь-якого типу розгортання використовуються 36 контрольних символів (рис 2.15). Таким чином, загальна кількість елементів для корисної завантаженості IoT буде:

$$N_{payload} = 168 - 36 = 132 \quad (2.4)\ [8]$$

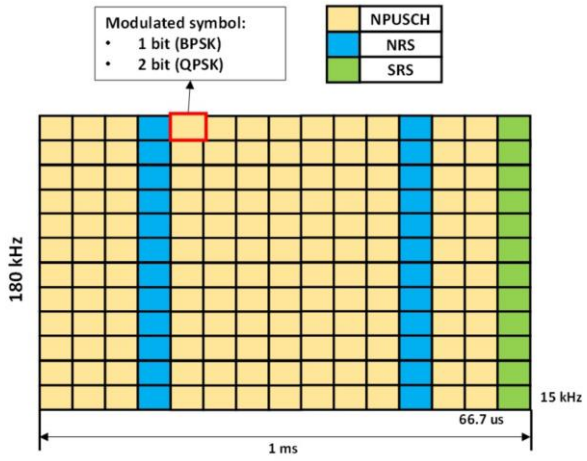

Рисунок 2.15 — NB-IoT UL ресурс блоки [8]

Для передачі корисного навантаження NB-IoT використовує транспортні блоки з 1000 біт для висхідної лінії зв'язку та 680 біт для низхідної лінії, що в сотні разів менше, ніж у звичайному LTE.

3 РОЗРОБКА УНІФІКОВАНОЇ АРХІТЕКТУРИ

3.1 Обґрунтування вибору мови програмування

Із такими даними не можна погодитись, що третьою найпопулярнішою мовою програмування є JavaScript. У сучасному IT світі, JavaScript — це найпопулярніша мова програмування, навіть обходить Java (підтверджують результати дослідження спільноти StackOverflow).

Хоча багато хто все ще думає про JavaScript як мову для появи модальних вікон на веб-сторінках, відносно незрозуміла популярність мови на сервері робить це дивовижним популярним вибором для додатків IoT. 41,8% розробників у опитуванні Eclipse вибрали JavaScript, і 31,5% зазначили, що вони використовують Node.js у своїх проектах.

Значна частина цієї роботи зосереджена на серверах і шлюзах або концентраторах, які збирають інформацію, а потім зберігають її. Менші інтелектуальні вузли та сенсори, які запускаються під керуванням Linux, зазвичай запускають Node.js.

Але навіть якщо більшість коду Node.js працює на машинах з більшою потужністю, існують певні зусилля, спрямовані на те, щоб привести його до менших. Два приклади мікроконтролерів Espruino та Tessel, які запускають JavaScript з самого початку. Наприклад, Tessel побудований навколо Node.js, що дозволяє веб-розробникам легко перейти до IoT без вивчення нової мови.

Все ж таки, що робить Node.js таким особливим та найбільш підходящим для програмування пристроїв IoT.

По-перше, програми Node.js написані на JavaScript, це мова веб, єдина мова програмування, яка з самого початку підтримується більшістю веб-браузерів. Цей аспект дозволяє використовувати лише сценарії, такі як однозначні набори програм і спільне використання коду між сервером і клієнтом. Сам Node.js сприяє підвищенню та еволюції мови JavaScript. Люди розуміють, що використання JavaScript на сервері не так вже й погано, як і в браузері. І вже зарах, його люблять через прагматизм та гібридний характер,

що є на півдорозі між об'єктно-орієнтованим та функціональним програмуванням.

Другий революціонізуючий фактор — це його однопотокова асинхронна архітектура. Окрім явних переваг з точки зору продуктивності та масштабованості, ця характеристика змінила спосіб розробки підходу конкуренції та паралелізму. Блокування замінюються чергами, потоками за допомогою зворотних викликів та подій, а також синхронізації за викликами.

Останній і найважливіший аспект Node.js полягає в його екосистемі: менеджер пакетів npm, постійно зростаюча база даних модулів, ентузіазм та спільнота, що готова допомогти, а головне — власна культура, що базується на простоті, прагматизмі та надзвичайній модульності.

Однак, завдяки цим особливостям, розробка на Node.js дає вам абсолютно інше відчуття в порівнянні з іншими платформами на стороні сервера, і будь-який розробник, новий для цієї парадигми, часто відчуває сумніви щодо того, як ефективно вирішувати навіть найбільш поширені проблеми з проектуванням та кодуванням.

Як і в будь-якій іншій формі зв'язку H2H або D2D, має бути протокол, який заохочує людей або пристрої розуміти один одного. У випадку комунікації IoT між D2D або M2M та хмарою існує широкий набір протоколів, що полегшують зв'язок. У таблиці 3.1 показано найпопулярніший стек протоколів IoT.

3.2 Архітектура

За думкою Роба Ван Краненбурга, інтернет речей являє собою “чотири-шаровий пиріг”. Перший рівень пов’язаний з ідентифікацією кожного об’єкта. Другий рівень представляє разом із сервісом по обслуговуванню потреб споживача (можна розглянути як мережу власних “речей”, наприклад, “розумний дім”). 3-й рівень пов’язаний з урбанізацією міського життя. Тобто, це концепція “розумного міста”, де вся інформація, що стосується мешканців даного міста, зосереджується в конкретному житловому кварталі, в

конкретному будинку та в сусідніх будинках. Нарешті 4-й рівень – сенсорна планета.

Іншими словами, інтернет речей можемо розглядати як мережу мереж, в якій невеликі мало суміжні мережі створюють більш масштабні мережі — як зображено на рис 3.1 [1].

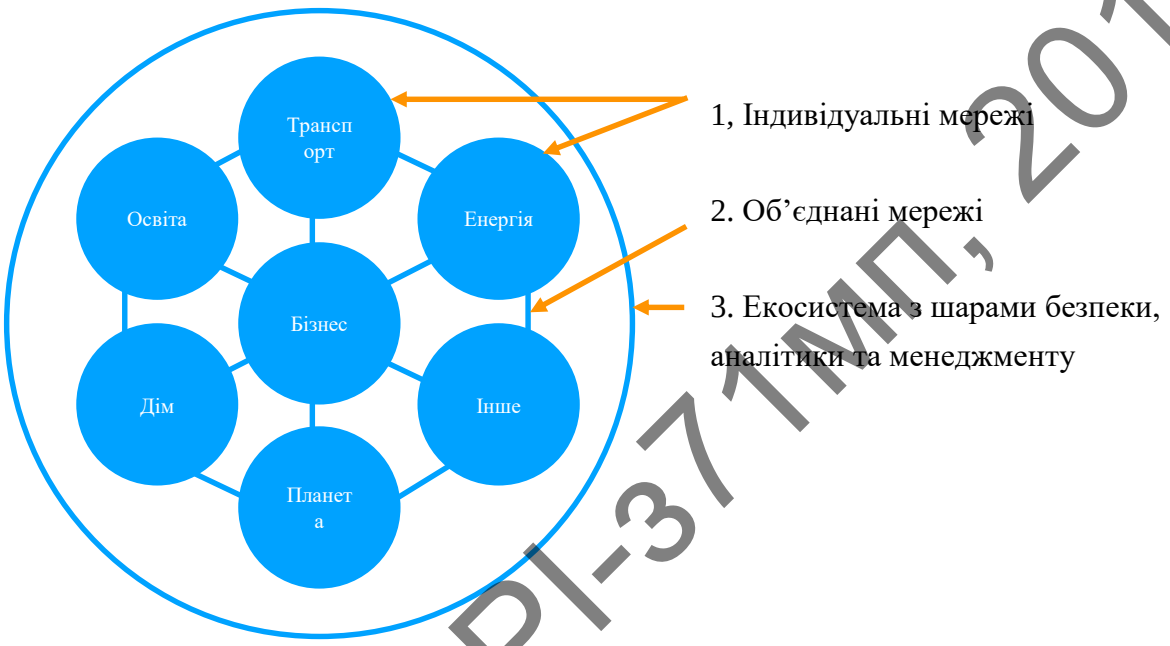

Рисунок 3.1 — Схема взаємодії мікромереж IoT [1]

Архітектура IoT має форму, аналогічну стандартній моделі ISO (ISO) та IEC (1994), Transmission Control Protocol (TCP) та Internet Protocol (IP) Suite, а також Міністерства оборони США 4-шарові моделі (DoD4), що показані в таблиці 3.1 [19].

Таблиця 3.1 OSI, TCP та DoD4 моделі [19]

OSI Model	**TCP Model**	**DoD4 Model**
Application	Application	Process
Presentation	Transport	Host-to-Host
Session	Internet	Internet
Transport	Network Access	Network Access
Network		
Data Link		
Physical		

З таблиці 3.1 можна побачити, що модель (TCP) та модель DoD4 є 4-шаровими, і вони схожі між собою. Насправді, на базі ISO, TCP та DoD4 архітектура IoT могла бути реалізована без подальшого моделювання, але вони не можуть відтворити особливості IoT та проблеми, такі як зв'язок та зв'язок, збір та аналіз даних, керування пристроями, масштабованість, взаємосумісність, інтеграція та безпека у повному обсязі [18].

Таким чином, існує потреба в реструктуризації всіх трьох моделей, щоб вони відповідали особливостям і проблемам IoT. Модель архітектури IoT складається з різних компонентів і може бути 4-шаровою центричною архітектурою, в якій на кожному рівні є можливість реалізувати конкретні технології. У таблиці 3.2 показана 4-х шарова модель IoT, яка показує, які компоненти реалізуються на кожному рівні.

Таблиця 3.2 Чотири шарова модель IoT [18]

Шар	**Компоненти**	
Програмний	Навколишнє середовище, енергетика, охорона здоров'я, транспорт, відстеження людей, спостереження, роздрібна торгівля	
Сервісний управляючий	Моделювання, конфігурація та управління	Управління потоками даних та контроль безпеки
Шлюз та мережа	WAN (GSM, UMTS, LTE, LTE-A, 5G в найближчому майбутньому)	WiFi, Ethernet
Рівень збору даних	Сенсорні мережі, датчики (RFID, штрих-код та подібні)	

3.2.1 Комунікація датчиків та мережа

Нижній рівень в основному являє собою пристрої IoT, і вони містять різні типи та форми архітектури, властивості та можливості. Пристрій може

розглядатися як пристрій IoT, якщо такий пристрій має будь-яку форму зв'язку, яку можна безпосередньо або опосередковано підключити до мережі. У таблиці 3.3 наведено приклад пристроїв, які можна знайти на підключенні до датчиків та мережевому рівні. Пристрої на сенсорному рівні мають можливість зчитувати та збирати інформацію в режимі реального часу для обробки. Вони мають низьку потужність та низьку швидкість передачі даних для підключення. Області застосування деяких з цих датчиків можна назвати сенсором тіла, датчиками довкілля та датчиками спостереження.

Таблиця 3.3 Технології та рівні підключення

Рівень підключення	**Технології**
LAN	WiFi, Ethernet
PAN	Bluetooth, ZigBee, Z-Wave, 6LoWPAN, UWB, Wired
Сенсори, датчики	Гіроскоп, акселерометр, GPS, фоторезистор, датчики температури, вологи, тиску тощо
Мітки	RFID, штрих-код

3.2.2 Шлюз та мережевий рівень

Шлюз та мережевий рівень, також відомий як комунікаційний рівень, підтримує зв'язок пристроїв на рівні пристрою або сенсорних мереж. Він складається з різноманітних протоколів, які допомагають у спілкуванні між пристроями та хмарним сховищем. Найбільш помітними з цих протоколів є HTTP з підходом RESTful, MQTT та CoAP. Таблиця 3.4 показує шлюз та мережевий рівень із технологіями, які беруть участь у ньому.

Таблиця 3.4 Шлюз та мережевий рівень [18]

Мережа	WAN, 3G, LTE, LTE-A, M LoRa, Sigfox, 5G	LAN, WiFi, Ethernet
Шлюз	Мікроконтроллери, радіокомунікаційний модуль, точка доступу, SIM модуль шифрування	

Крім того, одним із найважливіших аспектів шлюзу та мережевого рівня є їх здатність збирати дані, а також тримати зв'язок між посередниками. Комунікація з посередником та агрегація даних поєднують в собі безпосередньо зв’язок та дані з різних пристроїв, а потім перенаправляють інформацію на конкретний пристрій через шлюз. Шлюз та мережевий рівень також можуть підтримувати, наприклад, HTTP-сервер та брокер MQTT для забезпечення зв'язка між пристроями. Крім того, він служить мостом і перетворювачем між різними протоколами, такими як HTTP-API, на основі повідомлення MQTT на пристрої [20].

3.2.3 Управляючий сервісний рівень

Рівень сервісу управління складається з двох основних функціональних частин, як зазначено в таблиці 3.2. Дві основні функціональні частини — це моделювання, конфігурація та управління, а також частина управління потоком даних та контроль безпеки. Однак, перш ніж розглядати функції частин рівня керуючого сервісу, важливо також описати, що таке служби керування. У таблиці 3.5 наведено деякі служби, які може запропонувати рівень службової підтримки.

Як показано в таблиці 3.5, рівень управління сервісом має важливі ролі в архітектурі IoT. Ролі можна згрупувати у дві частини. Управління службами даних відповідає за процеси, такі як інформаційна аналітика, контроль безпеки, моделювання процесів та управління пристроями. Взагалі,

управління даними має дві форми методик, періодичні та неперіодичні схеми управління даними.

Таблиця 3.5 Компоненти управляючого сервісного рівня [18]

Сервіси	**Сервісні компоненти**
Операційна службова підтримка	Управління пристроями, конфігурація, управління продуктивністю, контроль безпекою
Аналітична сервісна платформа	Статистичний аналіз, добування даних, інтелектуальний аналіз
Система підтримки платежів	Платіжні звіти
Безпека	Котроль доступу, шифрування, ідентифікація доступу
Управління бізнес потребами	Визначення потреб, моделювання, симуляція, виконання
Управління бізнес потребами	Моделювання робочого процесу, симуляція, виконання

Як показано в таблиці 3.5, рівень управління сервісом має важливі ролі в архітектурі IoT. Ролі можна згрупувати у дві частини. Управління службами даних відповідає за процеси, такі як інформаційна аналітика, контроль безпеки, моделювання процесів та управління пристроями. Взагалі, управління даними має дві форми методик, періодичні та неперіодичні схеми управління даними.

У періодичній обробці даних IoT інформація або дані періодично збираються датчиком для аналізу. Наприклад, монітор температурного датчика фіксує певну кількість інформації про погоду або стан промислової машини протягом певного періоду часу. Проте не всі зібрані зібрані відомості будуть необхідні для аналізу, отже, очищення даних, зібраних датчиком, необхідне для відфільтрування небажаного та збереження необхідних для фактичної мети збору даних.

У неперіодичної технології збору даних датчик збирає дані і вимагає негайної реакції на інформацію, як тільки відбувається подія. Наприклад, якщо сенсорний пристрій IoT відстежує пацієнта та безпека контролюється, доставка інформації повинна бути негайною і вимагати негайної відповіді. Крім блоку управління даними, існує також блок, який забезпечує управління потоком даних, доступу до інформації, інтеграції та управління даними. Існує також блок абстракції даних, який надає послуги, такі як обробка збору інформації.

3.2.4 Програмний рівень

Програмний рівень являє собою верхній шар, який служить інтерфейсом між додатком датчика та кінцевими користувачами. Він застосовується у різних галузях, таких як охорона навколишнього середовища, промисловість, охорона здоров'я, відстеження майна інтелектуальної власності та інших, як це показано в таблиці 3.6. Це також рівень, на якому розміщуються протоколи IoT, такі як HTTP, MQTT, CoAP, AMQP, XMPP, SOAP.

Оскільки різні програми в різних галузях мають різні протоколи та класифікації, виходячи з типу мережі, зони покриття, розміру, бізнес-моделі, систем реального часу, протоколи програми шарів можуть виділяти, пов’язувати та обмінюватись даними або інформацією через інших системи та додатки. Класифікація IoT грунтується на доменах програм, таких як особисте та домашнє, промислове, службове та мобільне. Ці класифікації визначають розмір області застосування та визначають його характеристики.

Таблиця 3.6 Програмний рівень IoT

Сектори	Smart Environmental, Smart Energy, Smart Transportation, Smart Healthcare, Smart Retail, Smart Industry, Smart Military
Ринкові зони	Supply Chain, People Tracking, Asset Management, Fleet Management, Surveillance

Наприклад, домен особистого та домашнього застосування представляє невеликий масштаб. Це означає обмежене число користувачів, фізичних осіб. Промисловий домен IoT представляє велику кількість користувачів на рівні громади. Утиліта IoT представляє набагато більший обсяг користувачів, таких як національна або регіональна IoT підтримка, які, як правило, поширюються в інших областях через їхню мобільність, а пристрої, в основному, працюють на акумуляторі та портативні. Таблиця 3.6 ілюструє деякі основні домени додатків і ринкові зони, сектори, які можуть розміститися на рівні програми.

3.3 Контейнеризація

Технологія віртуалізації працює як абстрактний рівень для апаратного забезпечення, який дозволяє декілька наборів робочих навантажень розподіляти загальний набір ресурсів, тоді як робочі навантаження виконують властивість ізоляції один від одного.

Віртуалізація забезпечує різні переваги, такі як використання ресурсів, більш висока доступність системи, менший час відмови від роботи, покращення балансування навантаження, відокремлення між запущеним додатком, менша вартість адміністрування та обслуговування. Віртуалізація може бути розділена залежно від типу ресурсів, що будуть віртуалізовані, наприклад віртуалізація обладнання та віртуалізація рівня операційної системи.

Віртуалізація рівня апаратури — це віртуалізація на основі гіпервізора. У цьому виді віртуалізації гостьова ОС працює над операційною системою з власним ядром. Зв'язок гостьової системи з фактичним обладнанням здійснюється через абстрактний шар гіпервізора. Цей вид віртуалізації забезпечує безпеку та ізоляцію, але має великі затрати ресурсів за рахунок емуляції апаратного забезпечення (комунікаційні накладні витрати між гостьовою ОС на апаратне забезпечення хоста). Щоб зменшити цю витрату, можна використовувати інший тип віртуалізації, тобто операційну систему

віртуалізації рівня системи, це також відоме як віртуалізація на базі контейнера [3].

Контейнер — це легкий підхід до віртуального оточення, який ізолює ресурси та хост до будь-яких інших контейнерів. Контейнери використовують одне і те ж ядро хост-системи, що забезпечує практично нульові витрати на роботу та забезпечує кращу продуктивність обчислень ресурсів. Контейнери дозволяють обмінюватися доступом до файлів бібліотек та бібліотек з іншими контейнерами, тоді як апаратна віртуалізація обмежує область застосування в одній середовищі віртуальних машин [3,4]. Пристрої IoT часто не мають ресурсів, тому контейнерні технології допоможуть використати ресурси, оскільки вони поділяють файли і бібліотеки, що допомагають зберегти місце зберігання. Всі контейнерні технології не відповідають вимогам IoT додаткам [3, 12].

Docker надає просту у використанні сервіс-орієнтовану архітектуру (SOA) контейнерних технологій. Ця архітектура SOA може обробляти складні програми, де кожна служба може бути незалежно розгортатися та ізолюватися один від одного [5]. Докер в основному є легкою технологією контейнеризації з відкритим кодом, що дає платформу для розробки, запуску та масштабування програми [11].

Розробка додатків IoT є трохи складною і існує ще багато проблем, таких як сумісність, безпека, потужність, мережеві підключення та стандартні протоколи. Докер може дати PaaS та/або SaaS рішення для легкого розвитку програми IoT. Нижче наведено ключові моменти, які пояснюють Docker для вимог IoT.

Легке і швидке розгортання для додатків. Контейнер Docker може працювати на будь-якій сучасній системі Linux або системі Windows. Він має менший час запуску, оскільки накладні витрати часу майже дорівнюють нулю, це також зменшує час циклу розробки та тестування та розгортання програми

.

Кластерний планувальник та служба виявлення. Функція кластера планувальника докерів дозволяє керувати кластером докер-хосту, який забезпечує автоматичне переключення та переміщення контейнера у випадку обмежених ресурсів.

Функції виявлення служби допомагають зберігати IP-адреси та порти запуску програми у способі розподіленої клавіші-значення. Докер має механізм зв'язування контейнерів, в якому він передає змінні середовища та IP-адресу лише для об'єднання контейнера [5].

Безпека та конфіденційність. Додатки IoT — це M2M комунікації у розподіленому чи децентралізованому середовищі, тому безпека є однією з проблем вимірювання. Докер забезпечує хороший механізм безпеки. Це дозволяє безпеці рівня ядра або операційної системи хоста захищати від загрози. Він також забезпечує сертифікацію TLS / SSL для перевірки сервера / клієнта сховища. Docker дозволяє виконувати лише root доступ до програми, яка працює всередині контейнера [5,11].

Докер пропонує можливості дистанційного керування та переносимість різних пристроїв. Докер забезпечує масштабованість і простоту обробки великого обсягу даних.

Рішення для управління. Вартість керування, моніторингу та розробки програми IoT є високою. Індустрії IoT шукають рішення для скорочення вартості (гроші + робоча сила) для розробки програм. Докер може забезпечити низьку вартість; низький рівень технічного обслуговування додатку для розробника та адміністратора.

3.4 Вибір протоколу передачі даних

Для оцінки прозорості системи передачі даних ми обчислюємо кількість необхідних субкадрів для передачі одного транспортного блоку для DL та UL, для NB-IoT, MQTT та CoAP відповідно.

$$N_{SF} = \begin{cases} 1000/(132 * 2) = 4, & UL \\ 680/(108 * 2) = 4, & DL\ в\ діапазоні\ LTE \\ 680/(152 * 2) = 3, & DL\ в\ автономному\ режимі \end{cases} \quad (3.1)\ [8]$$

Для порівняння потужності CoAP та MQTT, вирахуємо кількість транспортних блоків, необхідних для передачі одного пакету для кожного протоколу:

$$N_{TB\_MQTT} = \begin{cases} 32768/1000 \approx 33, & UL \\ 32768/680 \approx 49, & DL \end{cases} \quad (3.2)\ [8]$$

$$N_{TB\_CoAP} = \begin{cases} 9216/1000 \approx 10, & UL \\ 9216/680 \approx 14, & DL \end{cases} \quad (3.3)\ [8]$$

Для кожного типу розгортання ми обчислюємо відповідну кількість субкадрів для передачі одного пакета наступним чином:

$$N_{SF\_MQTT} = \begin{cases} 4 * 33 = 132, & UL \\ 4 * 49 = 196, & DL\ в\ діапазоні\ LTE \\ 4 * 49 = 147, & DL\ в\ автономному\ режимі \end{cases} \quad (3.4)\ [8]$$

$$N_{SF\_CoAP} = \begin{cases} 4 * 10 = 40, & UL \\ 4 * 14 = 56, & DL\ в\ діапазоні\ LTE \\ 3 * 14 = 42, & DL\ в\ автономному\ режимі \end{cases} \quad (3.5)\ [8]$$

Розрахуємо пропускну спроможність для MQTT та CoAP відповідно:

$$R_{MQTT} = \begin{cases} 1000/132 * 32768 = 229kbps, & UL \\ 1000/196 * 32768 = 164kbps, & DL\ в\ діапазоні\ LTE \\ 1000/147 * 32768 = 197kbps, & DL\ в\ автономному\ режимі \end{cases} \quad (3.6)\ [8]$$

$$R_{CoAP} = \begin{cases} 1000/40 * 9216 = 230kbps, & UL \\ 1000/56 * 9216 = 157kbps, & DL\ в\ діапазоні\ LTE \\ 1000/42 * 9216 = 212kbps, & DL\ в\ автономному\ режимі \end{cases} \quad (3.7)\ [8]$$

Ми порівняли пропускну спроможність протоколів CoAP та MQTT над мережею NB-IoT. Відповідно до результатів, NB-IoT в UL має однакову пропускну здатність для обох протоколів. У DL MQTT має більшу пропускну спроможність для розгортання в діапазоні, а CoAP працює краще для автономної роботи.

3.5 Концепція уніфікованої архітектури

На Рис 3.2 представлена загальна архітектура IoT рішення. Вельми передбачувано, що все починається з датчиків (Detectors). Більш того, чим

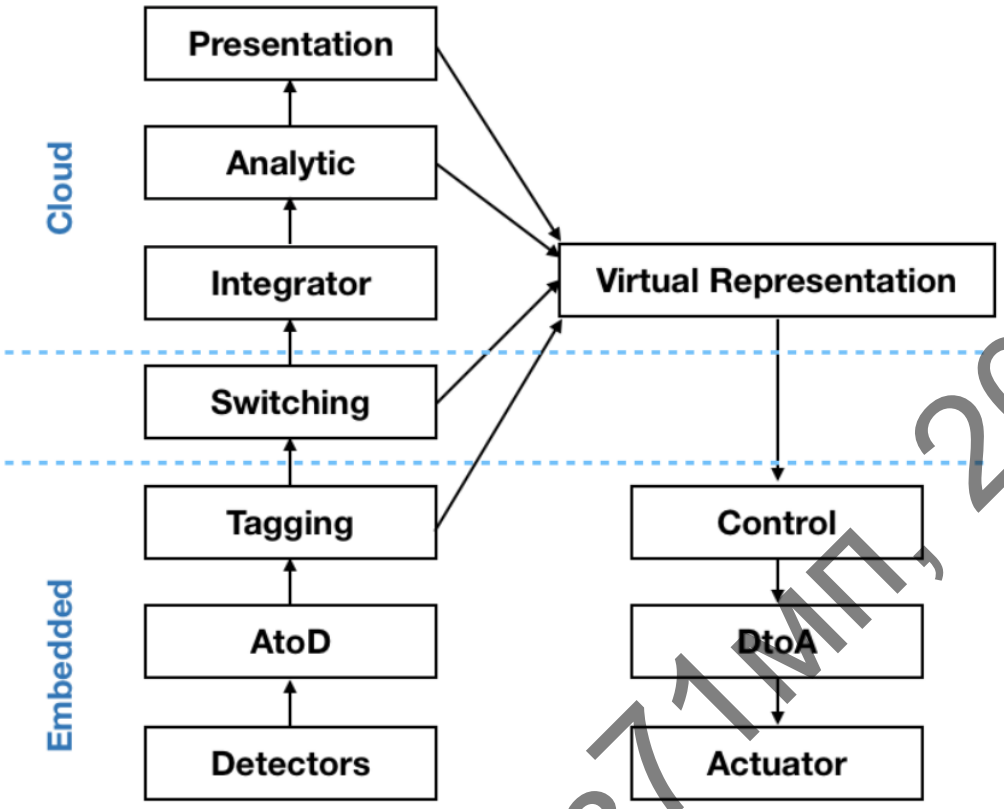

Рисунок 3.2 — Структурна схема архітектури IoT [7]

краще підходить датчик для виконання свого завдання, тим ефективніше буде система надалі.

Важливо відзначити, що датчик реєструє зміну навколишнього середовища, а не її статичний стан. Датчики поділяються на активні — випромінюють самі сигнали і приймають відображення; і пасивні — працюють тільки на прийом. Справедливо, що останні значно виграють за параметрами енергоспоживання.

Більшість датчиків базуються на прийомі хвиль — звукових, ультразвукових, світлових різного діапазону, теплових. Однак є категорія датчиків, що базуються на зміні їх фізичних характеристик, таких як індуктивність, ємність, тиск. Гарні результати будуть від комбінації декількох датчиків, наприклад PIR детектор і ємнісний датчик для визначення руху.

У будь-якому випадку, датчик формує аналоговий сигнал, який необхідно перевести в цифру для подальшої обробки, чим і займається перетворювач (AtoD). Після отримання цифрової інформації вона повинна бути оброблена

локальним процесором периферійного пристрою. Головне його завдання проставити ярлик (Tagging) отриманої інформації або просто класифікувати її.

Ярлики можуть бути найпростішими, як наприклад — є “рух", так і більш складними — “рух” + “швидкість”. Іноді потрібні багатовимірні ярлики — “рух”, “машина”. Чим складніший ярлик, тим більша потужність периферійного процесора і, відповідно, енергоспоживання. З іншого боку, чим більше інформативний ярлик, тим менше необхідну кількість інформації, що передається в хмару і соответсвенно потрібна менше смуга пропускання, а так само збільшується швидкість реакції на подію. Всі ярлики мають часову мітку (timestamp).

Наступна архітектурна одиниця (Switching) може знаходитися як в хмарі, так і на периферії, а іноді в обох частинах. Комутатор перенаправляє отриману інформацію в різні об'єкти, класифікуючи ярлики. Цими об'єктами можуть бути сервера, черги, лямбда або просто сховище.

До сих пір робота велася з інформацією від конкретного периферійного пристрою і фактично нічим не відрізнялася від роботи автоматизованих систем управління. Одначе, на наступному рівні — інтеграції, починається якісну відмінність (Integrator). Інформація від різних периферійних пристроїв підсумовується по однотипним ярликам. При цьому самі типи периферійних пристроїв можуть бути навіть різними. Важливо, що ярлики потрапляють в єдину точку, що відповідає за прийом відповідного події — ярлика.

Надалі інформація від всіх об'єктів, що підсумовують та класифікують ярлики, систематизується аналітичним блоком (Analytic). У ньому полягає основна логіка або, якщо так можна висловитися, мізки системи. Там знаходиться штучний інтелект (AI, Machine Learning), проста бізнес логіка, тощо. Результат роботи аналітичного блоку передається в блок презентації для відображення користувачеві (Presentation). Це може виглядати як відправка повідомлення на мобільний пристрій, графік на WEB або інше.

Оскільки IoT система є розподіленою і пов'язана іноді ненадійним каналом зв'язку, доводиться мати механізми гарантованої доставки

інформації. У тому випадку, якщо не вдається передати інформацію від переферійного пристрою в хмару, здійснюються повторні спроби передачі. Те саме має відбуватися і в зворотньому напрямку.

Для цього вводиться блок віртуального представлення (Virtual Representation) периферійного пристрою, в який записується інформація для передачі периферійних пристроїв або його новий стан. Часто це просто текстовий файл, але може бути і більш точна модель представлення. Варто відзначити, що зміни в модулі віртуального представлення можуть бути ініційовані з різних модулів вхідної мережі пристроїв.

Отже, датчики реєструють зміну навколишнього середовища в режимі реального часу. Наступний модуль класифікації виробляє первинне сегментування на певні події системи. В цілому розробка архітектури IoT програми і повинна починатися зі списку цих подій. Підсумовування ярликів проводиться по групі периферійних пристроїв з однотипними ярликами.

Модуль інтегрування призначений для винесення рішення по апроксимації (передбачення подальших подій) або детермінованість (виявлення ситуації з безлічі варіантів). Ця інформація служить своєрідним ключем-коефіцієнтом для модуля віртуальної моделі периферійного пристрою, в якому актуальна інформація від самого периферійного пристрою перетворюється на підставі ключа-коефіцієнта в новий стан периферійного пристрою.

Слід зазначити, що наступні правили повинні виконуватись задля забезпечення безпеки та цілісності даних:

– зберігання інформації має здійснюватись як на рівні периферійного пристрою, так і в хмарному сховищі. Периферійний пристрій зберігає свою програму, настройки, стан та тимчасово зберігає інформацію від датчиків, поки вона гарантовано не передана в хмарне сховище;

– кожен периферійний пристрій має авторизуватися в системі індивідуально [7].

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Враховуючи той факт, що виконання даної дипломної роботи потребує використання засобів обчислювальної техніки, то в цьому розділі запропоновані технічні рішення та практичні заходи, що пов’язані із створенням безпечних умов праці при використанні персональної електронно-обчислювальної машини (ПЕОМ).

В першу чергу, передбачається, з урахуванням вимог ДСТУ ISO 9241-6:2004, ДСанПіН 3.3.2.007 та ДНАОП 0.00-1.31-99, визначити потенційно небезпечні і шкідливі фактори, що виникають при експлуатації відеодисплейних терміналів (ВДТ) ПЕОМ, вплив цих факторів на користувачів ВДТ, розглянути принципи їх нормування, а також розробки комплексних заходів щодо запобігання шкідливого впливу цих факторів на людину. Також цьому розділі запропоновано технічні рішення та організаційні заходи з безпеки в надзвичайних ситуаціях (НС).

4.1 Визначення основних потенційно небезпечних та шкідливих виробничих факторів

Основними шкідливими та небезпечними виробничими факторами, які пов’язані з використанням ПЕОМ, є такі фактори:

– електромагнітне випромінювання радіочастотного діапазону;

– наявність іонізуючого рентгенівське випромінювання (НРВ);

– випромінювання оптичного діапазону (ультрафіолетове, інфрачервоне і випромінювання видимого діапазону);

– можливість ураження електричним струмом;

– електростатичне поле;

– недостатня освітленість робочої зони;

– підвищений рівень шуму;

– значна напруга зорових органів і пов’язане з цим перевтомлення користувача ПЕОМ;

– значне навантаження на пальці і кисті рук, що при відсутності профілактики і медичного контролю, може викликати професійні захворювання;

– тривале перебування в одному й тому ж самому положенні сидячи, що викликає застійні явища в організмі людини;

– відблиски на екрані монітора

– можливість виникнення пожежі.

4.2 Технічні рішення та організаційні заходи з безпеки і гігієни праці та виробничої санітарії

4.2.1 Організація робочих місць користувачів ВТД ПЕОМ

Комп'ютерна техніка, встановлена в даному приміщенні, є сучасною технікою, яка виконана з урахуванням усіх вимог щодо охорони праці. Зокрема, відеомонітори мають тип LR/NI (Low Radiation), який характеризується низьким рівнем випромінювання екрана монітору. Тип NI (Non-Interlaced) має послідовне розгорнення, що сприяє меншому стомленню очей при роботі з ВДТ. ВДТ ПЕОМ є джерелом як електромагнітних випромінювань (м’якого рентгенівського, ультрафіолетового, інфрачервоного та радіочастотного діапазону) так і електростатичного поля.

ВДТ ПЕОМ є пристроєм для візуального зображення інформації, збереженої електронним засобом. Він складається з дисплейного екрана, системного блока обробки виведеної інформації, і клавіатури.

У разі тривалої роботи за комп'ютером при неправильному, з фізіологічної точки зору, положенні тіла може викликати в організмі людини такі види захворювань, як сколіоз — дугоподібне викривлення хребта, чи остеохондроз — дистрофічний процес у кістковій та хрящовій тканині.

Частіше всього користувачі комп’ютерної техніки скаржаться на біль у руках, плечових суглобах, шиї, у верхній частині ніг та у спині. Основні симптоми захворювань, що пов’язані з постійним інтенсивним використанням клавіатури, це больові відчуття у суглобах та м'язах кистей рук, оніміння та

дуже повільна рухливість пальців, судоми м'язів кистей рук, поява ниючого болю в ділянках зап'ястка.

Саме праця за клавіатурою потребує найбільш інтенсивної динамічної роботи кістково-м’язового апарату кистей рук і одночасно статичного напруження м'язів передпліччя і плеча. Виконання однотипних, фізично неважких рухів кистей, що здаються зовсім необтяжливими для людини, можуть призвести навіть до функціональних змін в її організмі, при цьому розвиватися вони можуть непомітно протягом кількох років.

Слід зауважити, що не тільки робота за клавіатурою призводить до виникнення порушень у кістково-м'язовому апараті рук. Як вже було сказано вище, використання в роботі такого пристрою як ”миша”, також несприятливо впливає на організм користувача комп'ютерної техніки. Маніпулюючи ”мишею” людина здійснює велику кількість дрібних однотипних рухів, що призводить до постійного навантаження на кисть руки, передпліччя та плече. Все це обумовлює появу неприємних, а згодом і болісних відчуттів у ділянці зап'ястка, у ліктьовому і особливо плечовому суглобах.

Таким чином, можна констатувати, що основне перенапруження опорнорухової системи людини при роботі з комп’ютерною технікою спричиняється, в першу чергу, багатогодинною напруженою роботою в одноманітному положенні сидячи, а значить і обмеженою загальною руховою активністю (гіподинамією), а також однотипними інтенсивними циклічними навантаженнями, які мають місце при роботі з клавіатурою та з пристроєм типу "миша".

Крім того, праця користувачів персональних комп'ютерів супроводжується активізацією уваги й інших вищих психічних функцій, а також може супроводитися порушеннями режиму праці і відпочинку.

4.2.2 Електробезпека

Згідно ГОСТ 12.2.007.0-75 все електрообладнання, що знаходиться в робочому приміщені, можна віднести до 0I, I, II та III класів за електрозахистом:

– 0I клас — обладнання, що має робочу ізоляцію, елемент заземлення та провід без заземлюючої жили;

– I клас — обладнання, що приєднується до ланцюга живлення трьох контактними вилками, один з цих контактів при цьому з'єднується з заземленим контактом розетки (систем ПЕОМ);

– II клас — обладнання, що має всі доступні до дотику частини відділені від струмопровідних частин подвійною або підсиленою ізоляцією та не має контакту (затискача) для приєднання захисних провідників (ВДТ ПЕОМ);

– III клас — обладнання, що має робочу напругу не вище 42 В змінного струму або 120 В постійного струму.

– Згідно з ПУЕ робоче приміщення за ступенем небезпеки ураження людей електричним струмом можна віднести до приміщень без підвищеної небезпеки з наступних причин:

– відносна вологість повітря приміщення не перевищує 75%;

– відсутня можливість одночасного дотику людини до елементів металоконструкцій будівель, технологічних апаратів, механізмів, що мають з'єднання з землею, з одного боку та до металевих корпусів електрообладнання — з іншого боку;

– матеріал підлоги (паркет) є діелектриком;

– температура повітря не досягає значень, що перевищують 35 ˚С.

В робочому приміщенні передбачене захисне відключення напруги живлення мережі при аварійному режимі роботи обладнання. Для зменшення значення напруги дотику та відповідних їй величин струмів при нормальному і аварійному режимах роботи електрообладнання необхідно виконати повторне захисне заземлення нульового дроту.

4.2.3 Розрахунок електромережі на здатність відключення при аварійному режимі роботи електрообладнання

Струм короткого замикання визначається за формулою:

$$I_{к.з.} = \frac{U_ф}{(R_ф + R_н + Z_т)}, \quad (4.1)$$

де $U_ф$ — фазова напруга мережі, становить 220 В; $R_ф$ — опір фазового проводу, становить ≈ 1,6 Ом; $R_н$ — опір нульового проводу, становить ≈ 1,6 Ом; $Z_т$ — розрахунковий опір трансформатора, становить 0,1 Ом;

Підставимо значення у формулу (4.1), остаточно отримаємо:

$$I_{к.з.} = \frac{220}{(1,6 + 1,6 + 0,12)} \approx 66,26 \text{ А},$$

Для надійного спрацювання автоматів струмового захисту необхідно, щоб виконувалась умова:

$$\frac{I_{к.з.}}{I_{авт.макс.}} \geq 1,4, \text{ (при } I_{к.з.} < 100)$$

Мережа в робочому приміщенні обладнана автоматом струмового захисту, розрахованого на струм Iавт. = 15 А. Таким чином, струм короткого замикання при виникненні аварійної ситуації в (4,42 А) рази перевищує номінальний струм спрацювання автомату, що задовольняє встановленим нормам.

Напругу дотику до зануленого обладнання визначають за формулою:

$$U_{дот} = I_{к.з.} \cdot R_н = 66,26 \cdot 1,6 \approx 106,01 \text{ В}$$

Напруга дотику (Uдот < Uдоп = 500 В) за час спрацювання автоматів струмового захисту (t < 0,1 с) не перевищує допустимого значення, що відповідає вимогам ГОСТ 12.1.038-88.

Підключення обладнання виконано у відповідності з вимогами ПБЕ та ПУЕ. Додаткових заходів щодо підвищення рівня електробезпеки в робочому приміщені впроваджувати не потрібно.

4.2.4 Вимоги до робочих місць

Облаштування робочих місць, обладнаних ПЕОМ, ВДТ, повинно забезпечувати:

– належні умови освітлення приміщення і робочого місця, відсутність відблисків;

– оптимальні параметри мікроклімату (температура, відносна вологість, швидкість руху, рівень іонізації повітря);

– належні ергономічні характеристики основних елементів робочого місця (ДСТУ ISO 9241-6:2004).

Будівлі та приміщення, в яких експлуатуються ПЕОМ та виконуються їх обслуговування, налагодження і ремонт, повинні відповідати вимогам: СНиП 2.09.02-85 “Производственные здания”, СНиП 2.09.04-87 “Административные и бытовые здания”, “Правил устройства электроустановок”, затверджених Головдерженергонаглядом СРСР 1984 р. (ПВЕ), “Правил технической эксплуатации электроустановок потребителей”, затверджених Головдерженергонаглядом СРСР 21.12.84 (ПТЕ, СНиП 2.08.02-89 “Общественные здания и сооружения” з доповненнями, затвердженими наказом Держкоммістобудування України від 29.12.94 № 106, СН 512-78 “Инструкция по проектированию зданий и помещений для электронновычислительных машин”, затверджених Держбудом СРСР, ДСанПіН 3.3.2.-007-98 ‘‘Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин‘‘, затверджених МОЗ України 10.12.98.

Заборонено розміщувати робочі місця з ВДТ, ПЕОМ у підвальних приміщеннях, на цокольних поверхах, поряд з приміщеннями, в яких рівні шуму та вібрації перевищують допустимі значення (поряд з механічними цехами, майстернями тощо), з вибухопожежонебезпечними приміщеннями категорій А і Б, а також над такими приміщеннями або під ними.

Приміщення мають бути обладнані системами водяного опалення, кондиціонування або припливно-витяжною вентиляцією відповідно до СНиП 2.04.05-91.

Згідно з ДНАОП 0.00-1.31-99 площу приміщень визначають із розрахунку, що на одне робоче місце вона має становити не менше ніж 6 м2, а об'єм не менше ніж 20 м3 з урахуванням максимальної кількості осіб, які одночасно працюють у зміні. Приміщення являє собою кімнату довжиною □ *= 12 м* та шириною □ *= 6 м.* Висота кімнати становить *h = 4 м.* Розмір дверного прорізу становить □ *= 1,5 м*.

Площу та об’єм приміщення знайдемо за формулами:

де a — довжина приміщення; b — ширина приміщення; h — висота приміщення.

Підставивши значення в формули (4.2) та (4.3), отримаємо:

$$□ = □ \cdot □, \quad (4.2)$$

$$□ = □ \cdot h, \quad (4.3)$$

Зведемо нормативні та фактичні дані приміщення в табл 4.1.

$$□ = 12 \cdot 6 = 72 \text{ м}^2,$$

$$□ = 72 \cdot 4 = 288 \text{ м}^3,$$

Таблиця 4.1 Параметри приміщення

Назва характеристики	**Нормативне значення**	**Фактичне значення**
Площа приміщення з розрахунку на 1 особу	> 6 м2	72 м2

Таблиця 4.1 Параметри приміщення

Об’єм приміщення з ро- зрахунку на 1 особу	> 20 м3	288 м3
Висота приміщення	3,5 – 4 м	4 м
Розміри дверей (Ш х В)	≥ 1,1 х 1,8 м	1,5 х 2 м
Відстань від стіни зі світловими прорізами до ВДТ	≥ 1 м	1,5 м

На підставі отриманих результатів можна зробити висновок, що геометричні розміри приміщення цілком відповідають нормативним вимогам.

Оздоблюють стіни, стелю, підлогу приміщення з матеріалів, які дозволені органами державного санітарно-епідеміологічного нагляду. Заборонено застосовувати полімерні матеріали (деревостружкові плити, шпалери, що можна мити, рулонні синтетичні матеріали, шаруватий паперовий пластик, тощо), що виділяють у повітря шкідливі хімічні речовини. За розміщенням робочих місць з ВДТ, ПЕОМ потрібно витримувати такі відстані: від стін зі світловими прорізами не менше 1 м; між бічними поверхнями ВДТ не менше 1,2 м; між тильною поверхнею одного ВДТ та екраном іншого не менше 2,5 м; прохід між рядами робочих місць не менше 1 м. Робочі місця з ВДТ щодо світлових прорізів розміщують так, щоб природне світло падало збоку, переважно зліва. Екран ВДТ і клавіатура мають розміщуватися на оптимальний відстані від очей користувача, але не ближче 600 мм з урахуванням розміру алфавітно-цифрових знаків і символів. Розміщення екрана ВДТ має забезпечувати зручність зорового спостереження у вертикальній площині під кутом ± 30° від лінії зору працівника.

Усі вище перераховані вимоги відповідають робочому приміщенню, де проводяться дослідження.

4.2.5 Відповідність параметрів мікроклімату робочої зони санітарним нормам

Для підвищення працездатності і збереження здоров'я важливо створити для людини стабільні метеорологічні умови — мікроклімат повітряного середовища, у поняття якого входять температура, відносна вологість, швидкість руху повітря та інтенсивність теплового опромінення.

Стан повітряного середовища визначається метеорологічними умовами в лабораторії, виробничим мікрокліматом, а також запиленістю повітря і його загазованістю.

Мікроклімат виробничого приміщення визначається поєднаннями температури, вологості, швидкості руху повітря та інтенсивності теплового випромінювання, що впливають на організм людини.

Для того, щоб фізіологічні процеси в організмі людини протікали нормально, температура його тіла повинна бути постійною. Надлишкова теплота повинна виділятися в навколишнє середовище. Відповідність між цією теплотою та охолоджувальною здатністю навколишнього середовища визначає комфортні умови.

При відхиленні параметрів мікроклімату від комфортних в організмі людини відбуваються процеси, спрямовані на терморегулювання. Розрізняють хімічну і фізичну терморегуляцію організму. Хімічна відбувається зниженням рівня обміну речовин, а фізична — за допомогою теплопровідності, конвекції, випромінювання і випаровування.

Робочою зоною вважається простір, обмежений конструкціями виробничих приміщень, що мають висоту 2 м над рівнем підлоги чи площадки, на яких знаходяться місця постійного чи непостійного перебування працюючих. Склад повітря робочої зони залежить від складу атмосферного повітря і впливу на нього виробничих факторів, що утворюються в процесі трудової діяльності людини.

Для нормальної життєдіяльності людини важливий парціальний тиск кисню — 21331 Па (160 мм рт. ст.) і необхідно, щоб у повітрі приміщень містилось не менш 19,5 – 20% кисню.

Вміст шкідливих речовин у повітрі робочої зони не має перевищувати ГДК. Відповідно до ГОСТ 12.1.005-88 вміст озону — не більше 0,1 мг/м3, вміст оксидів азоту — не більше 5 мг/м3, вміст пилу — не більше 4 мг/м3.

Важливо, щоб повітря мало визначений іонний склад. У повітрі містяться негативні і позитивні іони, що по рухливості розділяють на легкі, середні і важкі. На життєдіяльність організму людини благотворно впливають негативні іони кисню. Вміст легких іонів у повітрі виробничих і громадських приміщень, повітряне середовище яких піддається спеціальній обробці в системах кондиціонування, приведено в табл. 4.2. Іонний склад повітря має відповідати вимогам СН 2152-80.

Метеорологічні умови повітря робочої зони регламентує ГОСТ 12.1.005-88 та ДСН 3.3.6.042-99.

Таблиця 4.2 Припустимий вміст легких іонів у повітрі робочої зони

Рівень	**Кількість іонів у 1 см^3 повітря**		**Коефіцієнт полярності, π**
	Позитивних, n$^+$	**Негативних, n$^-$**	
Мінімальний	400	600	-0,2
Оптимальний	1000 – 3000	3000 – 5000	-0,67 – 0
Максимально припустимий	50000	50000	-0,05 – +0,05

Норми визначаються категорією тяжкості праці, періодом року. Роботи ведуться в лабораторії, виконуються в сидячому положенні та не потребуються значного фізичного напруження, тому вони відносяться до категорії Іа (споживання енергії до 138 Дж/год., або 120 ккал/год.).

Згідно ГОСТ 12.1.005-88 та ДСН 3.3.6.042-99 метеорологічні умови в даній лабораторії повинні знаходитися у межах, вказаних у табл. 4.3.

Таблиця 4.3 Параметри мікроклімату в робочій зоні розробника

Період року	Категорія робіт	Температура, ˚С			Відносна вологість повітря		Швидкість руху повітря, м/с	
		Фактична	Припустима на робочих місцях		Фактична	Припустима на постійному та непостійному робочих місцях	Фактична	Припустима на постійному і непостійному робочих місцях
			Оптимальна	Допустима				
Холодний	Ia	22-24	21-25	18-26	40-60	75	0,1	не більше 0,1
Теплий	Ia	23-25	22-28	20-30	40-60	55 при 28 ˚С	0,1	0,1-0,2

Температура в приміщенні, де проводять роботу, може перевищувати граничну припустиму температуру через джерела тепла. Для видалення надлишку тепла використовується місцева механічна вентиляція. У теплий період року використовується природна вентиляція через віконні прорізи. Це дозволяє знизити температуру до оптимального значення. У холодний період року може застосовуватись система центрального опалення у відповідності з вимогами СНиП 2.04.05-91 та ГОСТ 12.4.021-75.

4.2.6 Відповідність освітлення на робочих місцях санітарним нормам

Перевіримо освітленість, що забезпечується загальним штучним освітленням. Для визначення освітленості застосовується метод коефіцієнта використання світлового потоку. Загальне штучне освітлення в робочому

приміщенні забезпечується за допомогою люмінесцентних ламп ЛБ-80 потужністю 80 Вт, а місцеве — за допомогою світильників з лампами накалювання потужністю 60 Вт.

Фактичне освітлення робочих місць штучним освітленням визначається

$$\square_{ф} = \frac{\square \cdot \square \cdot Ф}{\square \cdot \square \cdot \square} \cdot \square, \qquad (4.4)$$

за формулою:

де N — кількість світильників, N = 8; n — кількість ламп у світильнику, n = 2; η — коефіцієнт використання світлового потоку; S — площа приміщення, S = 72 м2; K — коефіцієнт запасу, K = 1,5; Z — коефіцієнт нерівномірності освітлення; Ф — світловий потік світильника, Ф = 5200 Лм.

Для визначення коефіцієнту використання світлового потоку визначимо індекс приміщення і та коефіцієнт відбиття стелі ρ1, стін ρ2 та робочої поверхні ρ3:

$$\square_{к.з.} = \frac{a \cdot b}{h \cdot (a \cdot b)}$$

де a — довжина приміщення, м; b — ширина приміщення, м; ha — висота підвісу світильника, м.

$$\square = \frac{12 \cdot 6}{4 \cdot (12 + 6)} = 1$$

Відповідно до індексу приміщення, коефіцієнт відбиття побіленої стелі становить ρ1 = 0,7; побілених стін — ρ2 = 0,6; середніх робочих поверхонь — ρ3 = 0,1. Тоді, коефіцієнт використання світлового потоку становить η = 0,5.

Підставивши отримані результати в формулу (4.4), отримаємо:

$$\square_{ф} = \frac{8 \cdot 2 \cdot 5200}{72 \cdot 1,5 \cdot 1,1} 0,5 = 350 \text{ лк}$$

Штучне освітлення в приміщеннях регламентується нормами

ДБН В.2.5-28-2006. Для зорової роботи категорії 3в (середній контраст розрізнення об'єкту, фон - середній, робота високої точності) при загальному освітленні норма становить 300 лк. Оскільки фактичне значення освітленості робочого місця Еф більше норми, то вимоги ДБН В.2.5-28-2006 виконуються.

4.2.7 Заходи щодо нормалізації умов праці

Для усунення шкідливого впливу НРВ на організм інженерів-програмістів можна рекомендувати скоротити час перебування за екраном дисплея до 4 годин у зміну, причому після 2 годин безупинної роботи радиться 30-хвилинна перерва (бажано на відкритому повітрі).

З метою автоматичної підтримки параметрів мікроклімату в необхідних межах протягом всіх сезонів року, очищення повітря від пилюки і шкідливих речовин, зниження рівня іонізації в помешканні лабораторії провадиться вентиляція за допомогою двох витяжних вентиляторів типу АИСИ-4 і щоденне вологе прибирання.

Світильники розташовані в два ряди і під’єднані до різних фаз електромережі для усунення мерехтіння світлового потоку (коефіцієнт пульсацій менше 10%).

Для ослаблення шкідливого впливу електростатичних полів у лабораторії застосовуються захисні скляні фільтри (екрани) з електропровідним покриттям, що має відвід для заземлення, що прикріплюються на екран монітора.

Для зниження рівня шуму в лабораторії використовується шумопоглинаюче облицювання з перфорованим покриттям: гіпсові плити товщиною 7-9 мм із заповненням із склотканини.

4.3 Безпека в надзвичайних ситуаціях

Безпека в НС регламентується ПЛАС (План Ліквідації Аварійних Ситуацій). Основними складовими частинами ПЛАС є розробка технічних

рішень та організація заходів щодо оповіщення, евакуації та дій персоналу при виникненні НС, а також визначення основних заходів з пожежної безпеки.

4.3.1 Вимоги щодо організації ефективності роботи систем оповіщення персоналу у разі виникнення надзвичайної ситуації

Розміри зон оповіщення, черговість оповіщення та час початку оповіщення людей в окремих зонах визначаються, виходячи з умов забезпечення безпечної та своєчасної евакуації людей у разі виникнення НС.

Оповіщення про НС та управління евакуацією людей здійснюється одним з наступних способів або їх комбінацією:

– подання звукових і (або) світлових сигналів в усі виробничі приміщення будівлі з постійним або тимчасовим перебуванням людей;

– трансляція текстів про необхідність евакуації, шляхи евакуації, напрямок руху й інші дії, спрямовані на забезпечення безпеки людей;

– трансляція спеціально розроблених текстів, спрямованих на запобігання паніці й іншим явищам, що ускладнюють евакуацію;

– розміщення знаків безпеки на шляхах евакуації згідно з ДСТУ ISO 6309;

– ввімкнення евакуаційних знаків «Вихід»;

– ввімкнення евакуаційного освітлення та світлових покажчиків напрямку евакуації.

Кількість оповіщувачів, їх розміщення та потужність повинні забезпечувати необхідну чутність у всіх місцях постійного чи тимчасового перебування людей.

У місцях, де є небезпека механічного ушкодження оповіщувачів, повинен бути забезпечений їх захист, що не порушує працездатності оповіщувачів. Встановлення звукових та мовних оповіщувачів у виробничих приміщеннях повинно виключати можливість концентрації та нерівномірного розподілу звуку.

Оповіщувачі повинні підключатися до мережі без роз'ємних пристроїв і не мати регуляторів гучності. Сигнали оповіщення про пожежу повинні відрізнятися від сигналів іншого призначення. Комунікації системи оповіщення людей про пожежу можуть проектуватися суміщеними з радіотрансляційної мережею будівлі.

Електропостачання, заземлення, занулення, вибір та прокладання мереж оповіщення приймаються згідно з вимогами до систем пожежної сигналізації за ДБН В.2.5-56-2014 «Системи протипожежного захисту».

Управління системою оповіщення слід передбачати з приміщення пожежного поста, диспетчерської або іншого спеціального приміщення. Вимоги до такого приміщення приймаються за аналогією з вимогами до приміщень чергового персоналу з ДБН В.2.5-56-2010.

4.3.2 Обов'язки та дії персоналу у разі виникнення надзвичайної ситуації

У разі виявлення ознак НС працівник, який їх помітив, повинен:

- повідомити про НС керівника, адміністрацію, пожежну охорону підприємства;
- організувати оповіщення людей про НС;
- вжити заходів щодо евакуації людей та матеріальних цінностей;
- вжити заходів щодо ліквідації наслідків НС з використанням наявних засобів.

Керівник та пожежна охорона установки, яким повідомлено про виникнення пожежі, повинні:

- перевірити, чи викликані підрозділи ДСНС;
- вимкнути у разі необхідності струмоприймачі та вентиляцію;
- у разі загрози життю людей негайно організувати їх евакуацію, та їх рятування, вивести за межі небезпечної зони всіх працівників, які не беруть участь у ліквідації НС;
- перевірити здійснення оповіщення людей про НС;

– забезпечити дотримання техніки безпеки працівниками, які беруть участь у ліквідації НС;

– організувати зустріч підрозділів ДСНС, надати їм допомогу у локалізації і ліквідації НС.

Після прибуття на НС підрозділів ДСНС повинен бути забезпечений безперешкодний доступ їх до місця, де виникла НС.

4.3.3 Пожежна безпека

В лабораторії знаходиться значна кількість твердих горючих речовин та матеріалів (дерев'яні меблі, пластмасові вироби, гума, папір). Згідно НАПБ Б.03.002-2007 норм визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою, науково-дослідницька лабораторія відноситься до пожежонебезпечних приміщень категорії В (тверді горючі та важкогорючі речовини і матеріали, речовини і матеріали, які при взаємодії з водою, киснем, повітрям або один з одним здатні тільки горіти).

Згідно з класифікацією робочих зон відповідно до НПАОП 40.1-1.32-01 науково-дослідницька лабораторія відносяться до зон класу П-IIа — пожежонебезпечні, що містять тверді горючі речовини, які нездатні переходити у зважений стан.

Джерелами загоряння можуть бути електричні іскри, коротке замикання, перевантаження електропроводки, несправність апаратури, паління в приміщенні. Тому, для запобігання пожежі в приміщенні проводяться пожежнопрофілактичні заходи: застосування запобіжників в електричних мережах, використання пилонепроникних сполучних та розподільних коробок, а також, проводиться інструктаж з техніки пожежної безпеки.

Відповідно до ДСТУ 3675-98 та ISO 3941-77 в лабораторії знаходяться два вогнегасники: вуглекислотний типу ВВ-5 та порошковий типу ВП-2. Вогнегасник ВВ-5 розташований на висоті 1,5 м від підлоги поруч із виходом з приміщення.

У коридорі знаходяться коробки, у яких знаходяться пожежний кран і рукав, а також знаходиться вогнегасник типу ВВ-5.

Окрім цього, в лабораторії є план евакуації у випадку виникнення пожежі. Максимальна віддаленість робочих місць від евакуаційних виходів та ширина евакуаційних проходів відповідають вимогам СНиП 2.02.02-85, СНиП 2.01.02-85 та ДБН В.1.1.7-2002.

У робочому приміщенні виконані всі вимоги «Правил пожежної безпеки України».

Таким чином, у науково-дослідницькій лабораторії забезпечуються всі необхідні технічні та організаційні заходи з пожежної безпеки.

5 РОЗРОБКА СТАРТАП ПРОЕКТУ

Розробка стартап проекту є неможливою на даному етапі розробки, так як результат магістерської дисертаційної роботи не я продуктом чи послугою.

ВИСНОВКИ

1.Насьогодні існує безліч незалежних систем інтернету речей, які переважно побудовані з використанням застарілих технологій або з використанням монолітних архітектурних рішень. Більшість систем є вузькоспеціалізовані, що ускладнює інтеграцію їх між собою. Це спричинює підвищення вартості обслуговування та розробки.

2.Системи інтернету речей розробляються з використанням різноманітних протоколів даних. Протокол передачі даних NB-IoT являється універсальним для використання при розробці систем інтернету речей, але знаходиться на стадії розробки. Добре відомі MQTT, CoAP та HTTP з підходом RESTful являються найкращими кандидатами для експлуатації на даний час.

3.Використання технологій контейнеризації окремих модулів чи цілих систем, дозволить розробникам та кінцевим користувачам без проблем працювати на будь-якій платформі та з будь-якою мовою програмування.

4.Запропонована уніфікована артіхектура може бути використана для досягнення бізнес цілей у промисловості чи в будь-якій сфері життєдіяльності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Интернет вещей — а что это? [Електронний ресурсї — Режим доступу: https://habr.com/post/149593/ — Назва з екрану.

2. Arvind Ravulavaru. Practical Internet of Things with JavaScript. 2017 Packt Publishing ISBN 978-1-78829-294-8

3. Excerpt From: Arvind Ravulavaru. “Practical Internet of Things with JavaScript.” Apple Books.

4. Pooja, Asmita Pandey, Virtual machine performance measurement, Engineering and Computational Sciences (RAECS), 2014 Recent Advances in, IEEE, ISBN 978-1- 4799-2290-1.

5. Theodora Adufu, Jieun Choi,Yoonhee Kim, Is container- based technology a winner for high performance scientific applications? , Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific, IEEE, INSPEC Accession Number:15487467

6. Ren´ ePeinl·FlorianHolzschuher·FlorianPfitzer, Docker Cluster Management for the Cloud - Survey Results and Own Solution, Journal of Grid Computing, Volume 14 Issue 2, 2016, ISSN: 1570-7873

7. IoT архитектура — первый взгляд под капот [Електронний ресурс] — Режим доступу: https://habr.com/post/420173/ — Назва з екрану.

8. Taras Maksymyuk, Mykola Brych, Stepan Dumych, Haider Al-Zayad, Comparison of the IoT Transport Protocols Performance over Narrowband-IoT Networks, Internet of Things and Ubiquitous Communications, June 2017.

9. Сети ZigBee. Зачем и почему? [Електронний ресурс] — Режим доступу: https://habr.com/post/155037/ — Назва з екрану.

10. Profile of an IoT Developer: Results of the IoT Developer Survey [Електронний ресурс] — Режим доступу: https://ianskerrett.wordpress.com/2016/04/14/profile-of-an-iot-developer-results-of-the-iot-developer-survey/ — Назва з екрану.

11. About Docker Engine [Електронний ресурс] — Режим доступу: https://docs.docker.com/engine/ — Назва з екрану.

12. Operating-system-level virtualization [Електронний ресурс] — Режим доступу: https://en.wikipedia.org/wiki/Operating-system-level_virtualization — Назва з екрану.

13. The top 6 programming languages for IoT projects [Електронний ресурс] — Режим доступу: https://techbeacon.com/top-6-programming-languages-iot-projects— Назва з екрану.

14. HTML: HyperText Markup Language [Електронний ресурс] — Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML — Назва з екрану.

15. The Constrained Application Protocol (CoAP) [Електронний ресурс] — Режим доступу: https://tools.ietf.org/html/rfc7252 — Назва з екрану.

16. Information technology -- Message Queuing Telemetry Transport (MQTT) v3.1.1 [Електронний ресурс] — Режим доступу: https://www.iso.org/standard/69466.html — Назва з екрану.

17. Constrained Application Protocol for Internet Of Things [Електронний ресурс] — Режим доступу: https://www.cse.wustl.edu/~jain/cse574-14/ftp/coap/index.html — Назва з екрану.

18. Internet Of Things & Augmented Reality Emerging Technologies. [Електронний ресурс] — Режим доступу: https://www.coursera.org/learn/iot-augmented-reality-technologies — Назва з екрану.

19. Information Technology-Open Systems Interconnection-Basic Reference Model: The Basic Model, (ISO), I. O., & Iec, I.1994.

20. Fremantle, A Reference Architecture for The Internet of Things. London: WS02. P. 2015.

21. Mario Casciaro, Luciano Mammino. Node.js Design Patterns, 2 edition. 2014 Packt Publishing Ltd. ISBN 978-1-78588-558-7

22. Max Kanat-Alexander. Understanding Software. 2017 Packt Publishing Ltd. ISBN 978-1-78862-881-5

23. Sam Newman. Building Microservices. O'Reilly Media; 1 edition, 2015. P 282.

24. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. 2013. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions.

25. Fowler M., Lewis J. Microservices: A Definition of This New Architectural Term, 2014.

26. K. H. 2015. Observing Resources in the Constrained Application Protocol (CoAP). (Internet Engineering Task Force (Ietf)).

27. Vinicius Feitosa Pacheco. Microservice Patterns and Best Practices. 2018 Packt Publishing Ltd.

28. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. 2013. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions.

29. Chase, J. The Evolution of The Internet of Things. Texas Instruments Incorporated. 2013