Міністерство освіти і науки України

ІНТЕРНЕТ РЕЧЕЙ В ЕЛЕКТРОНІЦІ

Конспект лекцій
для здобувачів другого (магістерського) рівня вищої освіти
освітньої програми «Електроніка»
галузі знань 17 Електроніка, автоматизація та електронні комунікації
спеціальності 171 Електроніка
денної та заочної форм навчання

Луцьк 2025

УДК 004.738.5 (07)
І 73

Рекомендовано до видання вченою радою факультету КІТ ЛНТУ,
протокол № _____ від «___» _____ 20 25 року.

Голова вченої ради факультету КІТ _____ Інна КОНДІУС

Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ

Директор бібліотеки _____ Наталія ПОЛІЩУК

Розглянуто і схвалено на засіданні кафедри електроніки та телекомунікацій
ЛНТУ, протокол № _____ від « _____» _____ 20 25 року.

Завідувач кафедри ЕіТК _____ Валентин ЗАБЛОЦЬКИЙ

Укладач: _____ Анатолій ТКАЧУК, кандидат технічних наук,
доцент кафедри електроніки та телекомунікацій ЛНТУ

Рецензент: _____ Сергій МОРОЗ, кандидат технічних наук, доцент
кафедри електроніки та телекомунікацій ЛНТУ

Відповідальний за випуск: _____ Валентин ЗАБЛОЦЬКИЙ, кандидат
технічних наук, доцент, завідувач кафедри електроніки та телекомунікацій ЛНТУ

І 73 **Інтернет Речей в електроніці**: конспект лекцій для здобувачів другого (магістерського) рівня вищої освіти освітньої програми «Електроніка» галузі знань 17 Електроніка, автоматизація та електронні комунікації спеціальності 171 Електроніка денної та заочної форм навчання / уклад. А.А. Ткачук. Луцьк: ЛНТУ, 2025. 64 с.

Конспект лекцій з дисципліни «**Інтернет Речей в електроніці**»: складений відповідно до діючої програми курсу.

Призначений для здобувачів вищої освіти спеціальності 171 Електроніка освітньої програми «Електроніка».

ЗМІСТ

ТЕМА 1
ІСТОРІЯ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Історія розвитку Інтернету Речей

Термін «інтернет речей» зобов'язаний своєю появою Кевіну Ештону, який у 1997 р., працюючи на компанію Proctor and Gamble, для управління системою постачання застосував технологію радіочастотної ідентифікації (RFID). Завдяки цій роботі в 1999 р. його запросили до Массачусетського технологічного інституту, де він з групою однодумців організував дослідницький консорціум Auto-ID Center (докладнішу інформацію можна знайти на сайті www.smithsonianmag.com/innovation/kevinashton-describes-the-internet- of-things-180953749/). З того часу інтернет речей здійснив перехід від простих радіочастотних міток до екосистеми та індустрії, яка до 2020 р. залучила 5 трлн доларів зі 100 трлн світового ВВП, тобто 6% світового ВВП. Аж до 2012 р. ідея підключення речей до інтернету переважно ставилася до смартфонів, планшетів, ПК та ноутбуків. По суті, до тих речей, які у всіх відносинах виступають як комп'ютери. До цього, з появою перших зачатків інтернету (таких як створена 1969 р. мережа ARPANET), більшості технологій, у яких будується інтернет речей, просто не існувало. До 2000 р. більшість пристроїв, які можна було підключити до інтернету, були комп'ютерами різних розмірів. Таблиця 1.1 демонструє поступове підключення до Інтернету.

Таблиця 1.1 – Історія Інтернету Речей

Рік	Пристрій	Джерело
1973	Маріо У. Кардулло отримує патент на першу радіочастотну мітку	США, патент US 3713148 A
1982	Підключений до інтернету автомат із газованою водою в університеті Карнегі-Меллон	www.cs.cmu.edu/~coke/history_long.txt
1989	Підключений до Інтернету тостер на конференції Interop '89	Журнал IEEE Consumer Electronics Magazine
1991	Компанія HP представила HP LaserJet IIISi: перший підключений до мережі Ethernet мережевий принтер	hpmuseum.net/display_item.php?hw=350
1993	Підключена до інтернету кавоварка у Кембриджському університеті (перша підключена до інтернету камера)	www.cl.cam.ac.uk/coffee/qsf/coffee.html
1996	Підрозділ General Motors OnStar (дистанційна діагностика 2001)	en.wikipedia.org/wiki/OnStar
1998	Поява організації Bluetooth SIG	www.bluetooth.com/aboutus/our-history
1999	Холодильник LG Internet Digital DIOS	www.telecompaper.com/news/lg-unveils-internetready-refrigerator--221266
2000	Перші прояви розробленої компанією HP концепції всепроникної комп'ютеризації (Cooltown): HP Labs, система обчислювальних та комунікаційних технологій, що у поєднанні один з одним створюють підключення до інтернету для людей, місць та об'єктів	www.youtube.com/watch?v=U2AkkuIVV-I
2001	Випуск першого пристрою, що використовує технологію Bluetooth: мобільний телефон KDDI із підтримкою Bluetooth	edition.cnn.com/2001/BUSINESS/asia/04/17/tokyo.kddibluetooth/index.html
2005	Міжнародна спілка електрозв'язку, спеціалізована установа ООН, випустила звіт, в якому вперше були сформульовані прогнози розвитку інтернету речей	www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf
2008	Поява першої IoT-спільноти IPSO Alliance, метою якої було сприяння підключенню речей до інтернету	www.ipso-alliance.org
2010	Розробка напівпровідникових світлодіодних ламп призвела до розвитку концепції розумного освітлення	www.bu.edu/smartlighting/files/2010/01/BobK.pdf
2014	Компанія Apple створила протокол iBeacon для маячків	support.apple.com/ru-ua/HT202880

Поняття «Інтернет Речей» викликає велику цікавість та пильну увагу. Це легко помітити, хоча виходячи з того, що, починаючи з 2010 р., кількість одержуваних патентів бурхливо зростає (www.uspto.gov). Кількість пошукових запитів у системі Google (trends.google.com/trends/) та публікацій у колегіально рецензованому журналі IEEE різко поповзла вгору з 2013 р. (див. рис. 1.1).

1.2 Перспективи розвитку Інтернету Речей

Інтернет речей захопить практично кожен сегмент у сфері промисловості, бізнесу, охорони здоров'я та споживчих товарів. Важливо розуміти наслідки, а також те, чому ці різні галузі будуть змушені змінити свій підхід до виробництва товарів і надання послуг. Ймовірно, ви як архітектор матимете справу з якимось одним конкретним сегментом, проте вам не завадить розуміння того, як різні сфери економіки можуть взаємно впливати одна на одну в інших випадках сфери послуг, галузі промисловості та торгівлі до 2020 р. торкнулися своїм впливом від трьох (ARM Ltd. 01-1996-00-00-00-01-30-09/ARM-_2D00_-The route-to-a-trillion-devices-_2D00_-June-2017.pdf) до чотирьох відсотків (The In ternet of Things: Mapping Value Beyond the Hype, McKinsey and Company 2015: www.mckinsey.com/~/media/McKinsey/Business%20 20the%20physical%20world/Unlocking_the_potential_of_the_Internet_of_Things_Executive_summary.ash x) світового ВВП.

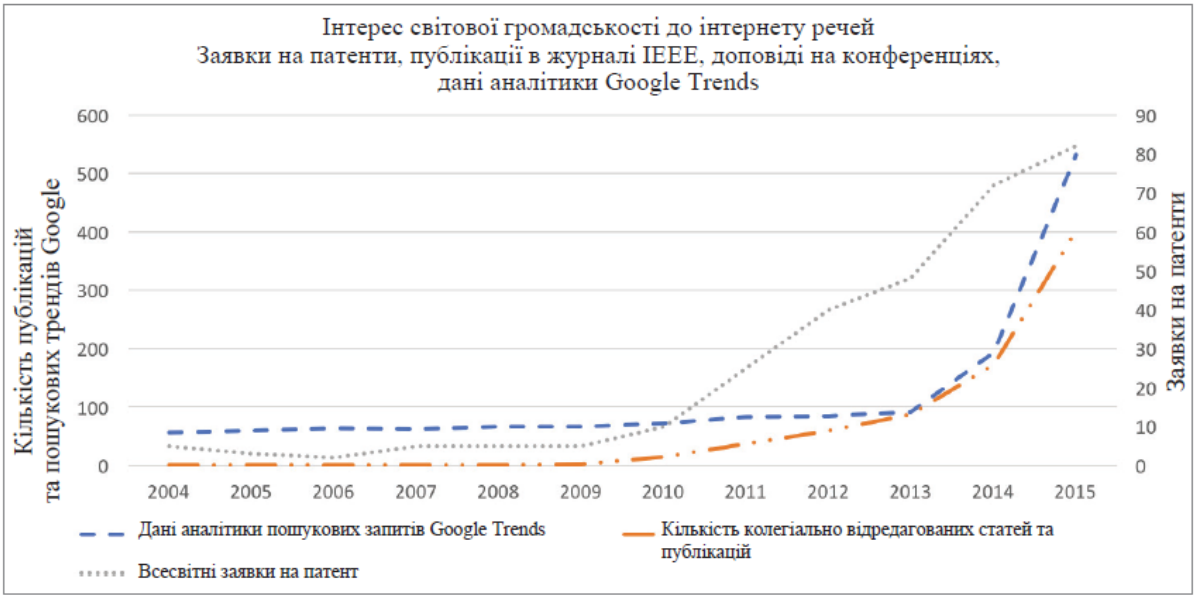


Рисунок 1.1 – Аналіз ключових слів при пошуку інформації про Інтернет Речей, патенти та технічні публікації

Чисельність взаємозалежних об'єктів безпрецедентна. Розмірковуючи про розвиток цієї сфери, неможливо не задуматися про поєднані ризики. Щоб спробувати згладити можливі наслідки, візьмемо кілька дослідницьких компаній та їх звітів про те, скільки об'єктів буде підключено до 2020 р. Розкид дуже великий, проте порядок величин приблизно однаковий. У середньому, згідно з цими 10 аналітичними прогнозами, до 2020…2025 років буде 33,4 млрд. підключених до Інтернету об'єктів. Нещодавно корпорація ARM провела дослідження та передбачила, що до 2035 р. підключеним до інтернету буде 1 трлн пристроїв. Зважаючи на все, відповідні проекти в найближчому майбутньому будуть розвиватися та нарощувати свій потенціал зі швидкістю 20% на рік (рис. 1.2).

Якщо за основу взяти консервативну точку зору, відповідно до якої до Інтернету буде підключено лише 20 млрд пристроїв (за винятком традиційної обчислювальної техніки та мобільних пристроїв), вийде, що до Інтернету кожної секунди підключатимуться 211 нових об'єктів. Для електронної промисловості та сфери інформаційних технологій ці дані мають велике значення, оскільки щорічний приріст населення Землі зараз становить приблизно 0,9…1,09% (esa.un.org/unpd/wpp/).

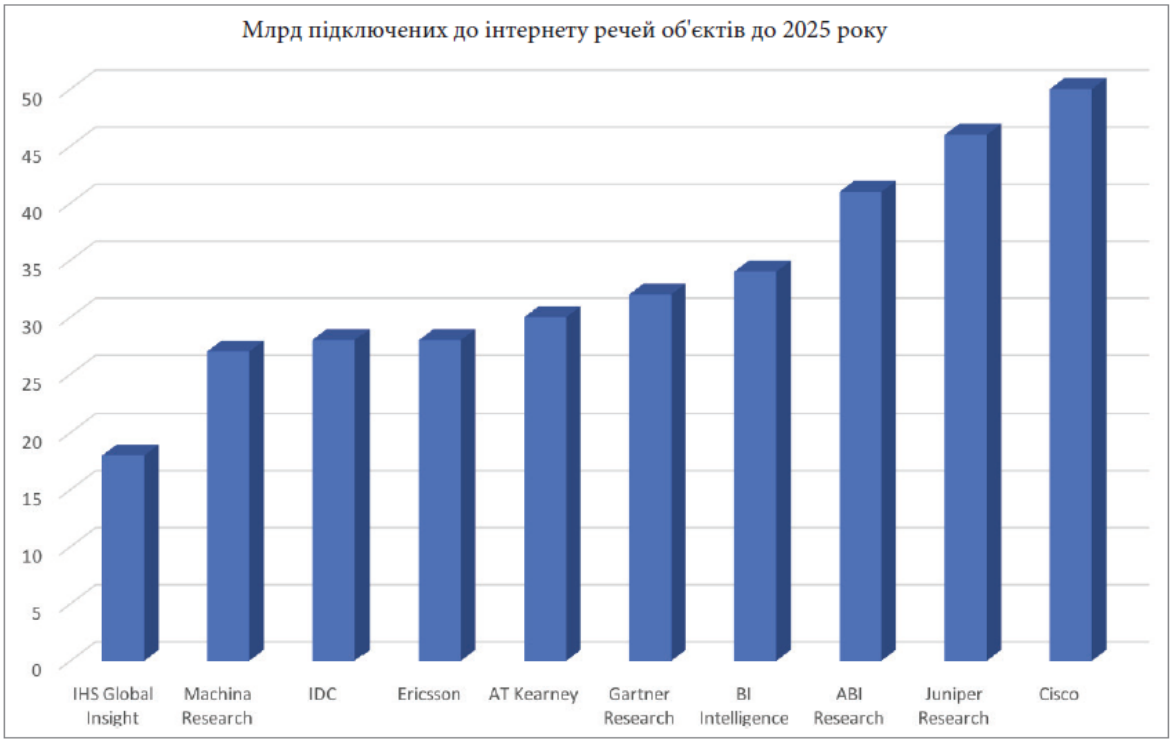


Рисунок 1.2 – Кількість підключених до Інтернету об’єктів за оцінками різних аналітиків та корпорацій

Темп зростання населення Землі досяг свого піку в 1962 р., коли він становив 2,6% на рік, і відтоді під впливом низки факторів повільно знижується. Перший і основний фактор – поліпшення економічних показників і підвищення світового ВВП негативно позначилися на народжуваності. До інших факторів належать війни та голод. Ця тенденція передбачає, що кількість об’єктів, пов’язаних з людьми, перестане зростати, а основний обсяг підключених до Інтернету пристроїв становитимуть підключені до Інтернету об’єкти та об’єкти з міжмашиною комунікацією. Це важливо, оскільки у сфері інформаційних технологій головним чинником цінності мережі є кількість розміщених у ній даних та кількість підключень. Саме так говорить закон Меткалфа. Також слід зазначити, що після того, як у 1990 р. організація CERN запустила перший інтернет-сайт, кількість користувачів мережі Інтернет зросла до 1 млрд людей лише за 15 років. Інтернет речей, за оцінками, зростатиме зі швидкістю 6 млрд підключених пристроїв на рік. Це, звісно, стане величезним чинником впливу (рис. 1.3).

Рисунок 1.3 – Дисбаланс між зростанням чисельності населення Землі та зростанням кількості підключених до Інтернету Речей. Спостерігається така тенденція: щорічний приріст підключених до Інтернету об’єктів становить 20% проти 0,9% щорічного приросту населення. Люди більше не будуть основним показником пропускної спроможності мережі та успішності IT-проекту

Слід зазначити, що з економічної погляду зміниться як спосіб отримання доходу. Вплив Інтернету Речей або будь-якої іншої технології проявляється у вигляді:

– нових джерел доходу (отримання електроенергії екологічно чистим методом);

– скорочення витрат (догляд за пацієнтами вдома);

– скорочення терміну виведення товару ринку (автоматизація виробництва);

– удосконалення структури ланцюжка поставок (облік матеріальних активів);

– скорочення виробничих втрат (крадіжка, псування товарів з коротким терміном придатності);

– підвищення продуктивності (машинне навчання та аналіз даних);

– витіснення (розумний термостат Nest витісняє з ринку звичайні термостати).

Насамперед необхідно пам’ятати про ту додаткову цінність, яку привносять IoT-рішення. Якщо це просто новий гаджет, обсяг ринку буде обмежений. Напрямок буде розвиватися і приносити хороші результати, тільки якщо очікувані переваги переважають можливі витрати. Загалом, цільова технологія повинна бути на п’ять порядків кращою за звичайну технологію. Прикидаючи витрати, необхідні для внесення змін, навчання, поширення, технічну підтримку та ін., необхідно виходити з принципу 5-кратного покращення. Далі розглянемо окремі галузі промисловості та те, як на них вплине Інтернет Речей.

1.3 Індустрія та виробництво

Промисловий Інтернет Речей (Industrial IoT, IIoT) – це один з найбільших сегментів інтернету речей, що швидко розвиваються, з точки зору кількості підключених пристроїв і ступеня корисності цих сервісів для виробництва та автоматизації підприємств. Цей сегмент зазвичай служить операційно-технологічної базою. Сюди входять апаратні та програмні засоби моніторингу фізичних пристроїв. Традиційні завдання інформаційних технологій вирішуються інакше, ніж операційно-технологічні завдання. Операційні технології (IIoT) зосереджені на оцінці продуктивності, часу безвідмовної роботи, зборі даних та реакції у відповідь в режимі реального часу, а також безпеки систем. Інформційні технології спрямовані на безпеку, групування, сервіси та надання даних. Оскільки інтернет речей починає займати важливе місце у сфері виробництва та промисловості, освіти IT та OT об’єднаються, особливо в галузі діагностичного обслуговування тисяч виробничих машин та верстатів, та зможуть забезпечувати безпрецедентним обсягом даних приватні та публічні хмарні інфраструктури. До характеристик цього сегмента належить необхідність надавати операційно-технологічній системі готові рішення у режимі реального часу або майже в режимі реального часу. Це означає, що у всьому, що стосується виробничого цеху, головним параметром для Інтернету Речей буде час відгуку. Крім того, найважливішу роль відіграватимуть тривалість простою та безпека. Це має на увазі потребу в запасі потужності і, ймовірно, у наявності приватних хмарних мереж та сховищ даних. Промисловий Інтернет Речей – це один з сегментів, що найбільш швидко розвиваються, на цьому ринку. Важливою особливістю цього напряму є те, що воно спирається на старі технології, тобто на апаратні та програмні засоби, які не можна назвати актуальними. Часто 30-річні верстати працюють на серійних інтерфейсах RS485, а не на сучасній бездротовій комірчастій архітектурі. Приклади та результати застосування промислового Інтернету Речей.

Приклади та результати застосування промислового інтернету речей включають наступне:

– профілактичне обслуговування нового та використовуваного раніше промислового обладнання;

– зростання продуктивності завдяки попиту в реальному часі;

– енергозбереження;

– системи безпеки, такі як вимірювання температури, замір тиску та контроль над витіканням газу;

– експертна система для виробничого цеху.

1.4 Споживач

Споживчі пристрої були однією з перших категорій предметів, що підключаються до Інтернету. Споживчий інтернет речей розпочався із підключеної до інтернету кавоварки в одному

університеті у 1990-х роках. Він розквітнув з поширенням технології Bluetooth на початку 2000-х років. Тепер мільйони будинків оснащені термостатами, світлодіодними лампочками, віртуальним голосовим помічником та ТВ-приставками. Крім того, люди користуються браслетами та іншими портативними пристроями. Споживчий ринок зазвичай першим переймає нові технології. Також ми можемо розглядати ці пристрої як гаджети. Всі вони поставляються в акуратній упаковці та обгортці, і, в основному, всі вони діють за принципом «встанови та увімкни».

Одна із складностей споживчого сегменту полягає у біфуркації стандартів. Наприклад, ми бачимо, що в основі деяких протоколів бездротової персональної мережі лежать стандарти Bluetooth, Zigbee та Z-wave (які не є інтероперабельними).

Цей напрямок також має дуже багато спільного з медичним сегментом, куди відносяться спеціалізовані портативні пристрої та домашні системи спостереження за станом здоров'я. Відзначимо, що медичний сегмент розвиватиметься і не обмежуватиметься простими домашніми приладами медичної діагностики (наприклад, функціоналом браслетів).

Приклади застосування Інтернету Речей:

– розумні пристрої для будинки: система поливу, гаражні двері, замки, ліхтарі, термостати та система охорони;

– портативне пристрої: трекери здоров'я та руху, розумний одяг / аксесуари;

– тварини: системи відстеження місцезнаходження домашніх тварин, розумні двері для собак.

Література: [1, 2, 3, 4, 9, 10].

ТЕМА 2
ДАТЧИКИ, КІНЦЕВІ ТОЧКИ ТА СИСТЕМИ ЖИВЛЕННЯ

Інтернет речей (IoT) починається із джерел даних або виконавчих пристроїв. Це називається кінцевими точками і, маючи вихід в Інтернет, вони можуть бути об'єднані в єдину мережу. Під час обговорення IoT загалом розгляд фактичних джерел даних часто ігнорується. Що таке ці джерела? Це датчики, а дані, які вони надають, утворюють розподілені потоком даних. Для таких потоків необхідно забезпечити можливість передачі, аналізу та збереження інформації. Цінність IoT у цьому, що це комплексне рішення, а дані, надані датчиком, грають у цьому комплексі ключову роль. Таким чином, проектувальнику необхідно розуміти, що це за дані і як їх правильно інтерпретувати. Крім розуміння того, які дані збираються і як вони утворюються в масиві IoT, корисно знати, що саме та в яких межах вимірюється. Проектувальник повинен розуміти причини, через які дані, отримані від датчиків, можуть бути ненадійними, а також причини, внаслідок яких польовий датчик може вийти з ладу. По суті, ми поєднуємо аналоговий світ із цифровим. Більшість аналогових пристроїв, що інтегруються в цифровий простір, це датчики, тому важливо розуміти їх роль та значення. Кількість датчиків та виконавчих пристроїв, об'єднаних в єдину мережу, значно зросла, тому проектувальнику важливо розуміти їхню взаємодію. Кожен повинен запитати себе: «Який тип датчика або кінцевого пристрою слід використовувати для вирішення проблеми, яка стоїть переді мною?» При розгортанні IoT необхідно враховувати множину аспектів: вартість, опціональність, розміри, тривалість безаварійної роботи та точність вимірів. Крім того, у літературі з IoT потужність та енергія, що споживаються периферійними пристроями, розглядаються рідко, але саме ці показники мають вирішальне значення при створенні надійних та довговічних технологій.

2.1 Сенсорні пристрої

Для початку розглянемо сенсорні, або інакше вхідні пристрої. Це можуть бути пристрої різних типів і складності: від простих термопар до відеосистем. Однією з причин вибухового зростання IoT є той факт, що ці сенсорні системи, завдяки здобуткам напівпровідникової промисловості та мікроелектроніки, отримали малі фізичні розміри та значно подешевшали.

2.2 Термопари та температурні датчики

Датчики температури – це найпоширеніший тип датчиків. Вони застосовуються повсюдно: від інтелектуальних термостатів для холодних складів до охолоджувачів промислового обладнання, і швидше за все це перший сенсор в IoT. Термопара – це пристрій для вимірювання температури, якому не потрібне джерело живлення, тому що він сам генерує сигнал малої амплітуди (зазвичай мікровольти). Термопара – це два провідники, виготовлені з двох різних матеріалів, з'єднані у точці вимірювання температури. На металевому електроді, залежно від температури, виникає електричний потенціал. У різних металів рівень цього потенціалу різний. Цей ефект відомий як електрорушійний ефект Зеєбека, його суть полягає в тому, що різниця потенціалів між двома різними металами перебуває в нелінійній залежності від їхньої температури. Величина напруги залежить від властивостей вибраного металу. Надзвичайно важливо, щоб кінці проводів були термічно ізольовані від системи (і проводи повинні мати однакову контрольовану температуру). На рис. 2.1 наведено блок-схему вимірювання температури за допомогою термопари. Для підвищення точності вимірювань різниця потенціалів, що індукується термопарою, зазвичай вимірюється методом, який називається методом компенсації. Оскільки різним температурам відповідають різні рівні напруги, а залежність між вимірюваною температурою і напругою, що індукується, нелінійна, для переведення виміряного потенціалу в температуру, як правило, використовується довідкова таблиця.

Термопари слід використовувати при виконанні не надто відповідальних вимірювань, оскільки показання окремих термопар, за інших рівних умов, можуть відрізнятися. Це викликано тим, що різні тонкі домішки, що входять до складу вимірювальних електродів можуть призводити до невідповідностей з довідковими таблицями. Можна, звичайно, скористатися високоточними (прецизійними) термопарами, але вони коштуватимуть дорожче. Іншим ефектом, що впливає на точність вимірів, є старіння.

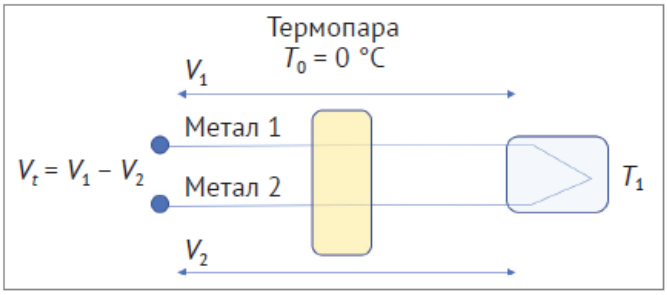


Рисунок 2.1 – Схема вимірювання температури за допомогою термопари

Так як термопари часто використовуються у промислових умовах, високотемпературні середовища з часом можуть погіршувати точність датчиків. Тому IoT-рішення повинні враховувати зміни, що відбуваються з датчиками у процесі їх експлуатації. Термопари добре працюють у широкому діапазоні температур. Для різних комбінацій металів прийняті кольорове та літерне маркування, що вказують тип термопари (наприклад, E, M, PT-PD). Зазвичай подібні датчики використовуються у промислових та високотемпературних середовищах під час проведення вимірювань у місцях, віддалених від оператора. На рис. 2.2 наведено залежність ЕРС від температури для кількох типів термопар.

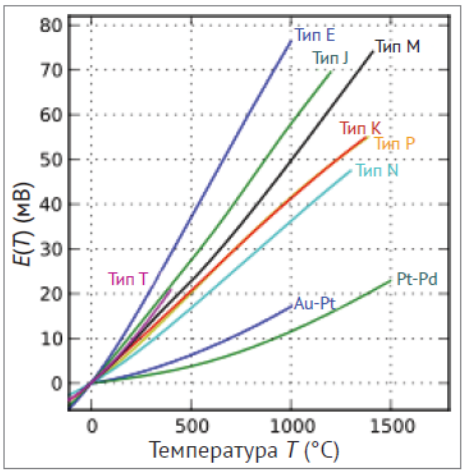


Рисунок 2.2 – Залежність ЕРС термопари від її температури E(T) / T

Резистивні датчики температури (Resistance Temperature Detectors – RTD) працюють у вузькому діапазоні температур (нижче 600°C), але дозволяють виконувати вимірювання з більшою точністю, ніж термопари. Зазвичай вони виготовляються з дуже тонкого платинового дроту, щільно намотаного на керамічний або скляний сердечник. Електричний опір такої конструкції пропорційний її температурі. Оскільки в основі вимірювань лежить вимірювання опору, для роботи з RTD необхідне зовнішнє джерело живлення з вихідною силою струму 1 мА. Наприклад, для датчика 200 PT100 RTD крок вимірювань становить 0,00200 Ом/°C, а діапазон вимірювань лежить у межах від 0 до 100°C. У межах цього діапазону залежність опору RTD від температури зберігає лінійний характер. Відповідно до стандартів RTD випускаються у дво-, три- та чотирипровідному виконанні, чотирипровідні моделі використовуються виключно в системах високоточного калібрування. Для збільшення роздільної здатності вимірювань RTD часто використовують у мостових схемах, при цьому зазвичай показання лінеаризуються програмно (рис. 2.3).

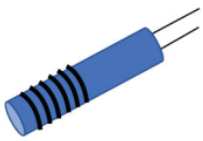


Рисунок 2.3 – Дротяний RTD

RTD рідко використовуються в діапазоні вище 600°C, що обмежує їхнє застосування в промисловості. При високих температурах платина може забруднюватися, що призводить до

помилкових показань, проте при вимірюваннях в межах заданого діапазону RTD демонструють досить точні та стабільні результати.

Термістор – це датчик температури, електричний опір якого залежить від його температури. Цей тип датчиків забезпечує більш високу точність вимірів у порівнянні з RTD. По суті це терморезистори, але з дуже нелінійною залежністю опору від температури. Їх часто використовують як згладжуючі фільтри, для обмеження стрибків струму, а також у випадках, коли необхідна висока ступінь дозволу вимірювань у вузькому діапазоні температур. Існує два типи термісторів: NTC (їх опір зменшується при підвищенні температури) та PTC (їх опір зростає із підвищенням температури). Основна відмінність від RTD полягає в тому, що термістори виготовляються з кераміки або полімерів, тоді як основою RTD є метал. Термістори знаходять застосування в медичному та науковому устаткуванні, харчовій промисловості, інкубаторах та таких побутових приладах, як термостати. У табл. 2.1 перераховано типи датчиків температури, наведено приклади їх використання, а також наведено переваги використання конкретних датчиків.

Таблиця 2.1 – Зведена таблиця датчиків температури

Категорія	Термопара	Резистивні датчики температури	Термістор
Температурний діапазон (°C)	Від -180 до 2,320	Від -200 до 500	Від -90 до 130
Час реакції	Швидко (мікросекунди)	Повільно (секунди)	Повільно (секунди)
Розміри	Великі (~10 мм)	Невеликі (~5 мм)	Невеликі (~5 мм)
Точність	Низька	Середня	Дуже висока

2.3 Ефект Холла та датчики струму

Датчик Холла – це смужка металу, якою пропущений електричний струм. Потік заряджених частинок, що проходять через магнітне поле, відхиляється від прямолінійного спрямування. Якщо напрямок магнітного поля перпендикулярно плоскому провіднику, то на його протилежних сторонах виникатиме різниця потенціалів, обумовлена тим, що різноіменно заряджені частинки збиратимуться на протилежних його сторонах. Таким чином, одна сторона плоского провідника виявиться заряджена позитивно, а інша негативно, і виникне різниця потенціалів. Така різниця потенціалів називається напругою Холла, а сам ефект виникнення цієї напруги називається ефектом Холла. Це показано на рис. 2.4. Коли через металеву смужку поміщену в магнітне поле, проходить струм, електрони притягуються до одного боку, а дірки – до іншого (див. криву на рис. 2.4).

Таке розшарування породжує електричне поле, яке можна виміряти. Якщо поле досить сильне, воно нейтралізує дію магнітного поля, і носії заряду зберігають прямолінійний рух: Ефект Холла застосовується в датчиках струму для вимірювання змінного і постійного струму. Існує два типи таких датчиків: із розімкненим та замкнутим контуром. Датчики із замкнутим контуром дорожчі, їх часто використовують у схемах із живленням від батарей.

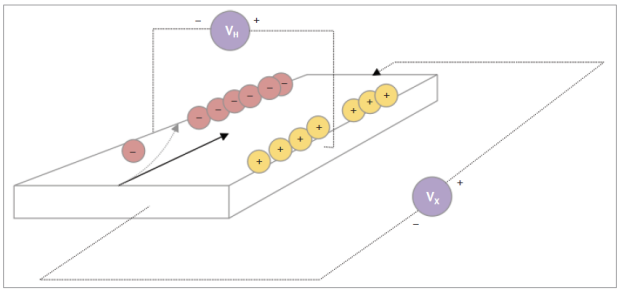


Рисунок 2.4 – Ілюстрація ефекту Холла

Типова сфера застосування датчиків Холла: датчики положення, магнітометри, високонадійні перемикачі та покажчики рівня води. Вони використовуються також у промислових датчиках для вимірювання швидкості обертання різних вузлів та механізмів. Крім того, що ці датчики недорогі у виготовленні, вони не вимогливі до умов експлуатації та можуть працювати у найсуворіших умовах.

2.4 Фотоелектричні датчики

Датчики виявлення світла чи визначення його інтенсивності використовуються у багатьох пристроях IoT. Такі пристрої необхідні, наприклад, у системах безпеки, інтелектуальних комутаторах або системах управління вуличним освітленням. Існує два типи таких датчиків, принцип дії яких зрозумілий з їхньої назви. Фоторезистор змінює опір залежно від інтенсивності світла, а фотодіод перетворює світло на електричний струм. Фоторезистори виготовляються із напівпровідників із високим опором. Їх опір зменшується зі збільшенням інтенсивності освітлення. У темряві опір фоторезистора може мати досить високий опір (порядку мегаом). Фотони, що поглинаються напівпровідником, переводять електрони в зону провідності, тим самим збільшуючи провідність матеріалу. Фоторезистори чутливі до довжини хвилі падаючого світла, тому їх типів та модифікацій існує безліч. А ось фотодіоди – це повноцінні напівпровідникові пристрої з PN-переходом. Такі пристрої реагують на світло, створюючи електронно-діркову пару. Потік дірок, що рухаються до анода, та електронів, що рухаються до катода, створює електричний струм. Таким чином, працюють традиційні сонячні батареї, що виробляють електрику під впливом сонячних променів. Якщо на фотодіод подати зворотну напругу, можна регулювати її чутливість або час відгуку (див. табл. 2.2).

Таблиця 2.2 – Фотоелектричні датчики

Категорія	Фоторезистор	Фотодіод
Світлочутливість	Низька	Висока
Активний/пасивний (напівпровідниковий)	Пасивний	Активний
Чутливість до температури	Висока чутливість	Низька
Час реакцію зміну освітленості	Тривале (від 10 мс до 1 с)	Короткий

Піроелектричний інфрачервоний (Pyroelectric Infrared – PIR) датчик складається з двох слотів, заповнених матеріалом, що реагує на інфрачервоне випромінювання та тепло. Типові варіанти застосування таких датчиків – це теплові датчики руху систем безпеки. Зазвичай такі датчики оснащуються лінзою Френеля, з якої формується зона виявлення. Така зона має форму арки, що розкривається назовні. Коли тепле тіло входить чи, навпаки, залишає зону виявлення, чутливі елементи формують електричний сигнал. У PIR-датчиках використовуються кристалічні матеріали, які здатні генерувати електричний струм під впливом ІЧ-випромінювання. Це утворює так званий польовий транзистор (Field Effect Transistor – FET), який фіксує зміну струму та посилає сигнал на підсилювальний пристрій. PIR-датчики добре працюють у діапазоні хвиль від 8 до 14 мкм, цей діапазон охоплює випромінювання людського тіла. Два елементи PIR, що формують дві зони виявлення (рис. 2.5). Це дозволяє не лише відстежувати весь простір кімнати, а й визначати напрямок переміщень.

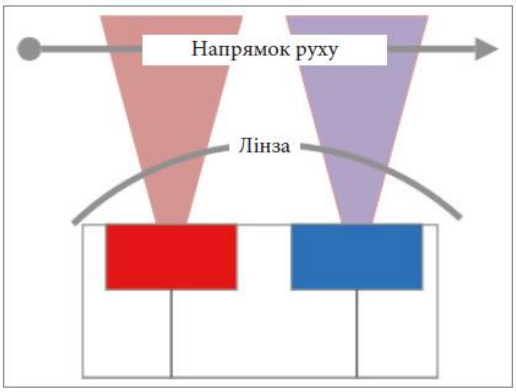


Рисунок 2.5 – Датчик PIR. Два елементи реагують на джерело ІЧ, що рухається в зоні виявлення

Для сканування більшої площі за допомогою одного датчика потрібно кілька лінз Френеля, які будуть фокусувати на PIR зображення окремих областей території, що відстежується. Таке фокусування забезпечує концентрацію інфрачервоного випромінювання безпосередньо в області FET. Як правило, такі пристрої дозволяють проектувальнику контролювати чутливість (діапазон) і

час реакції. Час реакції вказує, через який проміжок часу буде надіслано сигнал після виявлення руху на території, що охороняється. Чим менший час реакції, тим більше подій може бути зафіксовано. На рис. 2.6 наведено діаграму типового ПІР-датчика, оснащеного лінзою Френеля з фіксованою фокусною відстанню та сфокусованою на підкладку.

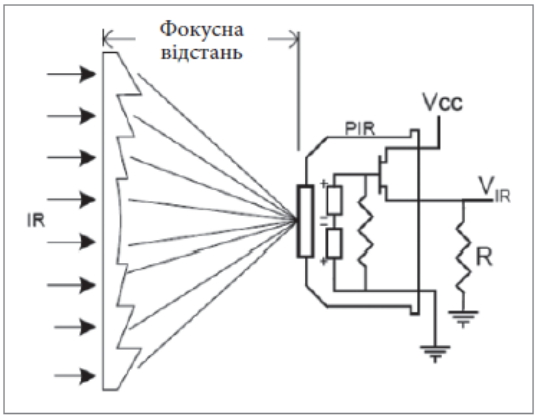


Рисунок 2.6 – Ліворуч: лінза Френеля фокусує ІЧ-випромінювання на PIR-датчик Cypress Microsystems AN2105

LiDAR та активні датчики. Активне зондування включає посилання сигналу і його аналіз після повернення. Такі датчики дозволяють судити про довкілля як якісно (зафіксувати факт руху), а й кількісно (виміряти швидкість руху). Ця область дуже велика, тому ми зосередимося на LiDAR-датчиках, які є основою активних зондувальних систем. Датчики світлового виявлення та вимірювання (Light Detecting and Ranging – LiDAR). Цей тип датчика вимірює відстань шляхом виміру відбитих від об'єкта лазерних імпульсів. Якщо датчик PIR лише виявить рух у межах свого діапазону, LiDAR здатний виміряти кількісні характеристики цього руху. Вперше такий датчик був продемонстрований у 1960-х рр., а в даний час широко використовується в сільському господарстві, автоматизованих і безпілотних транспортних засобах, робототехніці, при спостереженнях і дослідженнях навколишнього середовища. Цей тип активних вимірювальних пристроїв може аналізувати об'єкти будь-якого типу. Вони використовуються для аналізу газів, атмосфери, хмарних утворень і композицій, частинок, швидкості об'єктів, що рухаються LiDAR – активна сенсорна технологія, побудована на основі лазера. Коли лазерний промінь падає на об'єкт, якась його частина відображається і повертається до випромінювача LiDAR. Використовувані лазери зазвичай мають довжину хвилі від 600 до 1000 нм відносно недорогі. Їхня потужність обмежена з міркувань безпеки, щоб запобігти пошкодженню очей. Деякі датчики LiDAR працюють у діапазоні 1550 нм, оскільки ця довжина хвилі не сприймається людським оком, що робить їх нешкідливими навіть за високої інтенсивності. Системи LiDAR здатні сканувати дуже великі простори та можуть працювати навіть із супутників. Така система посилає лазерні імпульси з частотою до 150 000 імпульсів на секунду і фіксує їх відображення масивом фотодіодів. Іноді проходження посилених і відбитих лазерних імпульсів регулюється системою дзеркал, що обертаються, що формує тривимірне зображення навколишнього середовища. Для кожного переданого променевого імпульсу фіксується кут відображення, вимірюється час прольоту (Time of Flight – TOF) та місцезнаходження GPS. Ці параметри дозволяють отримати повне уявлення про досліджуваний простір. Рівняння для розрахунку відстані до об'єкта, що спостерігається, відносно просте:

$$\text{Відстань} = (\text{Швидкість прольоту} \times \text{Час прольоту}) / 2.$$

Інші активні датчики працюють за тим самим принципом, що й LiDAR. Кожен з них посилає який-небудь сигнал, який, відбиваючись, повертається до датчика, що і створює зображення сцени, що спостерігається, або вказує на те, що відбулася якась подія. Ці датчики набагато складніші, ніж прості пасивні датчики, споживають більше енергії, коштують дорожче та потребують більшого простору (рис. 2.7).

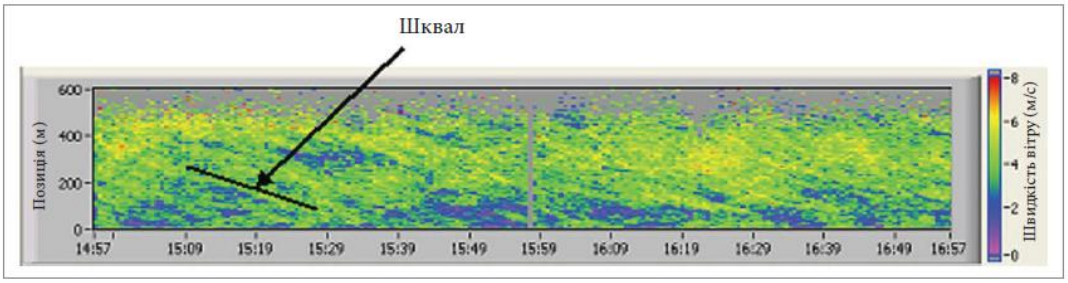


Рисунок 2.7 – LiDAR: приклад зображення, отриманого за допомогою LiDAR. Такі зображення використовуються для аналізу атмосферних поривів вітру, для забезпечення захисту вітрових турбін

Датчики MEMS. Промислове виробництво мікроелектромеханічних систем (Micro-electromechanical systems – MEMS) почалося в 1980-х рр., але вперше вони з'явилися ще в 1960-х рр., коли компанією Kulite Semiconductor був представлений п'єзорезисторний датчик тиску. По суті це мініатюрні механічні структури, які взаємодіють з електронним блоком управління. Як правило, розмір таких датчиків лежать в діапазоні від 1 до 100 мкм. На відміну від інших датчиків, згаданих у цьому розділі, механічні структури MEMS можуть обертатися, розтягуватися, згинатися, змінювати форму, що викликає зміни електричного сигналу. Цей сигнал, отриманий від окремого конкретного датчика, фіксується та вимірюється. Процес виготовлення MEMS-пристроїв типовий для виробництва напівпровідникових приладів. На кристал кремнію наноситься багатошарова маска, далі йдуть операції літографії, осадження та травлення. Потім матриця MEMS доповнюється іншими елементами, такими як операційні підсилювачі, аналого-цифрові перетворювачі та інші схеми допоміжні. Як правило, пристрої MEMS мають відносно великі розміри від 1 до 100 мікрон, тоді як типові кремнієві структури мають розміри 28 нм або менше. Тому виготовлення MEMS виконується пошарово, послідовне травлення різних шарів створює пристрій з тривимірною структурою). Можливість створення чутливих MEMS-пристроїв розміром із шпилькову головку дозволяє збільшити кількість об'єктів IoT до мільярдів. MEMS-акселерометри та гіроскопи. Акселерометри та гіроскопи широко застосовуються в різних мобільних пристроях для позиціонування та відстеження руху, в таких, наприклад, як крокоміри та фітнес-трекери. Ці пристрої використовують п'єзоелектричний елемент MEMS, на якому у відповідь на рух виникає ЕРС. Гіроскопи виявляють обертальний рух, а акселерометри реагують зміну лінійного руху. Діаграма на рис. 2.8 ілюструє принцип роботи акселерометра. Зазвичай масивне тіло, закріплене в певній точці, при виникненні прискорення через пружину створює механічну напругу MEMS, змінюючи його електричну ємність. Величина прискорення визначається при вимірюванні ємності ланцюга MEMS. Може здатися, що такий акселерометр реагуватиме на прискорення лише в одному напрямку. Але, як показано на рисунку нижче, такий акселерометр відповідатиме на прискорення у будь-якому з напрямків координатних осей (X, Y, Z).

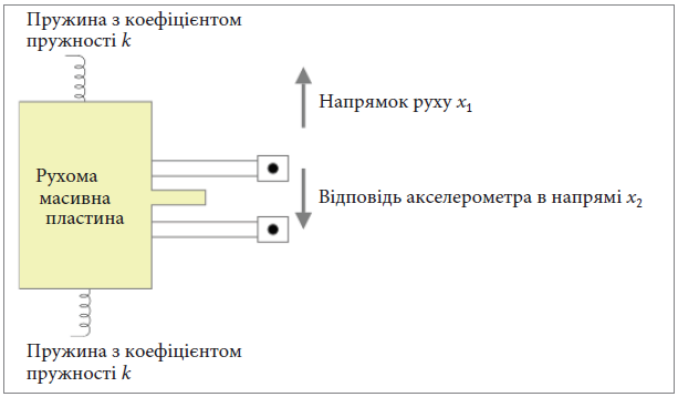


Рисунок 2.8 – Акселерометр: принцип вимірювання прискорення з використанням масивного тіла підвішеного на пружині. Як правило, використовується для вимірювання прискорення у напрямку координатних осей

Принцип дії гіроскопів заснований на ефекті Коріоліса для системи відліку, що обертається. На рисунку 2.9 представлена суть цього ефекту. Об'єкт, поміщений на диск, що обертається,

рухаючись рівномірно внаслідок обертання самого диска, починає рухатися по дузі. Таким чином, для збереження прямолінійного руху та досягнення північної точки диска об'єкту потрібне додаткове прискорення. Це прискорення Коріоліса. У пристрої MEMS немає диска, що обертається, зате є резонансна частота, прикладена до послідовності концентричних кілець, влаштованих на кремнієвій підкладці в процесі виготовлення MEMS.

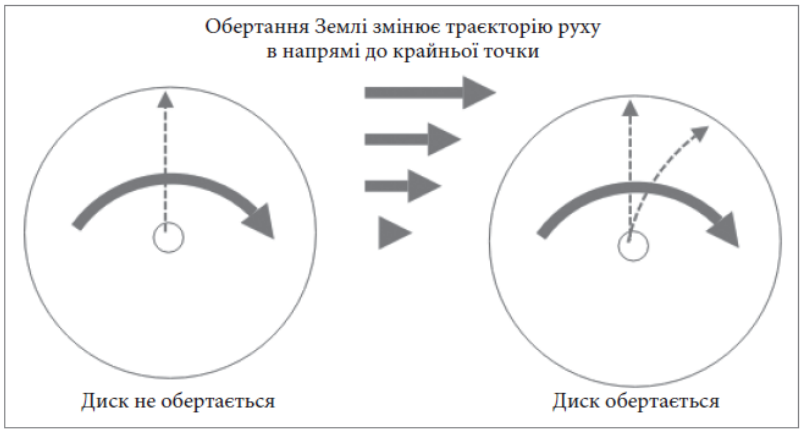


Рисунок 2.9 – Акселерометр: вплив обертання диска на шлях, що здійснюється під час руху на північ

Ці концентричні кільця розрізані на невеликі дуги. Концентричні кільця дозволяють точно оцінювати обертальний рух у широкому діапазоні. Поодинокі кільця необхідно влаштовувати на жорстких опорних балках. При розбивці кілець на дуги структура втрачає жорсткість і стає більш чутливою до обертання. Джерело постійного струму через електроди, прикріплені до кілець, створює електростатичне поле. Це поле викликає резонансні коливання системи кілець. Якщо фіксується зміна таких резонансних коливаннях кілець, значить, присутнє прискорення Коріоліса, яке визначається наступним рівнянням:

$$a = -2\omega \times v.$$

Це рівняння показує, що прискорення, викликане обертанням системи, пропорційно кутової швидкості диска, що обертається, як показано на рис. 2.9 або резонансної частоти MEMS-пристрою, як показано на рис. 2.10. Сила Коріоліса змінює проміжок між кільцями, встановлений статичним полем джерела постійного струму, і, як наслідок, загальну ємність системи. Зовнішні електроди фіксують відхилення в кільці, а внутрішні забезпечують вимірювання ємності.

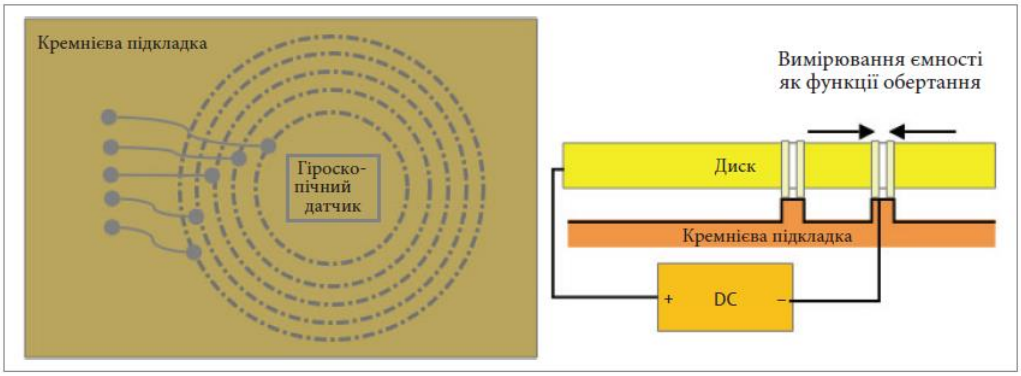


Рисунок 2.10 – Праворуч: концентричні кільця, що розрізають, влаштовані на кремнієвій підкладці, утворюють гіроскопічний датчик. Зліва: дискові проміжки з'єднані відповідним чином

Для роботи як гіроскопів, так і акселерометрів потрібне джерело живлення та операційний підсилювач для формування сигналу, придатного для обробки цифровим процесором. Подібні пристрої можуть бути виготовлені в дуже невеликих розмірах. Наприклад, пристрій InvenSense MPU-6050 містить 6-осьовий гіроскоп та акселерометр у корпусі розміром 4×4×1 мм. Він споживає струм 3,9 мА і, отже, є датчиком з низьким енергоспоживанням. Мікрофони MEMS. MEMS-пристрої можуть також використовуватися для фіксації звуку та вібрації. Цей тип MEMS-пристроїв має

безпосереднє відношення до розглянутих раніше акселерометрів. При розгортанні систем IoT фіксація звуку та вібрації широко застосовується в їх промислових додатках для профілактичного та прогностичного обслуговування. Наприклад, в хімічній промисловості апарат, який обертає реактиви, що завантажуються для їх перемішування, або центрифуги повинні бути строго горизонтально орієнтовані. А звуковий чи вібраційний MEMS-блок зазвичай використовується для контролю за справністю та безпекою такого обладнання. Цьому типу датчиків потрібен аналого-цифровий перетворювач із досить високою частотою дискретизації. Необхідний підсилювач вихідного сигналу. Імпеданс MEMS-мікрофона становить декілька сотень. Мікрофон MEMS може бути аналоговим чи цифровим. Аналоговий мікрофон підключається до джерела постійного струму та аналого-цифрового перетворювача. Цифровий мікрофон має вбудований АЦП у безпосередній близькості від детектора звуку. Це є перевагою при перешкодах від стільникових телефонів або Wi-Fi-пристроїв, які можуть вплинути на роботу АЦП.

Для передачі вихідного сигналу цифрового MEMS-мікрофона використовують два типи модуляції: модуляція щільності імпульсів (pulse density modulated – PDM) та модуляція I2S. PDM – це протокол з високою частотою дискретизації та двома каналами прийому-передачі (лівий та правий канали). Прийом-передача сигналу та тактової частоти відбувається за окремими лініями, а прийом-передача сигналу в лівий та правий канали розділена у часі (чергується) відповідно до тактового сигналу. Протокол I2S не має високої частоти дискретизації, але під час передачі сигналу звуковий частоти (в діапазоні від Hz до kHz) працює з прийнятним рівнем якості. Він також дозволяє передавати-приймати сигнал двома каналами, зате може обходитися зовсім без АЦП, оскільки децимація (вибірка) може відбуватися у мікрофоні. Для демодуляції PDM сигналу необхідний процесор цифрового сигналу (digital signal processor – DSP). Датчики тиску MEMS. Тензодатчики тиску знаходять застосування у широкому діапазоні IoT: від контролю над інфраструктурою розумних міст до промислового виробництва. Зазвичай вони використовуються для вимірювання тиску рідин та газів. Основою датчика є п’єзоелектричний елемент, якого організований фізичний доступ через діафрагму (отвір) в підкладці над або під елементом. Підкладка робиться гнучкою, що дозволяє п’єзоелементу під впливом тиску згинатися, змінюючи свій електричний опір (рис. 2.11).

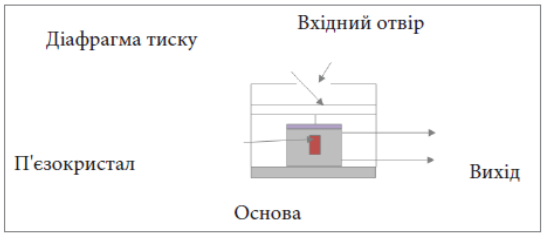


Рисунок 2.11 – Датчик тиску

Принцип дії цього типу датчиків аналогічний іншим, перерахованим у цьому розділі. Струм від зовнішнього джерела живлення, що пройшов через датчик, вимірюється за допомогою Уітстонського моста. Такі мости можуть бути виконані у двох-, чотирьох-або шестипровідних модифікаціях. Міст вимірює зміни потенціалу ланцюга, коли п’єзоелектрична підкладка згинається і змінює свій опір (рис. 2.12).

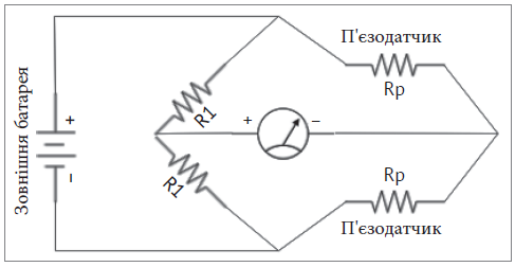


Рисунок 2.12 – Міст Уітстона, що використовується для вимірювання опору датчика тиску MEMS

Література: [5, 8, 9, 11].

ТЕМА 3
ІНТЕЛЕКТУАЛЬНІ КІНЦЕВІ ТОЧКИ IOT

Ми розглянули прості датчики, які перетворювали інформацію у двійковій чи аналоговій формі, яку ще треба обробляти. Однак існують пристрої та датчики IoT, які мають значну обчислювальну потужність і продуктивність, достатню для самостійної обробки даних і прийняття рішень. Інтелектуальні датчики мають у своєму складі такі пристрої, як відеокамери та навіть цілі системи відеоспостереження. Такі датчики можуть проводити значні обсяги обчислень завдяки вбудованим процесорам цифрового сигналу: ПЛІС (програмована логічна інтегральна схема) та ASIC (application-specific integrated circuit – інтегральна схема спеціального призначення).

3.1 Відеосистема

На відміну від простих датчиків, розглянутих раніше, відеосистеми набагато складніші, оскільки вимагають серйозного апаратного забезпечення, оптики та світлочутливих матриць зображення. Такі системи починаються з об'єктива, з якого здійснюється спостереження. Об'єктив забезпечує як різкість зображення, а й велику світлочутливість активного елемента. У сучасних системах бачення використовується один із двох типів чутливих елементів: прилади із зарядовим зв'язком (ПЗЗ) або комплементарні металооксидні напівпровідники (КМОП). Різницю між КМОП та ПЗЗ можна виразити як:

– ПЗЗ (CCD): сигнал від датчика до периферійного обладнання мікросхеми передається за допомогою аналого-цифрового перетворювача. Ці датчики створюють зображення з високою роздільною здатністю та малим шумом. Зате вони споживають значну потужність (100× від КМОП) та складні у виготовленні;

– КМОП (CMOS): зображення будується з окремих пікселів (точок), кожен піксель формується окремим транзистором, тобто кожен піксель зчитується окремо. КМОП сприйнятливіший до шуму, але дуже енергетично економічний.

Більшість відеодатчиків, представлених на сучасному ринку, побудовано за КМОП-технологією. Такий датчик вбудований в кремнієву підкладку і виглядає як двовимірна матриця транзисторів, розташованих рядами та стовпцями. Кожен осередок такої матриці складається з трьох фотодіодів для трьох кольорів – червоного, зеленого та синього. Кожен фотодіод має мікролінзу, яка фокусує випадкові промені певного кольору, послаблюючи інші. Ці лінзи далеко не ідеальні, у них виникає хроматична аберація, тобто різні довжини хвиль переломлюються з різною швидкістю, що призводить до розмиття зображення. Лінзи можуть також викликати спотворення зображення внаслідок подушкоподібних спотворень. Далі ми розглянемо дії, що вживаються для фільтрації шумів, нормалізації та оцифрування зображення, щоб надати йому зручну для використання форму. Основа процесу – це процесор відеосигналів зображення (image signal processor – ISP). Вся процедура може виконуватись у такому порядку (рис. 3.1).

Зверніть увагу на численні перетворення на кожному етапі формування зображення: ці перетворення виконуються для кожного пікселя. Для обробки такого обсягу даних потрібні значні обчислювальні потужності. Нижче наведено функціональні блоки, необхідні для обробки зображення:

– аналого-цифрове перетворення: посилення сигналу датчика з подальшим його перетворенням цифрову форму (10 біт). Дані, що представляють захоплене зображення, зчитуються з матриці фотодіодів як з двомірної таблиці, що складається з рядків і стовпців;

– оптичний затискач: видаляє ефекти затемнення, що виникають внаслідок засвічення окремих пікселів датчика (sensor black level);

– баланс білого: імітує хроматичне сприйняття людським оком різних колірних температур, внаслідок чого нейтральні тони виглядають нейтральними. Виконується з використанням перетворення матриць;

– корекція мертвого пікселя: визначає пікселі, що вийшли з ладу, та компенсує їхню втрату з використанням інтерполяції, значення мертвих пікселів замінюються усередненими значеннями сусідніх;

– розфарбовування (Debayer filtering) та збирання мозаїчного зображення (demosaicing): надає монохромному зображенню кольору, причому насиченість зеленого кольору визначається як функція від рівнів червоного та синього кольорів. Створює плоске зображення за допомогою черезрядкової розгортки. Для отримання різкіших зображень з чітко окресленими межами і контурами використовуються складніші алгоритми;

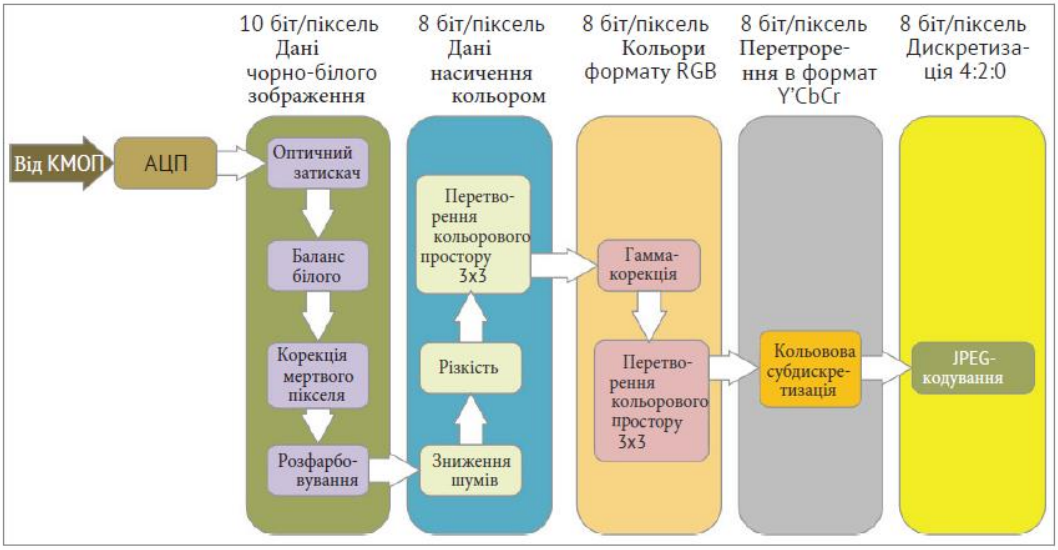


Рисунок 3.1 – Датчик зображення: типовий процесор відеосигналу зображення для створення кольорового відео

– зниження шуму: всі датчики створюють шум. Шум може бути пов'язаний з нерівномірністю чутливості пікселів на рівні транзистора або витік фотодіода. Це утворює темні області. Існують інші форми шуму. На даному етапі білий і когерентний шуми видаляються шляхом введення проміжного фільтра (масив 3×3) для всіх пікселів. В якості альтернативи іноді використовують фільтр плям (despeckle filter), що вимагає сортування пікселів, але існують інші методи. Всі перелічені методи обробляють всі пікселі по всій матриці загалом;

– різкість: проводиться розмиття зображення з використанням матричного множення, а потім у поєднанні з деталізацією по окремих областям створюється ефект наведення різкості;

– перетворення колірного простору 3×3: перетворення колірного простору на дані, що відповідають стандарту формату RGB;

– гамма-корекція: виправляє нелінійний відгук датчика зображення КМОП на дані RGB для різного освітлення. Гамма-корекція використовує довідкову таблицю (lookup table – LUT) для інтерполяції та виправлення зображення;

– перетворення колірного простору 3×3: додаткове перетворення колірного простору із формату RGB у формат Y'CbCr. YCC був обраний, оскільки Y можна зберігати з більш високою роздільною здатністю, ніж CbCr, без втрати візуальної якості. Структура дискретизації 4:2:2;

– колірна субдискретизація (Chroma subsampling): усуває нелінійність RGB-тонів, коригує зображення для імітації зображення на інших носіях, таких як плівка. Таким чином покращується відповідність тонів та якості;

– JPEG-кодування: стандартний алгоритм стиснення JPEG.

Тут слід підкреслити, що це хороший приклад того, наскільки складним може бути датчик, який обсяг даних, обладнання та складності можна співвіднести із простою відеосистемою. Обсяг даних, що обробляються відеосистемою з частотою 60 кадрів в секунду при роздільній здатності 1080p, просто величезний. Припускаючи, що це етапи обробки (крім стискування JPEG) виконуються у вигляді ISP, оскільки це одна мікросхема (ASIC) і всі етапи виконуються за цикл, то загальний обсяг оброблюваних даних становитиме 1368 ГБ/с. А з урахуванням стиснення JPEG, яке виконується користувачам CPU/DSP (центральний процесор/співпроцесор), обсяг даних складе вже більше ніж 2 ГБ/с. Ніхто ніколи не передає необроблений відеопотік (raw Bayer) у хмару для обробки, оскільки ця робота повинна виконуватися якомога ближче до відеодатчика.

3.2 Злиття датчиків

До всіх сенсорних пристроїв, описаних у цьому розділі, застосовується концепція злиття датчиків. Це процес об'єднання кількох різних датчиків з метою отримання більшого обсягу інформації, ніж може забезпечити один датчик. У просторі IoT це важливо, оскільки, наприклад, одиничний тепловий датчик не має уявлення про те, що саме викликає швидку зміну температури. Але в поєднанні з іншими датчиками, наприклад, датчиками PIR, що фіксують рух та інтенсивність освітленості, система IoT може зрозуміти, що у певній області зібралася велика кількість людей і яскраво світить сонце, на цій підставі вона може ухвалити рішення про посилення циркуляції повітря А один термодатчик просто зафіксує поточне значення температури без усвідомлення того, що температура зростає через те, що зібралися люди і світить сонце.

На основі більшої кількості даних від більшої кількості датчиків, відповідно корелюваних у часі, система може приймати більш виважені рішення. Це одна з причин того, що кількість датчиків, поміщених у хмару IoT, зростає, викликаючи зростання обсягів даних. Датчики стають дешевшими, легше інтегруються, і на прикладі TISensorTag легко бачити, як комбінація датчиків полегшує загальне бачення.

Існує два режими злиття датчиків:

– централізований: дані передаються до центрального офісу, де і відбувається їхнє злиття (приклад – хмарні технології);

– децентралізований: кореляція даних безпосередньо в датчику (або поруч із ним).

В основі кореляції даних датчика лежить центральна гранична теорема, на основі якої два незалежні виміри x_1 і x_2 об'єднуються з урахуванням їх дисперсій (відхилень від норми), щоб набути третього значення x_3. Тобто це просто розрахунок середньозваженого значення перших двох величин:

$$x_3 = (\sigma_1^{-2} + \sigma_2^{-2})^{-1} (\sigma_1^{-2} x_1 + \sigma_2^{-2} x_2).$$

Іншими методами злиття датчиків є фільтри Калмана та Байєсівські мережі.

3.3 Пристрої введення

Існує ще безліч типів датчиків, розгляд яких ми не торкаємося в цьому розділі. Це різні газові аналізатори, датчики вологості, датчики радіоактивного випромінювання, датчики диму, ультразвукові датчики тощо. Тим не менш, цей розділ повинен дати ґрунтовні уявлення про сенсорні пристрої та забезпечити його знаннями, необхідними при їх виборі. До цього ми розглядали кінцеві точки, представлені датчиками. Ці пристрої надсилають постійний потік даних на центральний пристрій або у хмару. IoT будується з урахуванням двонаправлених систем. Вхідні дані можуть передаватися в кінцеву точку з хмари, або, навпаки, дані можуть бути надіслані кінцевою точкою іншим абонентам хмари.

3.4 Пристрої виводу

Вихідні пристрої в екосистемі IoT можуть бути практично будь-якими: від простий світлодіод до повноцінної відеосистеми. До інших типів вихідних сигналів відносяться виконавчі механізми, крокові двигуни, гучномовці та аудіосистеми, промислові клапани. Зрозуміло, ці пристрої потребують різних систем управління різною складності. Залежно від типу виходу та використовуваного варіанта використання, також слід очікувати, що більша частина контролю та управління повинна проводитися безпосередньо поблизу пристрою (на противагу повному контролю у хмарі). Наприклад, відеосистема може передавати дані хмарних провайдерів, але для цього потрібне обладнання виводу та буферизації. У загальному випадку системам виведення потрібні значні обсяги електроенергії для її перетворення на механічний рух, теплову енергію чи світ. Наприклад, невеликий соленоїд для керування потоком рідини або газу при напрузі живлення 9…24 В буде споживати струм приблизно в 100 мА, при цьому створить спрямування зусилля всього в п'ять ньютонів. А промислові соленоїди працюють від напруги у сотні вольт.

3.5 Функціональні приклади

Набір датчиків досить непотрібний, якщо дані, які вони збирають, не можуть передаватися та оброблятися. Незалежно від того, чи встановлений локальний або вбудований контролер або дані відправляються на більш високий рівень, для побудови повноцінної системи потрібен більший набір обладнання, ніж датчики і котролер. Найбільш поширені інтерфейси введення-виводу, які використовуються в датчиках, це I2C, SPI, UART, хоча існують інші. Для таких пристроїв, як відеосистеми з високою роздільною здатністю та великою кадровою частотою, необхідні високошвидкісні IO-інтерфейси, такі як MIPI, USB або навіть PCIExpress. Датчики можуть також використовуватися з бездротовим обладнанням зв'язки, такі як Bluetooth, Zigbee або 802.11. Все це вимагає додаткових компонентів, які ми розглянемо в цьому розділі.

Texas Instruments CC2650 SensorTag є гарним прикладом сенсорного модуля IoT для розробки та проектування. Нижче наведено опис його функціоналу та вбудованих датчиків:

1. Вхідний датчик:
– датчик освітленості (TI Light Sensor OPT3001);
– інфрачервоний датчик температури (TI Thermopile infrared TMP007);
– датчик температури навколишнього середовища (TI light sensor OPT3001);
– акселерометр (Invensense MPU-9250);
– гіроскоп (Invensense MPU-9250);
– магнітометр (Bosch Sensortec BMP280);
– альтиметр (датчик висоти) / Датчик тиску (Bosch Sensortec BMP280);
– датчик вологості (TI HDC1000);
– мікрофон MEMS (Knowles SPH0641LU4H);
– магнітний датчик (Bosch SensorTec BMP280);
– 2 інтерфейси GPIO;
– герконове реле (Meder MK24).
2. Пристрої виведення:
– зумер / динамік;
– 2 світлодіоди.
3. Комунікації:
– bluetooth Low Energy (Bluetooth Smart);
– мережевий протокол – Zigbee;
– протокол взаємодії з бездротовими мережами – 6LoWPAN.

Весь комплект розрахований на живлення від однієї батареї-монети CR2032. Пристрій може працювати в режимі маяка (iBeacon) або використовуватися як оповіщувач. На рис. 3.2 наведено блок-схему модуля CC2650 SensorTag.

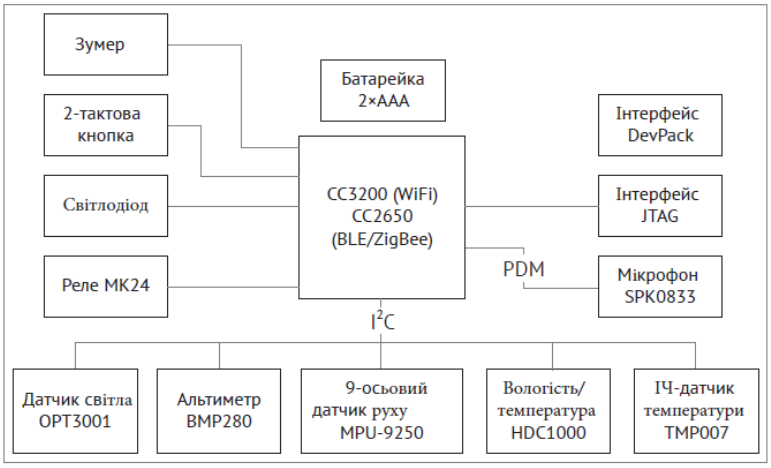


Рисунок 3.2 – TI CC2650 SensorTag. Виробник Texas Instruments, TI Multi-Standard CC2650 SensorTag Design Guide

На рисунку 3.3 наведено блок-схему пристрою для реалізації аудіо- та відеоконференцій (Multipoint Control Unit – MCU). MCU забезпечує можливість введення-виведення, сигнал

обробляється з процесором ARM Cortex M4, для підключення датчиків має різні інтерфейси. Пристрій оснащений кількома датчиками, системами зв'язку, інтерфейсами, але обчислювальна потужність невелика. У ньому використовується модуль обробки TI (MCU CC265), який включає невеликий процесор ARM Cortex M3 з флеш-пам'яттю об'ємом 128 КБ і 20 КБ SRAM. Має надзвичайно низьке енергоспоживання. Але, незважаючи на високу енергоефективність, обсяг оброблюваної інформації невеликий через обмеженість ресурсів. Як правило, такі пристрої супроводжуються шлюзами, маршрутизаторами, смартфонами чи іншими інтелектуальними пристроями. Сенсорні пристрої бюджетного виконання орієнтовані на низьке енергоспоживання та малу вартість не мають ресурсів для підтримки більш вимогливих додатків, таких як стеки протоколів MQTT, агрегація даних, стільниковий зв'язок або аналітика. Подібні пристрої призначені для стеження за польовими датчиками, вони розроблялися виключно з метою економії коштів та зниження витрат.

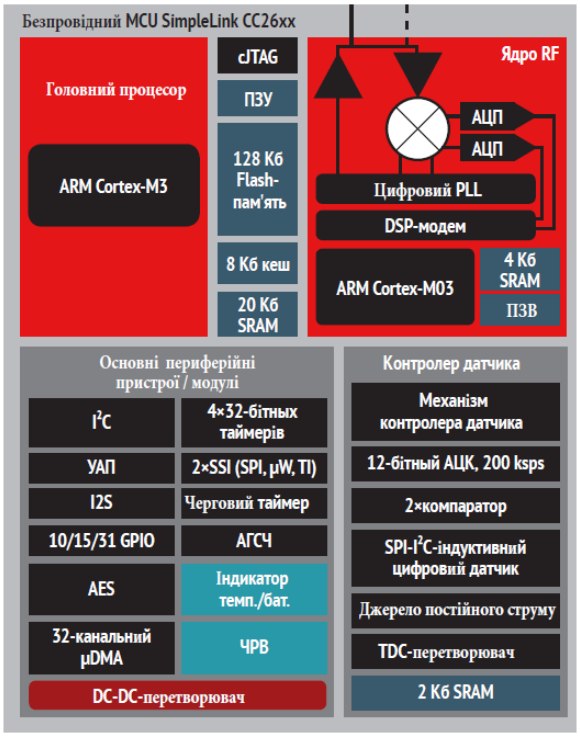


Рисунок 3.3 – TI CC2650. Блок-схема MCU. Надано Texas Instruments, TI Multi-Standard CC2650 SensorTag Design Guide

У багатьох попередніх прикладах сигнал датчика перед відправкою на наступний рівень вимагає посилення, фільтрації та калібрування. Крім цього, як правило, потрібний аналого-цифровий перетворювач. На рисунку 3.4 наведено простий 24-розрядний АЦП з опорною напругою 5 В.

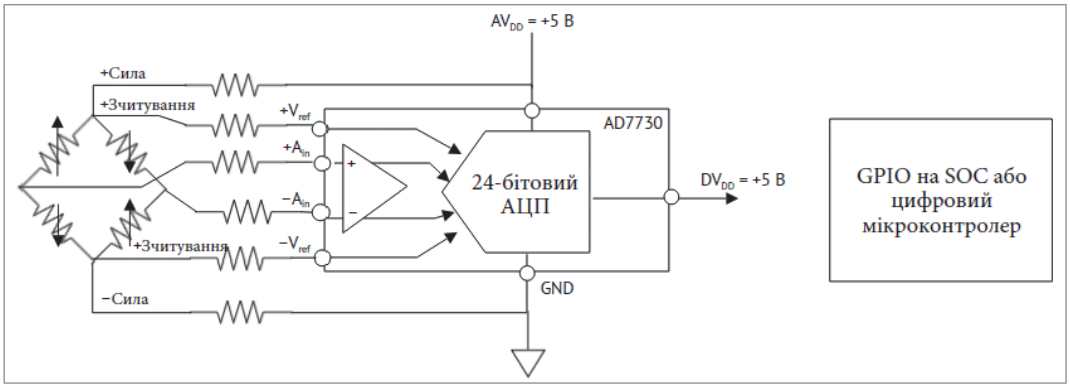


Рисунок 3.4 – Міст Уітстона: підключений до аналого-цифрового перетворювача AD7730, який виступає як вхідний пристрій для підключення до мікроконтролера або іншого вхідного перетворювача

Далі з виходу АЦП сигнал може надходити на вхід пристрою, яке перетворює його в імпульсно-модульований сигнал або команди якогось послідовного інтерфейсу, наприклад, I2C, SPI або UART, які вже будуть передані безпосередньо мікроконтролеру або процесору цифрових сигналів. Він призначений для точного визначення температури навколишнього середовища від -40 до +125°C. Усі компоненти, описані у цьому розділі, представлені на рисунку 3.5.

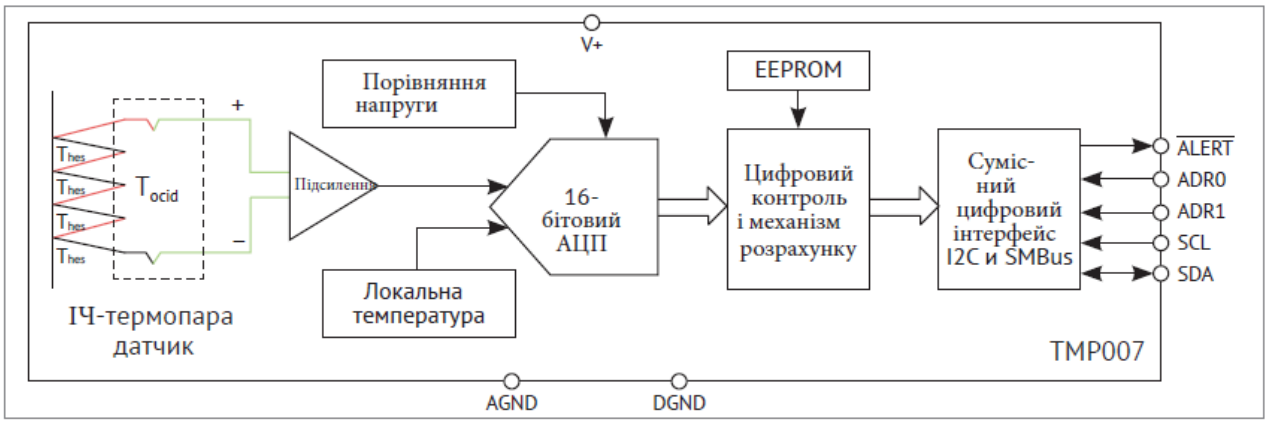


Рисунок 3.5 –TI Multi-Standard CC2650

Хорошим прикладом може бути інфрачервоний термочутливий датчик Texas Instruments (TMP007). Це безконтактний MEMS-датчик температури, який поглинає інфрачервоне випромінювання та перетворює його на напругу.

Література: [4, 5, 6, 8, 9, 11].

ТЕМА 4
ТЕОРІЯ КОМУНІКАЦІЇ ТА ІНФОРМАЦІЇ

Інтернет Речей – це більше, ніж дані від датчика. Ми повинні спочатку зрозуміти і зробити модель перенесення даних рухомих датчиків із найвіддаленіших місць на Землі в хмару. Існує значна кількість технологій та шляхів передачі для переміщення даних, і основна частина матеріалу висвітлюватиме аспекти, обмеження та порівняння варіантів комунікації для архітектора. Ми починаємо обговорення WAN з огляду бездротових радіосигналів та факторів, що впливають на якість, обмеження, перешкоди, моделі, пропускну здатність та діапазон. У різних діапазонах є багато протоколів обміну WAN, і архітектор має розуміти плюси та мінуси вибору одного радіочастотного спектра порівняно з іншим.

Рисунок 4.1 допомагає визначити різні діапазони та швидкості передачі даних для бездротових протоколів, які ми розглянемо. WPAN часто використовується з іншими акронімами для ближнього зв'язку, такими як Field Area Network (FAN), Wireless Local Area Network (WLAN), бездротова Home Area Network (HAN), Wireless Neighborhood Area Network (NAN) та Wireless Body Area Network (WBAN).

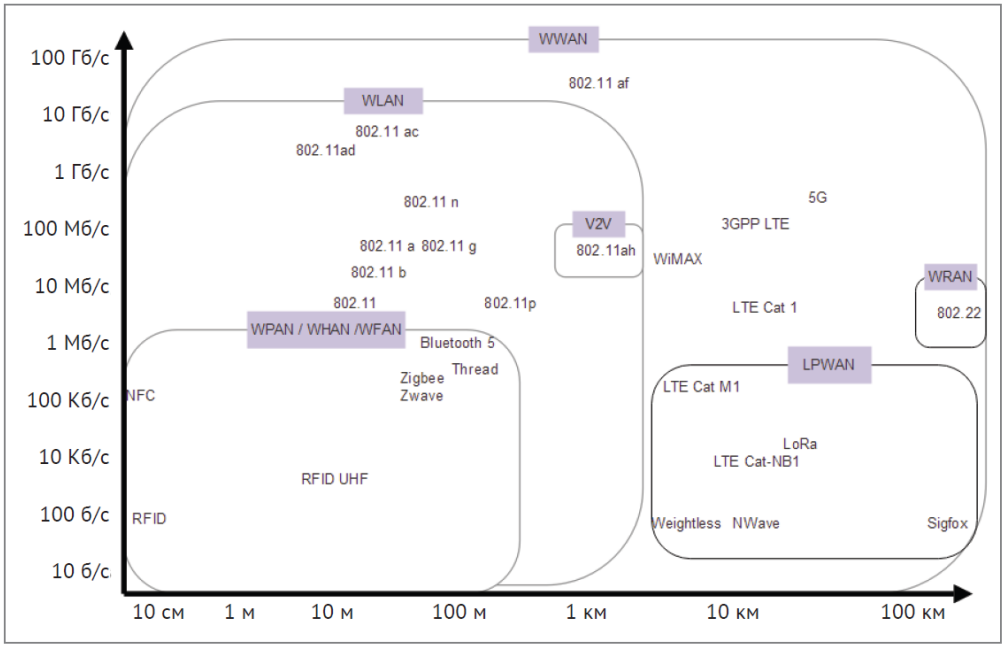


Рисунок 4.1 – Різні протоколи та категорії бездротового зв'язку, призначені для різних діапазонів, швидкості передачі даних та варіантів використання (потужність, транспорт тощо)

У цьому розділі будуть представлені основні моделі та теорія систем зв'язку, частотних просторів та теорії інформації. Комунікаційні обмеження та моделі порівняння будуть надані, щоб зрозуміти, як і чому працюють певні типи передачі даних та де вони не працюватимуть. Отже, ми починаємо з теорії комунікації, оскільки вона відіграє фундаментальну роль у виборі правильного поєднання бездротових технологій для розгортання рішення IoT.

4.1 Теорія комунікації

Інтернет Речей є конгломератом багатьох окремих пристроїв, які автономно виробляють та / або споживають дані на дуже далекому краю шарів мереж та протоколів. Важливо розуміти обмеження у побудові систем зв'язку для IoT або у будь-якій формі мереж. Інтернет Речей об'єднує персональні мережі, локальні мережі та дальні глобальні мережі у мережу каналів зв'язку. Більшість того, що робить IoT можливим, будується навколо комунікаційної основи; тому ця глава присвячена розгляду основ мережевих та комунікаційних систем. Зосередимося на системах зв'язку та сигналізації. Розглянемо діапазон, енергію та обмеження систем зв'язку та те, як архітектор використовуватиме ці інструменти для розробки успішного рішення IoT.

4.2 Радіочастотна енергія та теоретичний діапазон

Важливо розглянути діапазон передачі, коли йдеться про бездротові персональні мережі або будь-який радіочастотний протокол бездротового зв'язку. Конкуруючі протоколи використовують діапазон, швидкість та потужність як роздільники. Як архітектори ми повинні враховувати різні протоколи та варіанти дизайну при реалізації повного рішення. Діапазон передачі заснований на відстані між передавачем та приймальними антенами, частотою передачі та потужністю передачі. Найкраща форма радіопередачі – це безперешкодна пряма видима ділянка без додаткових радіосигналів. У більшості ситуацій ця ідеальна модель не існуватиме. У реальному світі є перешкоди, відображення сигналів, кілька бездротових радіочастотних сигналів та шум. При розгляді конкретної глобальної мережі та сигналу меншої швидкості, наприклад, 900 МГц порівняно з сигналом 2,4 ГГц, ви можете отримати послаблення функції довжини хвилі для кожної частоти. Це дасть зміни щодо потужності сигналу у будь-якому діапазоні. Загальна форма рівняння передачі Фріїса (рис. 4.2) має вигляд:

$$P_r = P_t G_{Tx} G_{Rx} \frac{\lambda^2}{(4\pi R)^2}.$$

Децибельна (дБ) форма рівняння Фріїса визначається як:

$$P_r = P_t + G_{Tx} + G_{Rx} + 20\log_{10}\frac{\lambda}{(4\pi R)^2},$$

де G_{Tx} і G_{Rx} – коефіцієнт посилення передавача та приймача, R – відстань між передавачем та приймачем, а P_r та P_t – потужність приймача та передавача відповідно.

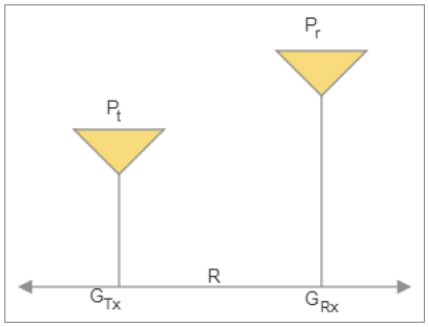


Рисунок 4.2 – Графічне подання рівняння Фріїса

Сигнал 900 МГц на 10 м матиме втрати 51,5 дБ, а сигнал 2,4 ГГц на 10 м матиме втрату 60,0 дБ. Ми можемо довести, як потужність та діапазон впливають на якість сигналу, використовуючи відношення, що має назву бюджетом посилання. Це порівняння потужності передачі з рівнем чутливості і воно вимірюється за логарифмічною шкалою (дБ). Можна просто хотіти підвищити рівень потужності задоволення вимог до діапазону, але часто це порушує нормативні вимоги чи впливає термін служби батареї. Інший варіант полягає у покращенні рівня чутливості приймача, який точно відповідає Bluetooth 5 останньої специфікації. Бюджет посилання визначається співвідношенням потужності передавача та чутливості приймача, як показано нижче.

БюджетПосилання = Потужність передачі T_x / Рівень чутливості S_x

Бюджет зв'язків вимірюється за логарифмічною шкалою у дБ; тому додавання децибелів еквівалентне множенню числових коефіцієнтів, які дають нам рівняння:

Потужність приймача (дБ) = Сила передачі (дБ) + Посилення (дБ) – Втрати (дБ).

Припускаючи, що немає ніякого фактора, що сприяє посиленню сигналу (наприклад, посилення антени), є лише два способи покращення прийому: збільшення потужності передачі або зменшення втрат. Архітектор повинен моделювати максимальний діапазон конкретного протоколу, він використовуватиме Free-Space Path Loss (FSPL). Це величина втрати сигналу електромагнітної хвилі прямої видимості у вільному просторі (без перешкод). Другим фактором FSPL є частота (f) сигналу, відстань (R) між передавачем і приймачем та швидкість світла (c). У термінах обчислення FSPLF у децибелах рівняння буде мати вигляд:

$$\begin{aligned} FSPL\,(\partial Б) = \quad & 10\log_{10}\left(\left(\frac{4\pi Rf}{c}\right)^2\right) \\ & 20\log_{10}\left(\frac{4\pi Rf}{c}\right) \\ & 20\log_{10}(R)+20\log_{10}(f)+20\log_{10}\left(\frac{4\pi}{c}\right) \\ & 20\log_{10}(R)+20\log_{10}(f)-147.55 \end{aligned}$$

Формула FSPL є простим розрахунком першого порядку. Найкраще наближення враховує відображення та хвильові перешкоди від земної поверхні, такі як формула втрат на плоскій землі. Тут h_t – висота передавальної антени, h_r – висота приймальної антени, k це число хвиль у вільному просторі і спрощується, як показано. Перетворимо рівняння для використання дБ-позначення:

$$\frac{P_r}{P_t} = L_{\text{втратна плоскій землі}} \approx \left(\frac{\lambda}{4\pi R}k\frac{2h_t h_r}{R}\right) \approx \frac{h_t^2 h_r^2}{R^4},\ k=\frac{2\pi}{\lambda}.$$

Те, що відомо як втрати на плоскій землі, полягає у тому, що відстань впливає втрати на 40 дБ за декаду. Збільшення висоти антени допомагає. Типи перешкод, які можуть статися природним чином, включають:

– відображення: коли електромагнітна хвиля, що поширюється, натикається на об’єкт і призводить до множинних хвиль;

– дифракція: коли радіохвильовий шлях між передавачем та приймачем утруднений об’єктами з гострими краями;

– розсіювання: коли середовище, через яке проходить хвиля, складається з об’єктів, розмір яких менше довжини хвилі і кількість таких перешкод велике.

Це важлива концепція, оскільки архітектор повинен вибрати рішення WAN, частота якого врівноважує пропускну здатність даних, максимальний діапазон сигналу та здатність сигналу проникати в об’єкти. Збільшення частоти, природно, збільшує втрати у вільному просторі (наприклад сигнал 2,4 ГГц має покриття на 8,5 дБ менше, ніж сигнал 900 МГц). Взагалі, сигнали 900 МГц будуть надійними на подвоєній дистанції сигналів 2,4 ГГц. Сигнали 900 МГц мають довжину хвилі 333 мм проти 125 мм сигналу 2,4 ГГц. Це дозволяє сигналу з частотою 900 МГц мати кращу проникаючу здатність і не так сильно залежати від розсіювання. Розсіювання є важливою проблемою для систем глобальної мережі, оскільки багато розгортань не мають прямої видимості між антенами – натомість сигнал повинен проникати крізь стіни та підлоги. Ми бачимо, що 900 МГц має перевагу перед 2,4 ГГц при проникненні через матеріал (див. табл. 4.1 та рис. 4.3).

Таблиця 4.1 – Матеріали та втрати радіосигналу

Матеріал	Втрати у дБ на 900 МГц	Втрати у дБ на 2,4 ГГц
Скло 6 мм	-0,8 дБ	-3 дБ
Кладка із цегли або цегляних блоків (20 см)	-13 дБ	-15 дБ
Гіпсокартон	-2 дБ	-3 дБ
Двері з цільного дерева	-2 дБ	-3 дБ

Як ми побачимо, багато протоколів є комерційно доступними і використовуються в усьому світі в спектрі 2,4 ГГц. 2,4 ГГц забезпечує пропускну здатність у п’ять разів більше у порівнянні з

сигналом 900 МГц і може мати набагато меншу антену. Крім того, спектр 2,4 ГГц неліцензований та доступний для використання у багатьох країнах. Порівняння частот показано у табл. 4.2.

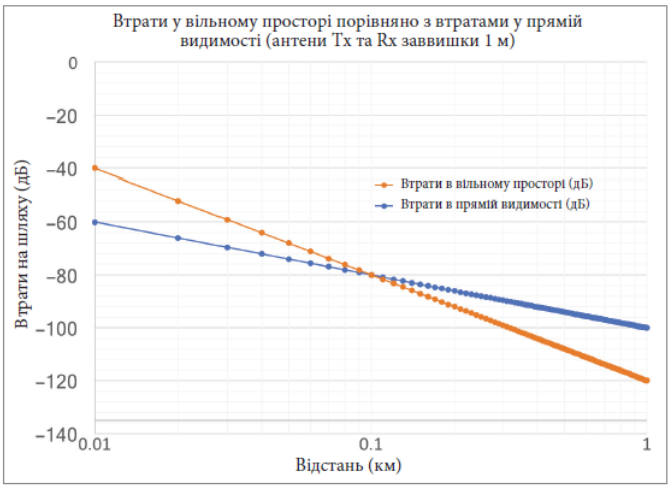


Рисунок 4.3 – Втрати у вільному просторі порівняно з плоскою землею (в дБ) з використанням сигналу 2,4 ГГц з антенами заввишки 1 метр

Таблиця 4.2 – Порівняння частот 900 МГц та 2,4 ГГц

	900 МГц	2,4 ГГц
Сила сигналу	В основному, надійний	Переповнений діапазон, схильний до інтерференції
Відстань	У 2,67х далі, ніж 2,4 ГГц	Коротше, але може компенсувати це покращеним кодуванням (Bluetooth 5)
Проникнення	Велика довжина хвилі дозволяє проникати через більшість матеріалів та перешкод.	Потенційно піддається впливу більшості будівельних матеріалів
Швидкість передачі	Обмежена	Від 2-х до 3-х разів швидше, ніж 900 МГц
Вплив на сигнал	Сигнал може залежати від високих об’єктів та перешкод, краще проходить через листя	Найменша ймовірність взаємодії каналу з певними об’єктами
Вплив на канал	Інтерференція з бездротовими телефонами 900 МГц, RFID сканерами, сигналами стільники, моніторами для дітей	Інтерференція з 802.11 Wi-Fi
Вартість	Середня	Низька

Ці рівняння дають теоретичну модель, однак ніякі аналітичні рівняння не дають точного передбачення для певних реальних сценаріїв, таких як багатошляхові втрати.

4.3 Радіочастотна інтерференція

У цьому розділі побачимо кілька нових схем зниження інтерференції сигналів. Це проблема для багатьох форм бездротової технології, оскільки спектр неліцензований та загальний. Через те, що може бути кілька пристроїв, що випромінюють радіочастотну енергію в загальному просторі, буде виникати інтерференція. Візьміть Bluetooth та 802.11 Wi-Fi; обидва працюють у загальному спектрі 2,4 ГГц, але залишаються працездатними навіть у перевантажених середовищах. Протокол Bluetooth Low Energy (BLE), буде випадковим чином вибирати один із каналів 40^{-2} МГц як форму стрибкоподібної зміни частоти. Ми бачимо на рисунку 4.4 одинадцять безкоштовних каналів на BLE, які мають 15% ймовірність колізій (тим більше, що 802.11 не перескакує між каналами). Нова специфікація Bluetooth 5 надає такі методи, як маски доступу до слотів для блокування областей Wi-

Fi зі списку частотних переходів. Тут описано діапазон ILM для Zigbee та Bluetooth Low Energy. Також показано можливе суперництво із трьома каналами Wi-Fi у спектрі 2,4 ГГц.

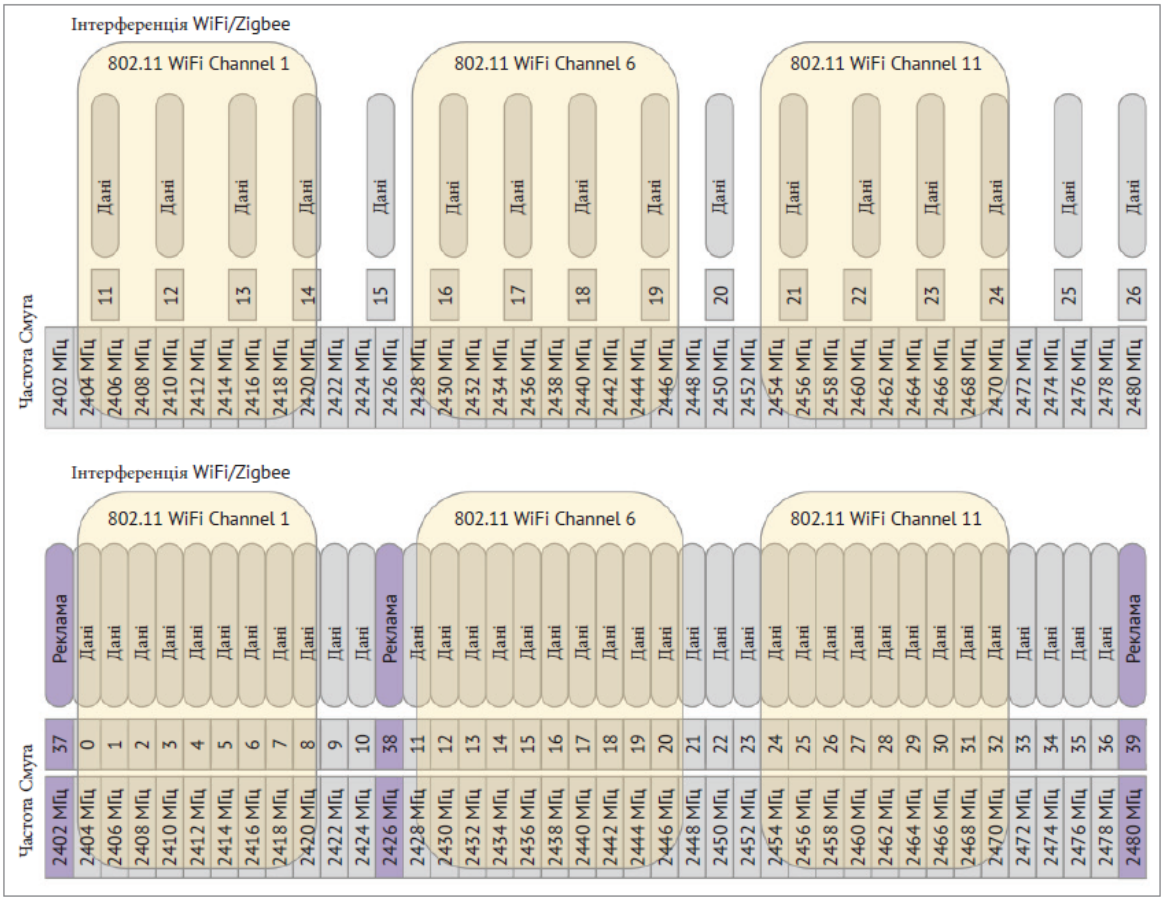


Рисунок 4.4 – Порівняння Bluetooth Low Energy (BLE) та Zigbee Interference з 802.11 Wi-Fi-сигналами в діапазоні 2,4 ГГц. BLE забезпечує більшу кількість слотів та стрибкоподібну перебудову частоти для зв'язку у разі колізій Wi-Fi

Існують попередні теорії, які потрібно розуміти, перш ніж деталізувати специфікацію WAN. Дві області, пов'язані з комунікацією – це те, як бітрейт впливає потужність передачі, що, своєю чергою, впливає діапазон. Як ми знаємо, існують обмеження цілісності даних та бітрейтів. Крім того, нам необхідно класифікувати вузькосмуговий та широкосмуговий зв'язок.

4.4 Межі бітрейту та теорема Шеннона-Хартлі

У комунікаціях на дальній відстані та ближньому зв'язку мета полягає у максимізації бітрейту та відстані в межах обмежень спектру та шуму. Теорема Шеннона-Хартлі складається з роботи Клода Шеннона з Массачусетського технологічного інституту у 1940-х роках. і Ральфа Хартлі з Bell Labs в 1920-х рр. Фундаментальна робота була зроблена Гаррі Найквістом, також Bell Labs, який визначив максимальну кількість імпульсів (або біт), які могли переміщатися по телеграфу за одиницю часу. По суті, Найквіст розробив межу вибірки, яка визначає, скільки теоретичної ширини смуги пропускання має задану частоту дискретизації. Це називається нормою Найквіста і показано в наступному рівнянні:

$$f_p \leq 2B,$$

тут f_p – частота імпульсів, а B – ширина лінії Герцах.

Це означає, що максимальний бітрейт обмежений удвічі більшою частотою дискретизації. Розглядаючи це інакше, рівняння ідентифікує мінімальний бітрейт, у якому потрібно вибірка сигналу кінцевої смуги пропускання, щоб зберегти всю інформацію. Скасування дискретизації призводить до ефекту згладжування та спотворення. Хартлі розробив спосіб кількісної оцінки інформації так званої

швидкості лінії. Швидкість лінії можна розрахувати як біти за секунду (наприклад, Мбіт/с). Це відомо як закон Хартлі, і він є попередником теореми Шеннона. Закон Хартлі просто стверджує, що максимальна кількість помітних амплітуд імпульсів, які можуть передаватися надійно, обмежена динамічним діапазоном сигналу та точністю, з якою приймач може точно інтерпретувати кожен окремий сигнал. Показано закон Хартлі у термінах *M* (кількість унікальних форм амплітуди імпульсу), що еквівалентно співвідношенню кількості напруги:

$$M = 1 + \frac{A}{\Delta V}.$$

Перетворення рівняння в логарифм на основі 2 дає нам швидкість лінії *R*:

$$R = f_p \log_2(M).$$

Якщо ми поєднаємо це з попередньою нормою Найквіста, ми отримаємо максимальну кількість імпульсів, які можуть бути передані по одному каналу пропускної спроможності *B*. Однак Хартлі не займався точністю; значення *M* (кількість окремих імпульсів) може впливати шум:

$$R \leq 2B\log_2(M).$$

Шеннон посилив рівняння Хартлі, розглядаючи ефекти гаусівського шуму, і завершив рівняння Хартлі відношенням сигнал/шум. Шеннон також представив концепцію кодування з виправленням помилок замість використання індивідуально помітних амплітуд імпульсів. Це рівняння тепер відоме як теорема Шеннона-Хартлі:

$$C = B\log_2\left(1 + \frac{S}{N}\right),$$

тут *C* – ємність каналу в бітах в секунду, *B* – смуга пропускання каналу в герцах, *S* – середній прийнятий сигнал, виміряний у ватах, а *N* – середній шум на каналі, виміряний у ватах. Ефект цього рівняння невеликий, але важливий. Для кожного рівня збільшення шуму сигналу в децибелах потужність різко падає. Аналогічним чином, покращення відношення сигнал-шум збільшить пропускну здатність. Без шуму потужність була б нескінченною. Також можна поліпшити теорему Шеннона-Хартлі, додавши в рівняння множник *n*. Тут *n* являє собою додаткові антени чи труби. Ми розглянули це як технологію множинного введення, множинного виведення (MIMO):

$$C = B \cdot n \cdot \log_2\left(1 + \frac{S}{N}\right).$$

Щоб зрозуміти, як правило, Шеннона відноситься до обмежень бездротових систем нам потрібно виразити рівняння з точки зору енергії на біт, а не співвідношення сигнал / шум (SNR). Корисним прикладом практично є визначення мінімального SNR, який буде необхідний досягнення певного бітрейту. Наприклад, якщо ми хочемо передати *C* = 200 кбіт/с каналом з пропускною здатністю *B* = 5000 кбіт/с, тоді мінімальне SNR буде дорівнювати:

$$C = B\log_2\left(1 + \frac{S}{N}\right),$$

$$200 = 500 \times \log_2\left(1 + \frac{S}{N}\right),$$

$$\frac{S}{N} = 0{,}028\,,$$

$$\frac{S}{N} = -15{,}528\ дБ\,.$$

Це показує, що можна передавати дані з використанням сигналу, який слабший за фоновий шум. Проте, існує межа швидкості передачі. Щоб показати ефект, нехай E_b представляє енергію одного біта даних у джоулях. Нехай N_o представляє спектральну щільність шуму у ватах / герцах. E_b / N_o є безрозмірною одиницею (як правило, вираженою в дБ), яка представляє SNR на біт, або широко відому як енергоефективність. Вирази енергоефективності усувають зміщення методів модуляції, кодування помилок та ширини смуги сигналу з рівняння. Припустимо, що система досконала та ідеальна, де $RB = C$,
де R – пропускна спроможність. Теорема Шеннона-Хартлі може бути переписана як:

$$\frac{C}{B} = \log_2\left(1 + \frac{E_b C}{N_0 B}\right),$$

$$\frac{E_b}{N_0} = \frac{2^{\frac{C}{B}} + 1}{\frac{C}{B}},$$

$$\frac{E_b}{N_0} \geq \lim_{\frac{C}{B}\to 0} \frac{2^{\frac{C}{B}} - 1}{\frac{C}{B}} = \ln(2) = 1.59\ дБ.$$

Це рівняння має назву межа Шеннона для адитивного білого гаусовского шуму (AWGN). AWGN є каналом і просто базовою формою шуму, що зазвичай використовується в теорії інформації для вираження ефектів випадкових процесів у природі. Ці джерела шуму завжди присутні у природі і включають такі речі, як теплові коливання, випромінювання чорного тіла та залишкові ефекти Великого Вибуху. «Білий» аспект шуму має на увазі рівну кількість шуму, що додається до кожної частоти. Межу можна намалювати на графіку, що показує спектральну ефективність порівняно з SNR на біт і показаний на рисунку 4.5.

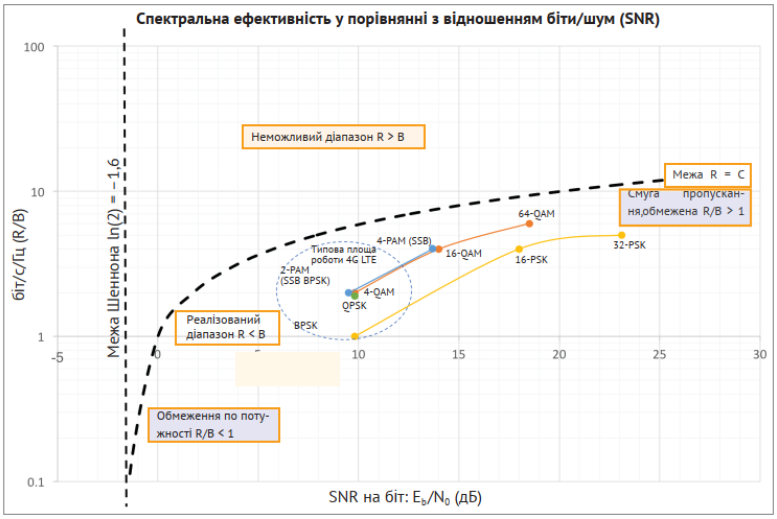


Рисунок 4.5 – Крива спектральної ефективності від SNR (енергетична ефективність). Пунктирна лінія є межею Шеннона, який сходиться при ln (2) = –1,6. Різні схеми модуляції показані межі Шеннона з типовим діапазоном сигналів 4G LTE

Області, що становлять інтерес на рисунку 4.5 включають R > B «Неможлива область». Ця область вище межі Шеннона для кривої. Він каже, що надійна форма обміну інформацією не може бути вищою за граничну. Область нижче межі Шеннона називається регіоном, що «реалізується», де R < B. Кожен протокол та технологія модуляції у будь-якій формі комунікації намагаються наблизитися якомога ближче до межі Шеннона. Ми можемо бачити, де знаходиться типовий 4G LTE з використанням різних форм модуляції.

Є ще дві області, які становлять інтерес. Область «Смуга пропускання обмежена» в напрямку праворуч догори допускає високу спектральну ефективність і хороші значення SNR EB / NO. Єдиним обмеженням у цьому просторі є компроміс з фіксованою або санкціонованою спектральною ефективністю проти необмеженої потужності передачі *P*, що означає, що ємність значно зросла доступною смугою пропускання. Протилежний ефект називається областю обмеженої потужності у напрямку нижнього лівого кута діаграми.

Область «Потужність обмежена» – це та, де SNR EB/NO дуже низька, тому межа Шеннона призводить до низьких значень спектральної ефективності. Тут ми втрачаємо спектральну ефективність для отримання заданої якості передачі P.

Література: [4, 5, 6, 8, 9].

ТЕМА 5
МАРШРУТИЗАТОРИ ТА ШЛЮЗИ

Інтернет Речей має, швидше, галузеву та економічну спрямованість через кількість пристроїв, які будуть розгорнуті, та обсяг даних, які ці пристрої будуть виробляти. Існує два методи щодо того, як формуватиметься IoT:

– граничні, або крайові, датчики та пристрої забезпечать прямий шлях до хмари. Це означає, що ці вузли та датчики граничного рівня матимуть достатньо ресурсів, апаратних засобів, програмного забезпечення та угод про рівень обслуговування для прямої передачі даних через WAN;

– датчики граничного рівня утворюють групи та кластери навколо шлюзів та маршрутизаторів для забезпечення проміжних областей, перетворення протоколів та можливостей обробки на в туманах та керуватимуть безпекою та автентифікацією між датчиками та глобальною мережею.

Перша модель є складною та дорогою для потужних та недорогих датчиків / перетворювачів / пристроїв граничного рівня. Більш логічним буде другий варіант. Роль граничного маршрутизатора або шлюзу для датчиків / пристроїв включає формальні мережеві можливості, які надають сучасні маршрутизатори, такі як маршрутизація каналів, переадресація портів, тунелювання, безпека і резервування. Принципи маршрутизації та ретрансляції TCP / IP описані у багатьох джерелах. У цій темі розглядається роль та необхідність маршрутизаторів граничного рівня, а також наведено поради та рекомендації щодо функцій, які слід враховувати при розгортанні масового рішення IoT.

5.1 Функції маршрутизації

В архітектурі IoT маршрутизатор відіграє значну роль у загальному управлінні, масштабуванні та безпеці системи. Часто буває, що роль маршрутизатора спрощується, щоб він діяв просто як шлюз з одного протоколу до іншого (перетворювач стільникового до Bluetooth). Для комерційного або промислового розгортання необхідно враховувати набагато більше, особливо коли пристрої знаходяться у віддалених та рухомих системах.

Розглянемо кілька типів бездротового зв'язку, кожен з яких вимагає будь-якої форми передавача та приймача, а також механізм перетворення трафіку для підключення до Інтернету. Це найважливіша роль граничного шлюзу IoT. Незалежно від того, чи є середовище Bluetooth-мережею, що вимагає вузла моста або eNodeB стільникової мережі. Шлюз перетворює та спрямовує дані між двома неподібними мережами. Маршрутизатор може бути шлюзом. Маршрутизатори направляють та керують трафіком між подібними мережами. В архітектурі IoT може бути встановлена 6LoWPAN, яка адресується IPv6, обмінюючись інформацією із зовнішнім світом через пристрій, який діє як шлюз, що з'єднує протоколи 802.15.4 з фізичним транспортом Wi-Fi або 802.3, але також направляючи і маршрутизуючи дані з використанням рівня, що розділяється між інтернетом і mesh-мережею 6LoWPAN. Часто граничний шлюз є центральним контролером PAN. Це означає, що всі функції керування мережею PAN, забезпечення безпеки, автентифікація та надання нових вузлів, даних керування та пристроїв керування живленням належать до відповідальності граничного шлюзу.

5.2 Маршрутизація

Основною функцією маршрутизатора є забезпечення з'єднань між сегментами мережі. Маршрутизація вважається функцією третього рівня стандартної моделі OSI, оскільки вона використовує рівень IP-адресації для керування передачею пакетів. Всі маршрутизатори покладаються на таблицю маршрутизації управління потоком даних. Таблиця маршрутизації використовується для пошуку найкращої відповідності IP-адреси призначення пакета. Існує кілька перевірених алгоритмів, що використовуються для ефективної маршрутизації. Одним типом маршрутизації є динамічна маршрутизація, де алгоритми реагують на зміни в мережі та топології.

Інформація про стан мережі поширюється протоколом маршрутизації за часом або за оновленням. Прикладами динамічної маршрутизації є маршрутизація вектора відстані та маршрутизація стану каналу. В якості альтернативи статична маршрутизація важлива і корисна для невеликих мереж, яким потрібні певні шляхи між маршрутизаторами. Статичні маршрути

неадаптивні, тому не потрібно сканувати топологію або оновлювати метрики. Вони встановлені на маршрутизаторі:

– маршрутизація найкоротшим шляхом – побудова графа, що представляє маршрутизатори у мережі. Дуга між вузлами є відомим зв'язком або з'єднанням. Алгоритм просто знаходить найкоротший шлях із будь-якого джерела до будь-якого пункту призначення;

– лавинна маршрутизація – кожен пакет повторюється та транслюється кожним маршрутизатором у кожну кінцеву точку його зв'язків. Це генерує величезну кількість дублюючих пакетів і вимагає, щоб у заголовку пакета було встановлено лічильник переходів, щоб гарантувати, що пакети мають обмежений час життя. Альтернативою є вибіркове розсилання, яке наповнює мережу лише в основному напрямку пункту призначення. Лавинні мережі є основою мереж Bluetooth;

– маршрутизація на основі потоку – перевіряє поточний потік у мережі до визначення шляху. Для будь-якого даного з'єднання, якщо відомі пропускна здатність та середній потік, обчислюється середня затримка пакета цього з'єднання. Цей алгоритм знаходить мінімальне середнє значення;

– маршрутизація на основі вектора відстані – таблиця маршрутизаторів містить найвідомішу відстань до кожного пункту призначення. Таблиці оновлюються сусідніми маршрутизаторами. Таблиця містить запис для кожного маршрутизатора підмережі. Кожен запис містить бажаний маршрут / шлях та приблизну очікувану відстань до пункту призначення. Відстань може бути метрикою кількості переходів, затримки чи довжини черги;

– маршрутизація станом каналу – маршрутизатор спочатку виявляє всіх своїх сусідів за допомогою спеціального пакету HELLO.

Маршрутизатор вимірює затримку для кожного зі своїх сусідів шляхом надсилання пакета ECHO. Ця інформація про топологію та час потім розсилається всім маршрутизаторам у підмережі. Повна топологія будується та публікується всіма маршрутизаторами:

– ієрархічна маршрутизація – маршрутизатори поділені на регіони та мають ієрархічну топологію. Кожен маршрутизатор підтримує розуміння свого регіону, але не всієї підмережі. Ієрархічна маршрутизація є ефективним засобом управління розміром таблиці маршрутизації та ресурсами в обмежених пристроях;

– широкомовна маршрутизація – кожен пакет містить список адрес призначення. Широкомовний маршрутизатор досліджує адреси та визначає набір вихідних ліній передачі пакета. Маршрутизатор генеруватиме новий пакет для кожної вихідної лінії і включатиме лише адресати, необхідні для цього новосформованого пакета;

– маршрутизація за допомогою багатоадресної розсилки: мережа поділена на чітко визначені групи. Програма може надсилати пакет усій групі, а не одному адресату або на широкомовну адресу.

Типові граничні маршрутизатори будуть підтримувати такі протоколи маршрутизації, як Border Gateway Protocol (BGP), Open Shortest Path First (OSPF), Routing Information Protocol (RIP) та RIPng. Архітектор, який використовує граничні маршрутизатори в реальній роботі, повинен знати про навантаження та вартість використання певного протоколу маршрутизації в порівнянні з іншими, особливо якщо з'єднання між маршрутизаторами є WAN-з'єднанням з обмеженням даних:

– BGP – BGP-4 є стандартом для протоколів інтернет-доменної маршрутизації та описаний у RFC 1771; він використовується більшістю інтернет-провайдерів. BGP – це алгоритм динамічної маршрутизації на основі вектора відстані, він рекламує цілі шляхи у повідомленнях оновлень маршрутизації. Якщо таблиці маршрутизації є більшими, це потребує значної пропускної спроможності. BGP надсилає 19-байтове повідомлення keepalive кожні 60 с для підтримки з'єднання. BGP може бути слабким протоколом маршрутизації для топології mesh-мережі, оскільки BGP підтримує з'єднання із сусідами. BGP також страждає від зростання таблиць маршрутизації у високих топологіях. BGP також унікальний, оскільки це один із єдиних протоколів маршрутизації на основі TCP-пакетів;

– OSPF – цей протокол описаний у RFC 2328, він забезпечує переваги масштабування мережі та збіжності. Інтернет-магістраль та корпоративні мережі інтенсивно використовують OSPF. OSPF – це алгоритм стану з'єднання, який підтримує IPv4 та IPV6 (RFC 5340) та працює з IP-пакетами. Перевага полягає у виявленні за секунди динамічних змін з'єднань та реагуванні;

– RIP – друга версія RIP є алгоритм маршрутизації вектора відстані, заснований на підрахунку переходів з використанням протоколу внутрішнього шлюзу. Спочатку заснований на алгоритмі

Беллмана-Форда, тепер він підтримує підмережі з розмірами, що змінюються, долаючи обмеження вихідної версії. Петлі у таблиці маршрутизації обмежені завданням максимальної кількості переходів у дорозі (15). RIP працює по UDP і підтримує лише трафік IPv4. RIP має триваліший час збіжності, ніж такі протоколи, як OSPF, але його легко адмініструвати для топологій невеликого граничного маршрутизатора. Тим не менш, збіжність для RIP з кількома маршрутизаторами може зайняти кілька хвилин;

– RIPng – RIPng означає RIP наступного покоління (RFC 2080). Він дозволяє підтримувати трафік IPv6 та IPsec для автентифікації.

5.3 Відмовостійкість та позасмугове управління

Відмовостійкість критична для деяких граничних маршрутизаторів IoT, особливо для додатків для транспортних засобів та догляду за пацієнтами. Відмовостійкість, як випливає з назви – це перемикання з одного інтерфейсу WAN на інший, коли основне джерело втрачено. Втрата WAN може бути пов'язана з втратою стільникового зв'язку в тунелі. Компанії-перевізнику з автопарком пересувних холодильних камер може знадобитися гарантоване підключення, при тому, що послуги стільникового зв'язку змінюються по всій країні. Перехід від одного стільникового оператора до іншого за допомогою декількох ідентифікаторів SIM-картки може допомогти пом'якшити та згладити перепідключення. Іншим варіантом може бути використання клієнтського Wi-Fi як основний інтерфейс WAN для внутрішнього моніторингу стану, але підключення до стільникової WAN заради відмовостійкості, якщо сигнал Wi-Fi втрачено. Відмовостійкість має бути безболісною та автоматичною без втрати пакетів або помітного впливу на затримку даних.

Позасмугове керування (OOBM) також має розглядатися для пристроїв IoT. OOBM корисний у відмовостійких умовах, коли для керування обладнанням необхідний виділений та ізольований канал. Іноді це називається керованим лампочками (LOM), якщо первинні системи вийшли з ладу, пошкоджені або мають втрату потужності; як і раніше, можна керувати і перевіряти обладнання віддалено через канал бічної смуги. В IoT це може бути корисно для ситуацій, що вимагають гарантованого часу безвідмовної роботи та дистанційного керування, таких як моніторинг нафти та газу або промислова автоматизація. Добре продумана система OOBM не повинна мати жодного відношення до контрольованої системи для свого функціонування. Типові способи управління, такі як тунелі VNC або SSH, вимагають, щоб пристрій був завантажений і функціонував. OOBM повинен бути додатковим та ізольованим від системи, як показано на рисунку 5.1.

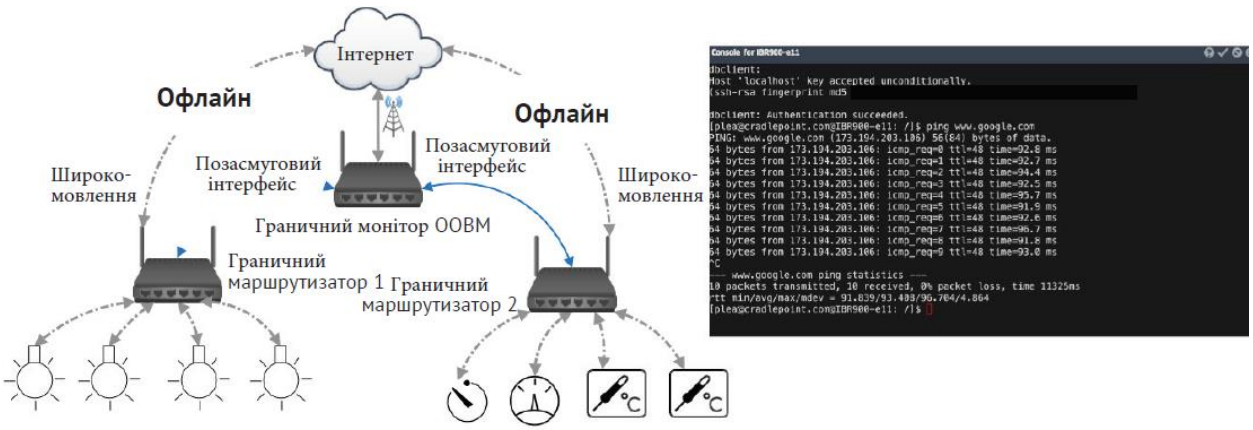


Рисунок 5.1 – Приклад конфігурації для позасмугового керування

5.4 VLAN

VLAN функціонує як будь-яка інша фізична локальна мережа, але дозволяє групувати комп'ютери та інші пристрої, навіть якщо вони фізично не прив'язані до одного мережного комутатора. Поділ відбувається лише на рівні лінії передачі (другий рівень) моделі OSI. VLAN – це форма мережевої сегментації пристроїв, програм або користувачів, хоча вони знаходяться в одній і тій самій фізичній мережі. VLAN також може групувати хости разом, хоча вони не знаходяться на одному мережному комутаторі, що суттєво полегшує розбиття на мережі без використання

додаткових кабелів. Стандарт IEEE 802.1Q – це стандарт, за яким побудовано VLAN. По суті VLAN використовує ідентифікатор або тег, що складається з 12 біт у кадрі Ethernet. Таким чином, існує жорстке обмеження 4096 потенційних VLAN в одній фізичній мережі. Комутатор може призначити порт безпосередньої прив'язки до певної VLAN. Оскільки VLAN працює на другому рівні стека, трафік може бути тунельований через третій рівень, що дозволяє географічно розділеним VLAN використовувати загальну топологію (рис. 5.2). Вище показано корпоративну точку продажу (POS) та систему VOIP, яка фактично ізольована від набору пристроїв IoT, а також гостьового Wi-Fi. Це робиться за допомогою адресації VLAN, хоча система використовує ту саму фізичну мережу. Тут ми припускаємо, що це інтелектуальне рішення IoT, де всі граничні пристрої та датчики IoT підтримують IP-стек та адресуються через локальну мережу.

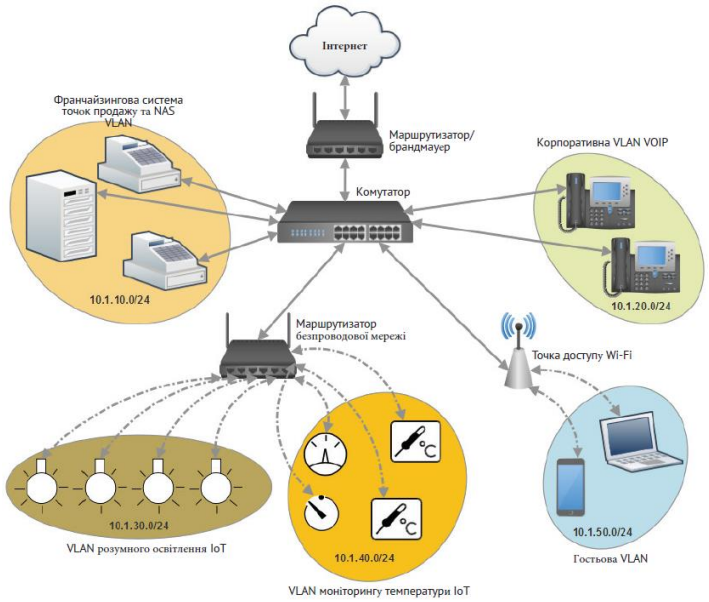


Рисунок 5.2 – Приклад архітектури VLAN у сценарії філій або роздрібної торгівлі

5.5 VPN

VPN-тунелі використовуються для встановлення безпечного підключення до віддаленої мережі через публічну мережу. Наприклад, VPN-тунелі можуть використовуватися через Інтернет для окремого користувача для підключення до захищеної корпоративної мережі під час поїздок або для з'єднання двох офісних мереж для роботи як одна мережа. Дві мережі налаштовують безпечне з'єднання через (зазвичай) незахищений інтернет, застосовуючи протоколи шифрування VPN.

Існує кілька VPN:

– Internet Protocol Security (IPSec) VPN – традиційна технологія VPN, яка знаходиться на мережевому рівні стека OSI та забезпечує передачу даних через тунель між двома кінцевими точками;

– OpenVPN – це VPN з відкритим вихідним кодом для безпечного з'єднання «точка-точка» та «місце-місце» у маршрутизованих чи мостових конфігураціях. Вона включає спеціальний протокол безпеки, який використовує SSL/TLS (OpenSSL) для обміну ключами та управління шифруванням та обміну даними;

– Generic Routing Encapsulation (GRE) – створює з'єднання «точка-точка» між кінцевими точками через тунель, аналогічний тунелю VPN, але інкапсулює його корисне навантаження. Він обертає внутрішній пакет у зовнішній пакет. Це дозволяє передавати дані корисного навантаження через інші IP-маршрутизатори та тунелі незміненими.

Крім того, тунелі GRE можуть транспортувати IPV6 та багатоадресні передачі;

– Layer 2 Tunneling Protocol (L2TP) – створює з'єднання між двома приватними мережами через дейтаграми UDP, які типово використовуються для VPN або як частина служб доставки провайдерами. У протоколі немає вбудованої безпеки або шифрування, і для цього часто використовується IPsec.

VPN повинен або довіряти базовим мережевим протоколам, або забезпечувати власну безпеку. VPN-тунелі зазвичай використовують IPsec для аутентифікації та шифрування пакетів, що передаються тунелями. Щоб налаштувати тунельний маршрутизатор VPN на одному кінці, має бути інший пристрій (зазвичай це маршрутизатор), який також підтримує IPsec на іншому кінці. Internet Key Exchange (IKE) – це протокол безпеки в IPsec. IKE має дві фази. Перша фаза відповідає встановлення безпечного каналу зв'язку, але в другому етапі встановлений канал використовується партнерами IKE. Маршрутизатор має кілька параметрів протоколу безпеки для кожної фази, але вибір за промовчанням буде достатнім для більшості користувачів. Кожен обмін IKE використовує один алгоритм шифрування, одну хеш-функцію та одну групу DH для безпечного обміну:

– Encryption – використовується для шифрування повідомлень, що надсилаються та одержуються по IPsec. Типові стандарти та алгоритми шифрування включають AES 128, AES 256, DES та 3DES;

– Hash – використовується для порівняння, аутентифікації та перевірки даних за VPN, забезпечення того, що вони приходять у правильній формі, та для отримання ключів, які використовуються IPSec. Типові функції хешу, які слід очікувати в маршрутизаторі корпоративного рівня, включають MD5, SHA1, SHA2 256, SHA2 384 і SHA2 512. Зверніть увагу, що деякі комбінації шифрування/хешу (такі як 3DES з SHA2 384/512) є дорогими що впливає продуктивність WAN. AES забезпечує хороше шифрування та працює набагато швидше, ніж 3DES;

– групи DH: група DH (Diffie-Hellman) є властивістю IKE та використовується для визначення довжини простих чисел, пов'язаних із генерацією ключів. Надійність генерованого ключа частково визначається надійністю групи DH. Група 5, наприклад, має більшу міцність, ніж група 2:

– група 1: 768-бітний ключ;

– група 2: 1024-бітний ключ;

– група 5: 1536-бітний ключ.

У першій фазі IKE можна вибрати одну групу DH, лише якщо використовуєте агресивний режим обміну. Алгоритми перераховані у порядку пріоритету. Ви можете змінити порядок цього списку пріоритетів, клацнувши та перетягнувши алгоритми вгору або вниз. Будь-який обраний алгоритм можна використовувати для IKE, але алгоритми у верхній частині списку, ймовірно, будуть використовуватися частіше.

5.6 Управління швидкістю трафіку та QoS

Функції керування швидкістю трафіку та якості послуг (QoS) корисні при розгортанні, що вимагає гарантованого рівня обслуговування під час роботи з перевантаженнями або змінними навантаженнями на мережу. Наприклад, у разі використання IoT при змішуванні живих відеопотоків і публічного Wi-Fi, відеопотоки можуть потребувати пріорітизації та гарантованого рівня якості, особливо для громадської безпеки або спостереження. Залишені дані, що надходять, від глобальної мережі до прикордонного маршрутизатора обслуговуватимуться за принципом «першим прийшов – першим обслужили»:

– функції QoS – дозволяють адміністратору призначати рівні пріоритету для заданої IP-адреси, розміщеної на маршрутизаторі або певному порту. Функції QoS керують лише каналом висхідної лінії зв'язку. Вони особливо корисні в тих випадках, коли канал висхідної лінії зв'язку має набагато меншу пропускну здатність, ніж низхідна лінія. Як правило, споживчий широкосмуговий доступ матиме щось на кшталт висхідної лінії 5 Мбіт/с і низхідній лінії зв'язку 100 Мбіт/с, а QoS забезпечує спосіб балансування навантаження на обмежену висхідну лінію. QoS не призначає жорсткі ліміти і сегментує з'єднання, як це робить управління швидкістю трафіку;

– функції керування швидкістю трафіку – керування швидкістю трафіку – це статична форма зумовлення смуги пропускання. Наприклад, зв'язок 15 Мбіт/с може бути поділено на менші сегменти по 5 Мбіт/с. Ці сегменти мають бути призначені заздалегідь. Як правило, це зайві витрати, оскільки виділені частини смуги пропускання можуть не звільнятися у разі потреби;

– динамічне керування швидкістю та пріоритет пакетів – сучасні маршрутизатори підтримують атрибути динамічного керування швидкістю. Це дозволяє адміністратору динамічно призначати правила сегментації смуги пропускання для вхідного та вихідного трафіку. Він також може керувати чутливими до затримок пакетами (наприклад, відео або інтерфейсом користувача) для

додатків реального часу. Динамічне керування швидкістю та пріоритет пакетів дозволяють створювати правила на основі типу даних або програми, а не лише IP-адреси чи порту.

Іншим аспектом якості мережі є середня оцінка (MOS). MOS – середнє арифметичне окремих значень за шкалою якості системи з погляду користувача. Це зазвичай використовується в додатках Голос по протоколу інтернет (VOIP), але, безумовно, може використовуватися для систем відеоспостереження, обробки зображень, потокової передачі даних та для зручності використання інтерфейсу користувача. Він ґрунтується на суб'єктивному рейтингу від одного до п'яти (де одиниця означає найгіршу якість, п'ять – найкращу якість), і його слід використовувати у петлі зворотного зв'язку для збільшення ємності або зменшення розмірів даних для відповідності ємності.

5.7 Функції безпеки

Граничний маршрутизатор або шлюз виконує ще одне важливе завдання, забезпечуючи безпеку між WAN, Інтернетом і базовими пристроями PAN / IoT. Багатьом пристроям не вистачає необхідних ресурсів, пам'яті та обчислювальної потужності для забезпечення надійності безпеки та забезпечення. Незалежно від того, чи створює архітектор свій власний шлюзовий сервіс або набуває його, для захисту компонентів IoT слід враховувати наступний перелік властивостей. Захист за допомогою брандмауера – базова форма безпеки. Існують дві основні форми брандмауерів для телекомунікацій. Перший – це мережевий брандмауер, який фільтрує та керує потоком інформації з однієї мережі до іншої. Другий – це брандмауер на базі хоста, який локально захищає програми та служби на цій машині. У випадку граничних маршрутизаторів IoT ми фокусуємось на мережевих брандмауерах. За замовчуванням брандмауер запобігає проникненню певних типів мережного трафіку в зону, захищену брандмауером, але будь-який трафік, що походить із цієї зони, може йти назовні. Брандмауер знайде та ізолює інформацію на основі пакетів, станів або програм залежно від складності брандмауера. Як правило, зони створюються за інтерфейсами мережі з правилами, призначеними для управління потоком трафіку між зонами. Прикладом може служити граничний маршрутизатор із гостьовою зоною Wi-Fi та корпоративною приватною зоною. Пакетний брандмауер може ізолювати та пригнічувати певний трафік на основі IP-адреси джерела або одержувача, портів, MAC-адрес, IP-протоколів та іншої інформації, що міститься в заголовку пакета. Брандмауер із станом працює на четвертому рівні стеку OSI. Він збирає та об'єднує пакети, шукаючи шаблони та інформацію про стан, таку як нові з'єднання та існуючі з'єднання. Фільтрація додатків ще складніша, тому що вона може здійснювати пошук за певними мережевими потоками додатків, включаючи FTP-трафік або HTTP-дані. Брандмауер також може використовувати демілітаризовану зону (DMZ). DMZ – це логічна зона. Хост DMZ ефективно не захищений брандмауером у тому сенсі, що будь-який комп'ютер в інтернеті може спробувати віддалено отримати доступ до мережевих служб IP-адреси DMZ. Типові види використання включають запуск загального веб-сервера та обмін файлами. Хост DMZ зазвичай задається прямою IP-адресою. Переадресація портів – це концепція, що дозволяє відкривати певні порти за брандмауером. Для декількох пристроїв IoT необхідний відкритий порт для надання сервісів, які контролюються компонентами хмар. Знову ж таки, побудовано правило, яке дозволяє зазначеній IP-адресі в захищеній зоні брандмауера мати відкритий порт.

Література: [1, 3, 4, 7, 8].

ТЕМА 6
ІОТ-ПРОТОКОЛИ ПЕРЕДАЧІ ДАНИХ ВІД ГРАНИЧНОГО ПРИСТРОЮ В ХМАРУ

6.1 Протоколи

При передачі даних в Інтернет технологія переноситься на фундаментальні рівні TCP/IP. Протоколи TCP і UDP є очевидним і єдиним вибором передачі даних, TCP значно складніше у реалізації, ніж UDP. Однак UDP не має стабільності та надійності TCP, змушуючи в деяких розробках компенсувати це додаванням відмовостійкості на рівнях додатків вище UDP.

Багато протоколів є реалізаціями проміжного ПЗ, орієнтованого повідомлення (MOM). Основна ідея MOM полягає в тому, що зв'язок між двома пристроями відбувається з використанням розподілених черг повідомлень. MOM надсилає повідомлення від однієї програми в просторі користувача іншому. Деякі пристрої виробляють дані для додавання до черги, в той час як інші споживають дані, що знаходяться у черзі. Деякі реалізації вимагають, щоб центральним сервісом був брокер чи посередник. У цьому випадку виробники та споживачі мають публіцистичні та підписні зв'язки з брокером. AMQP, MQTT та STOMP є реалізаціями MOM; інші включають служби обміну повідомленнями CORBA та Java. Реалізація MOM з використанням черг може використовувати їх для стійкості розробки. Дані можуть зберігатися в чергах, навіть якщо сервер відмовить.

Альтернативою реалізації MOM є RESTful. У моделі RESTful сервер має стан ресурсу, але стан не передається у повідомленні від клієнта на сервер. RESTful використовує HTTP-методи, такі як GET, PUT, POST та DELETE для розміщення запитів щодо універсального ідентифікатора ресурсу (URI) (рис. 6.1). Ця архітектура не потребує брокера або посередника. Оскільки вони засновані на стеку HTTP, вони користуються більшістю сервісів, таких як безпека HTTPS. Проекти RESTful типові для клієнт-серверних архітектур. Клієнти ініціюють доступ до ресурсів через синхронні шаблони запиту-відповіді. Крім того, клієнти відповідають за помилки, навіть якщо сервер падає. На рисунку 6.1 показано MOM порівняно з сервісом RESTful. Зліва знаходиться служба обміну повідомленнями (на основі MQTT), яка використовує серединний брокерський сервер, видавців та передплатників подій. Тут багато клієнтів можуть бути як видавцями, так і передплатниками, і інформація може зберігатися або не зберігатися у черзі для швидкого відновлення. Праворуч знаходиться проект RESTful, де архітектура побудована на HTTP та використовує HTTP-парадигми для зв'язку від клієнта до сервера.

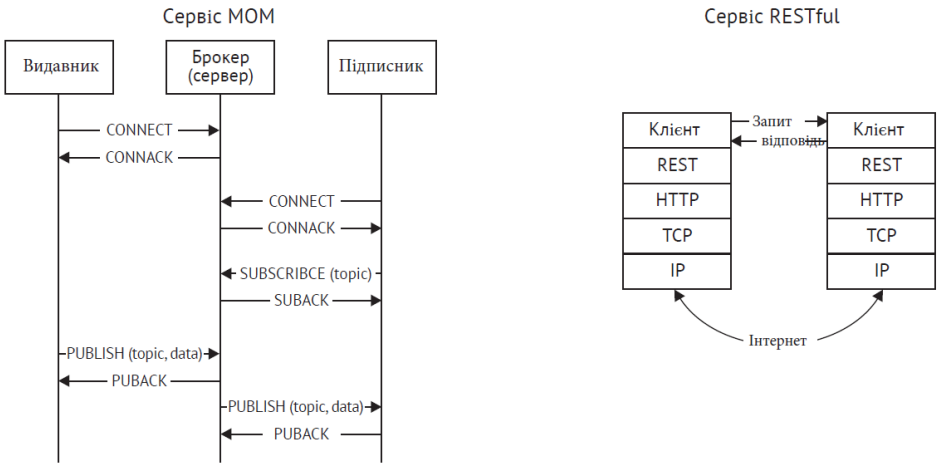


Рисунок 6.1 – Приклад для порівняння MOM з реалізацією RESTful

6.2 MQTT

Технологія IBM Websphere Message Queue була вперше вигадана у 1993 р. для вирішення проблем у незалежних та неконкурентних розподілених системах для забезпечення захищеного зв'язку. Похідна від Web Sphere Message Queue була створена Енді Стенфордом-Кларком та Арленом Ніппером в IBM у 1999 р. для вирішення конкретних проблем, пов'язаних із підключенням віддалених нафто- та газопроводів через супутниковий зв'язок. Цей протокол став відомий як MQTT. Цілі цього транспортного протоколу, що базується на IP, наступні:

– він має бути простим у реалізації;
– забезпечувати форму якості обслуговування;
– бути дуже легким та ефективним з точки зору пропускної спроможності;
– бути незалежним від платформ;
– постійне відстеження сеансу;
– вирішення проблем безпеки.

MQTT забезпечує виконання всіх цих вимог. При розгляді протоколу найкраще користуватися стандартним сайтом (mqtt.org), на якому наведена добре відома вибірка з опису протоколу: MQTT був внутрішнім та пропрієтарним протоколом для IBM протягом багатьох років, доки не був випущений у версії 3.1 у 2010 р. як безкоштовний продукт. У 2013 р. MQTT був стандартизований та прийнятий до консорціуму OASIS. У 2014 році OASIS опублікував його публічно як версію MQTT 3.1.1. MQTT також є стандартом ISO (ISO/IECPRF 20922).

У той час як архітектури клієнт-сервер багато років є основою для сервісів центрів обробки, моделі видання-підписка є альтернативою, яка корисна для використання IoT. Видання-підписка, також відома як pub/sub, є способом відокремити клієнта, який передає повідомлення від іншого клієнта, який отримує повідомлення. На відміну від традиційної моделі клієнт-сервер, клієнти не обізнані про будь-які фізичні ідентифікатори, на зразок IP-адреси або порту. MQTT – це архітектура pub/sub, але не черга повідомлень. Черги повідомлень за своєю природою зберігають повідомлення, а MQTT – ні. У MQTT, якщо ніхто не підписується (або не слухає) на тему, вона просто ігнорується та губиться. Черги повідомлень також підтримують топологію клієнт-сервер, де один споживач з'єднаний з одним виробником.

Клієнт, який передає повідомлення, називається видавцем; клієнт, який отримує повідомлення, називається передплатником. У центрі знаходиться брокер MQTT, який відповідає за з'єднання клієнтів і фільтрацію даних. Такі фільтри забезпечують:

– фільтрації за темами –клієнти підписуються на теми та певні гілки тому і не отримують даних більше, ніж хочуть. Кожне опубліковане повідомлення має містити тему і брокер несе відповідальність за повторну передачу цього повідомлення передплатникам або ігнорування його;

– фільтрація за типом – клієнт, який прослуховує потік даних, куди він підписаний, може також використовувати свої власні фільтри. Вхідні дані можуть аналізуватися і залежно від цього потік даних обробляється далі або ігнорується. У MQTT може бути багато виробників та багато споживачів, як показано на рисунку 11.2.

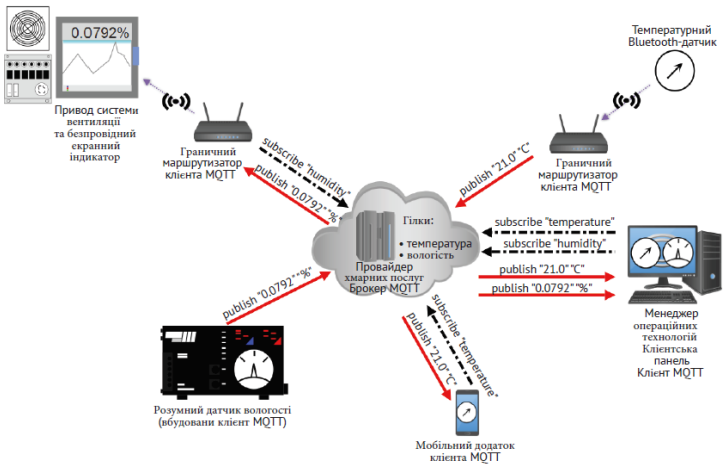


Рисунок 11.2 – Модель та топологія видання-підписка MQTT. Клієнти працюють на краю, публікують та / або підписуються на теми, керовані брокером MQTT

Тут розглянуто дві теми: вологість та температура. Клієнт може передплатити кілька тем. На рисмунку представлені розумні датчики, які мають достатні ресурси для управління власним клієнтом MQTT, а також граничні маршрутизатори, які надають клієнтські послуги MQTT від імені датчиків або пристроїв, які не підтримують MQTT.

MQTT успішно відокремлює видавців від споживачів. Оскільки брокер є керівним органом між видавцями та споживачами, немає необхідності безпосередньо ідентифікувати видавця та споживача на основі фізичних даних (таких як IP-адреса). Це корисно при розгортанні IoT, оскільки

фізичний ідентифікатор може бути невідомим чи загальним. MQTT та інші моделі pub/sub також є тимчасово незалежними. Це означає, що повідомлення, опубліковане одним клієнтом, передплатником може прочитати та відповісти на нього у будь-який час. Абонент може перебувати в місці з дуже низьким енергоспоживанням / обмеженою смугою пропускання (наприклад, Sigfox communication) та відповісти на повідомлення за кілька хвилин або годин. Через відсутність фізичних та тимчасових відносин моделі pub / sub добре підходять для підвищення продуктивності. Керовані хмарно брокери MQTT зазвичай можуть поглинати мільйони повідомлень на годину та підтримувати десятки тисяч видавців. MQTT не залежить від формату даних. Корисне навантаження може містити будь-який тип даних, тому і видавці, і передплатники повинні розуміти та узгоджувати формат даних. Можна надсилати текстові повідомлення, дані зображення, аудіодані, зашифровані дані, двійкові дані, об'єкти JSON або будь-яку іншу структуру в корисному навантаженні. Проте текстові та двійкові дані JSON є найпоширенішими типами даних корисного навантаження.

6.3 Деталі архітектури MQTT

Сама назва MQTT є неточною. У протоколі немає черги повідомлень. Хоча можна надсилати повідомлення у чергу, це необов'язково і часто не робиться. MQTT базується на TCP і тому є певна гарантія того, що пакет передається надійно. MQTT також є асиметричним протоколом, тоді як HTTP – це несиметричний протокол. Скажімо, вузол A повинен зв'язуватися з вузлом B. Асиметричний протокол між A і B вимагає, щоб протокол використовувала лише одна сторона (A), проте вся інформація, необхідна для повторного збирання пакетів, повинна міститися в заголовку фрагментації, надісланому A. В асиметричних системах є один ведучий та один ведений (FTP – класичний приклад). У симетричному протоколі він встановлений як на A, так і на B. A або B можуть приймати він роль ведучого або веденого (основний приклад – telnet). У MQTT ролі різні, що має сенс у топології датчиків/хмар. MQTT може зберігати повідомлення в брокері необмежено довго. Цей режим роботи керується прапором під час нормальної передачі повідомлення. Збережене на брокері повідомлення надсилається будь-якому клієнту, який підписується на цю тематичну гілку MQTT. Повідомлення негайно надсилається цьому новому клієнту. Це дозволяє новому клієнту отримати статус або сигнал із теми, на яку він нещодавно підписався, без очікування. Як правило, клієнт, який підписується на тему, може очікувати години або навіть дні, перш ніж клієнт опублікує нові дані. MQTT визначає додатковий об'єкт під назвою Остання воля та заповіт (LWT). LWT – це повідомлення, яке показує клієнт під час фази підключення. LWT містить тему «Остання воля», QoS та фактичне повідомлення. Якщо клієнт неправильно відключається від брокерського з'єднання (наприклад, тайм-аут keep-alive, помилка введення-виводу або клієнт закриває сеанс без відключення), тоді брокер зобов'язаний транслювати повідомлення LWT всім іншим клієнтам, що підписані на цю тему.

Незважаючи на те, що MQTT заснований на TCP, з'єднання все ще можуть обриватися, особливо у випадку бездротових датчиків. Пристрій може втратити живлення, втратити сильний сигнал або може бути просто польова поломка, і сеанс перейде у напіввідкритий стан. Тоді сервер буде вважати, що з'єднання, як і раніше, надійне і очікувати дані. Щоб вийти із цього напіввідкритого стану, MQTT використовує систему keep-alive. Використовуючи цю систему як брокер MQTT, так і клієнт мають гарантію того, що з'єднання залишається працездатним, навіть якщо протягом деякого часу не було передачі. Клієнт відправляє пакет PINGREQ брокеру, який, у свою чергу, підтверджує повідомлення за допомогою PINGRESP. Таймер встановлений на стороні клієнта та брокера. Якщо повідомлення не було передано ні ким із них протягом заданого проміжку часу, має бути надіслано пакет keep-alive. Як PINGREQ, так і повідомлення, скинуть таймер keep-alive. Якщо keep-alive не отримано і час таймера закінчується, брокер закриє з'єднання і відправить LWT-пакет всім клієнтам. Клієнт може в якийсь момент пізніше спробувати знову підключитись. У цьому випадку брокер закриває напіввідкрите з'єднання і відкриває нове з'єднання з клієнтом.

Коли keep-alive допомагає з порушеними з'єднаннями, повторне встановлення всіх підписок клієнта і параметрів QoS може призвести до непотрібних витрат на підключеному з'єднанні. Щоб зменшити ці додаткові витрати, MQTT дозволяє підтримувати постійні з'єднання. Постійне з'єднання зберігає на стороні брокера наступне:

– усі підписки клієнта;

– усі повідомлення QoS, які не були підтверджені клієнтом;

– нові повідомлення QoS, пропущені клієнтом.

Параметр client_id посилається на інформацію для унікальної ідентифікації клієнтів. Клієнт може запитувати постійне з'єднання, проте брокер може відхилити запит та примусово перезапустити чистий сеанс. Під час з'єднання брокером використовується прапорець cleanSession для дозволу чи заборони постійних з'єднань. Клієнт може визначити, чи зберігалося постійне з'єднання за допомогою повідомлення CONNACK.

У MQTT є три рівні якості обслуговування:

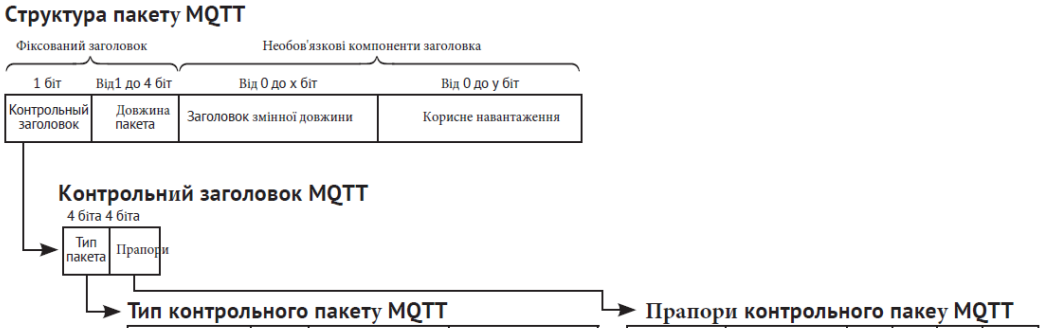
– QoS-0 (незавірена передача) – це мінімальний рівень QoS. Це аналогічно моделі «спалити і забути». Це найефективніший процес доставки без підтвердження одержувачем повідомлення та без повторної передачі повідомлення відправником;

– QoS-1 (гарантована передача) – цей режим гарантує доставку повідомлення хоча б один раз отримувачу. Повідомлення може бути доставлено кілька разів, та одержувач відправить назад підтвердження з відповіддю PUBACK;

– QoS-2 (гарантований сервіс для програм) – це найвищий рівень QoS, який переконується у доставці та інформує відправника та одержувача, що повідомлення було надіслано правильно. Цей режим генерує більше трафіку через багатокрокове рукостискання між відправником і одержувачем. Якщо одержувач отримує повідомлення з рівнем обслуговування QoS-2 він відповість відправнику повідомленням PUBREC. Це підтверджує повідомлення, та відправник відповість повідомленням PUBREL. PUBREL дозволяє одержувачу безпечно відкинути будь-які повторні передачі повідомлення. Потім PUBREL підтверджується одержувачем з допомогою PUBCOMP . Поки повідомлення PUBCOMP не буде надіслано, приймач кешуватиме вихідне повідомлення для забезпечення безпеки.

6.4 Структура пакету MQTT

Пакет MQTT знаходиться зверху рівня TCP мережевого стека в моделі OSI. Пакет складається з 2-байтового фіксованого заголовка, який завжди повинен бути присутнім, заголовка з змінним розміром (необов'язково) та закінчується корисним навантаженням (необов'язково) (рис. 6.3).



Тип контрольного пакету MQTT

Ім'я	Значення	Напрям	Опис
Зарезервовано	0	Заборонено	Зарезервовано
CONNECT	1	Від клієнта до сервера	Клієнтський запит з'єднання
CONNACK	2	Від сервера до клієнта	З'єднання ACK
PUBLISH	3	Від клієнта до сервера або від сервера до клієнта	Публікація повідомлення
PUBACK	4	Від клієнта до сервера або від сервера до клієнта	Публікація ACK
PUBREC	5	Від клієнта до сервера або від сервера до клієнта	Публікація отримання (гарантована доставка, частина 1)
PUBREL	6	Від клієнта до сервера або від сервера до клієнта	Публікація отримання (гарантована доставка, частина 2)
PUBCOMP	7	Від клієнта до сервера або від сервера до клієнта	Публікація завершена (гарантована доставка, частина 3)
SUBSCRIBE	8	від клієнта до сервера	Запит підписки клієнта
SUBACK	9	від сервера до клієнта	Підписка ACK
UNSUBSCRIBE	10	від клієнта до сервера	Запит відписки
UNSUBACK	11	від сервера до клієнта	Відписка ACK
PINGREQ	12	від клієнта до сервера	Запит PING
PINGRESP	13	від сервера до клієнта	Відповідь PING
DISCONNECT	14	від клієнта до сервера	Відключення клієнта
Зарезервовано	15	Заборонено	Зарезервовано

Прапори контрольного пакеу MQTT

Контрольний пакет	Фіксовані прапори заголовка	Біт 3	Біт 2	Біт 1	Біт 0
CONNECT	Зарезервовано	0	0	0	0
CONNACK	Зарезервовано	0	0	0	0
PUBLISH	Використовується MQTT 3.1.1	DUP	QoS	QoS	RETAIN
PUBACK	Зарезервовано	0	0	0	0
PUBREC	Зарезервовано	0	0	0	0
PUBREL	Зарезервовано	0	0	1	0
PUBCOMP	Зарезервовано	0	0	0	0
SUBSCRIBE	Зарезервовано	0	0	1	0
SUBACK	Зарезервовано	0	0	0	0
UNSUBSCRIBE	Зарезервовано	0	0	1	0
UNSUBACK	Зарезервовано	0	0	0	0
PINGREQ	Зарезервовано	0	0	0	0
PINGRESP	Зарезервовано	0	0	0	0
DISCONNECT	Зарезервовано	0	0	0	0

Рисунок 6.3 – Загальна структура пакетів MQTT

З’єднання з використанням MQTT починається з того, що клієнт надсилає повідомлення CONNECT брокеру. Тільки клієнт може ініціювати сеанс, та жоден клієнт не може безпосередньо зв’язатися з іншим клієнтом. У відповідь на повідомлення CONNECT брокер завжди буде надсилати CONNACK і код статусу. Після встановлення з’єднання залишається відкритим. У таблиці 6.1 наведено повідомлення та формати MQTT. Формат CONNECT (клієнт до сервера): типове повідомлення CONNECT міститиме дані (для запуску сеансу потрібно лише clientID), показані в таблиці 6.1.

Таблиця 6.1 – Повідомлення та формати MQTT

Поле	Необхідність	Опис
clientID	Потрібно	Ідентифікує клієнта на сервері. Кожен клієнт має унікальний ідентифікатор клієнта. Він може становити від 1 до 23 байтів UTF-8
cleanSession	Необов’язково	0: сервер має відновити зв’язок із клієнтом. Клієнт та сервер повинні зберегти стан сеансу після вимкнення. 1: Клієнт та сервер повинні скасувати попередній сеанс та розпочати новий
username	Необов’язково	Ім’я сервера для аутентифікації
password	Необов’язково	Двійковий пароль довжиною від 0 до 65536 байтів із префіксом 2 байти
lastWillTopic	Необов’язково	Тема гілки для публікації повідомлення
lastWillQos	Необов’язково	2 біти із зазначенням рівня QoS при публікації повідомлення останньої волі
lastWillMessage	Необов’язково	Визначає корисне навантаження повідомлення останньої волі
lastWillRetain	Необов’язково	Вказує, чи зберігається остання воля після публікації

Коди повернення CONNECT (від сервера до клієнта): брокер буде відповідати повідомлення CONNECT з кодом відповіді. Архітектор повинен знати, що не всі з’єднання можуть бути схвалені брокером. Коди відповіді показані у таблиці 6.2.

Таблиця 6.2 – Коди повернення CONNECT (від сервера до клієнта)

Код повернення	Опис
0	Успішне з’єднання
1	У з’єднанні відмовлено: неприйнятна версія протоколу MQTT
2	У з’єднанні відмовлено: ідентифікатор клієнта – це правильний UTF-8, але не дозволений сервером
3	У з’єднанні відмовлено: сервер недоступний
4	У з’єднанні відмовлено: неправильне ім’я користувача або пароль
5	У з’єднанні відмовлено: клієнт не авторизований для з’єднання

Формат PUBLISH (клієнт до сервера): на даний момент клієнт може публікувати дані в гілки теми. Кожне повідомлення містить тему (табл. 6.3).

Таблиця 6.3 – Формат PUBLISH (клієнт до сервера)

Поле	Необхідність	Опис
packetID	Потрібно	Унікально ідентифікує пакет у змінному заголовку. У зоні відповідальності клієнтської бібліотеки. Завжди встановлений на нуль (0) для QoS-0
topicName	Потрібно	Теми гілки для публікації (наприклад, США / Вісконсін / Мілуокі / температура)
qos	Потрібно	QoS рівня 0, 1 або 2
retainFlag	Потрібно	Ім’я сервера для аутентифікації
payload	Необов’язково	Корисне навантаження у будь-якому форматі
dupFlag	Потрібно	Повідомлення є дублікатом і надіслано повторно

Формат SUBSCRIBE (від клієнта до сервера): корисне навантаження пакета передплати включає щонайменше одну пару кодованих UTF-8 topicID і рівні QoS. У цьому корисному навантаженні може бути вказано кілька топи ID, щоб позбавити клієнта від декількох передач (табл. 6.4).

Таблиця 6.4 – Формат SUBSCRIBE (від клієнта до сервера)

Поле	Необхідність	Опис
packetID	Потрібно	Унікально ідентифікує пакет у змінному заголовку. Відповідальність клієнтської бібліотеки
topic_1	Потрібно	Тема теми гілки
qos_1	Потрібно	Рівень обслуговування QoS-повідомлень, опублікованих у topic_1
topic_2	Необов'язково	Ім'я сервера для аутентифікації
qos2	Необов'язково	Рівень обслуговування QoS-повідомлень, опублікованих у topic_2

Підстановочні знаки можуть використовуватись для передплати кількох тем в одному повідомленні. Для цих прикладів темою буде повний шлях «{країна}/{штати}/{міста}/{температура, вологість}».

– + підстановковий шаблон одного рівня – замінює один рівень у імені рядка теми. Наприклад, US /+/ Milwaukee замінить рівень штату на всі 50 штатів, від Аляски до Вайомінга;

– * багаторівневий шаблон – замінює кілька рівнів, а чи не один. Він завжди є останнім символом у назві теми. Наприклад, US / Wisconsin / * будуть відповідати всім містам Вісконсіна: Мілуокі, Медісон, Глендейл, Уайтфіш-Бей, Брукфілд і так далі;

– $ Спеціальні теми – це спеціальний статистичний режим для брокерів MQTT. Клієнти не можуть публікувати у темах $. Наразі офіційного стандарту для використання немає. Одна з моделей використовує $ SYS таким чином: $ SYS / broker / clients / connected.

Похідна для мереж датчиків MQTT називається MQTT-SN (іноді звучить як MQTT-S). Вона дотримується тієї ж філософії MQTT, як і легкий протокол для периферійних пристроїв, але сконструйована спеціально для нюансів бездротової локальної мережі, що є типовою для сенсорних середовищ. Це означає підтримку каналів з низькою пропускною здатністю, відстеження відмови каналу, короткі повідомлення та апаратне забезпечення з обмеженими ресурсами. MQTT-SN, по суті, настільки прозорий, що може успішно працювати поверх BLE та Zigbee.

MQTT-SN не вимагає стек TCP/IP. Його можна використовувати за послідовним з'єднанням (переважний варіант), де простий протокол зв'язку (щоб розрізняти різні пристрої на лінії) та накладні витрати дуже малі. Як альтернатива він може використовуватися протокол UDP, який вимагає менше ресурсів, ніж TCP.

Література: [3, 4, 7, 8, 9]

ТЕМА 7
ТОПОЛОГІЯ ХМАРНИХ ТА ТУМАННИХ ОБЧИСЛЕНЬ

7.1 Модель хмарних сервісів

Хмарні провайдери зазвичай підтримують цілу низку продуктів «Все як сервіс» (XaaS). Тобто послуга програмного забезпечення з оплатою за використання. Сервіс включає службу мережі (NaaS), програмне забезпечення як послугу (SaaS), платформу як послугу (PaaS) та інфраструктуру як послугу (IaaS). Кожна модель представляє все більше хмарних послуг від постачальників. Ці сервісні пропозиції – додана вартість хмарних обчислень. Як мінімум, ці послуги повинні компенсувати капітальні витрати, з якими стикається клієнт для придбання та обслуговування такого обладнання центру обробки даних, та врахувати це як експлуатаційні витрати. Стандартне визначення хмарних обчислень можна знайти в Національному інституті стандартів та технологій: Пітер М. Мелл та Тімоті Гранс. SP 800-145. NICE Визначення хмарних обчислень. Технічний звіт. NIST, Gaithersburg, MD, США. На рисунку 7.1 показані відмінності в керуванні хмарними моделями. NaaS включає такі сервіси, як SDP та SDN. IaaS підштовхує апаратні системи та сховище до хмари. PaaS включає інфраструктуру, але також керує операційною системою і часом виконання системи або контейнерами в хмарі. SaaS підштовхує всі сервіси, інфраструктуру та сервіси до хмарного провайдера.

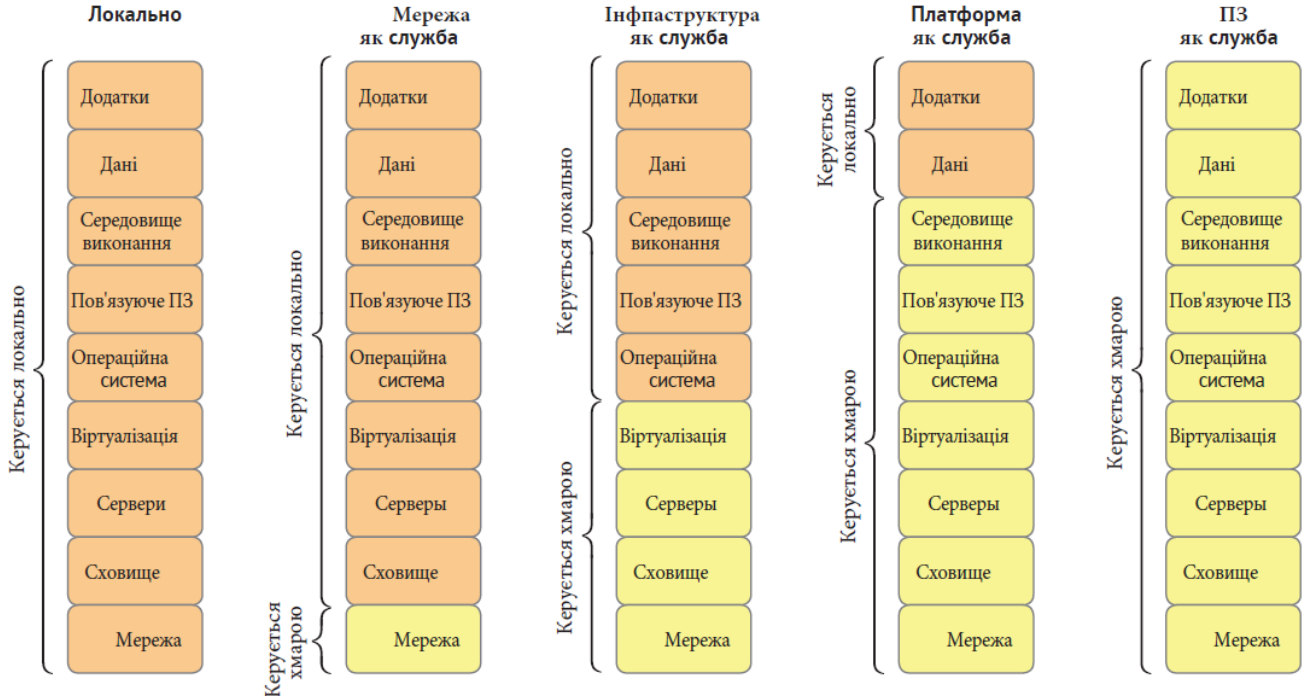


Рисунок 7.1 – Моделі хмарної архітектури. Власний об’єкт – це те, де управління всіма службами, інфраструктурою та сховищем здійснюється власником

Для NaaS характерні такі послуги, як мережева взаємодія, певне ПЗ (SDN) і програмно-визначені периметри (SDP). Ці продукти є керованими хмарами та організованими механізмами для забезпечення оверлейних мереж та безпеки підприємств Замість того, щоб створювати глобальну інфраструктуру та виділяти капітал для підтримки корпоративних комунікацій, під час створення віртуальної мережі може використовуватися хмарний підхід. Це дозволяє мережі оптимально масштабувати ресурси в бік збільшення або зменшення залежно від потреб, а нові мережеві якості можуть бути придбані та розгорнуті швидко.

SaaS є основою хмарних обчислень. У провайдера зазвичай є пропоновані програми або послуги, які пропонуються кінцевим користувачам з допомогою таких клієнтів, як мобільні пристрої, тонкі клієнти або фреймворки в інших хмарах. З погляду користувача, віртуальний SaaS фактично працює на клієнта користувача. Ця абстракція програмного забезпечення дозволила галузі досягти значного зростання в хмарний сервіс. Сервіси SaaS працюють для таких пристроїв, як Google Apps, Salesforce та Microsoft Office 365

PaaS використовує базове обладнання та програмні засоби нижнього рівня, що надаються хмарою. У такому випадку кінцевий користувач просто використовує апаратне забезпечення центру обробки даних, операційну систему, проміжне ПЗ та різні бази даних постачальника для розміщення своєї приватної програми або сервісів. Проміжне програмне забезпечення може складатися з систем баз даних. При побудові багатьох галузей промисловості було використано обладнання хмарних постачальників, наприклад для Swedbank, Trek Bicycles і Toshiba. Прикладами публічних постачальників PaaS є IBM Bluemix, Google App Engine та Microsoft Azure. Різниця між PaaS та IaaS полягає в тому, що ви отримуєте переваги масштабованості та OPEX з хмарною інфраструктурою, але у вас також є перевірене проміжне ПЗ та ОС від провайдера. Це такі системи, як Docker, де програмне забезпечення розгортається в контейнери. Якщо ваш додаток розгортається у межах обмежень наданої постачальником інфраструктури, ви можете очікувати на швидший вихід на ринок, оскільки більшість компонентів, ОС і проміжного програмного забезпечення гарантовано доступні.

IaaS була початковою концепцією хмарних послуг. У цієї моделі постачальник створює масштабовані апаратні служби в хмарі та надає модифікацію програмних фреймворків для створення віртуальних клієнтських машин. Це забезпечує максимальну гнучкість при розгортанні, але вимагає більших зусиль з боку клієнта.

7.2 Публічна, приватна та гібридна хмара

У хмарному середовищі існують три різні моделі топології хмар, які зазвичай використовуються: приватна хмара, хмара загального користування та гібридна хмара. Незалежно від моделі, фреймворки хмар повинні забезпечувати динамічну масштабованість, швидкість розробки та розгортання, а також поява в локальному місці незалежно від його близькості (рис. 7.2). Приватні хмари також мають на увазі керовані компоненти на запит. Сучасні корпоративні системи, як правило, використовують гібридну архітектуру для забезпечення безпеки критично важливих додатків та даних на місцевості та використовують публічну хмару для підключення, простоти та швидкості розгортання.

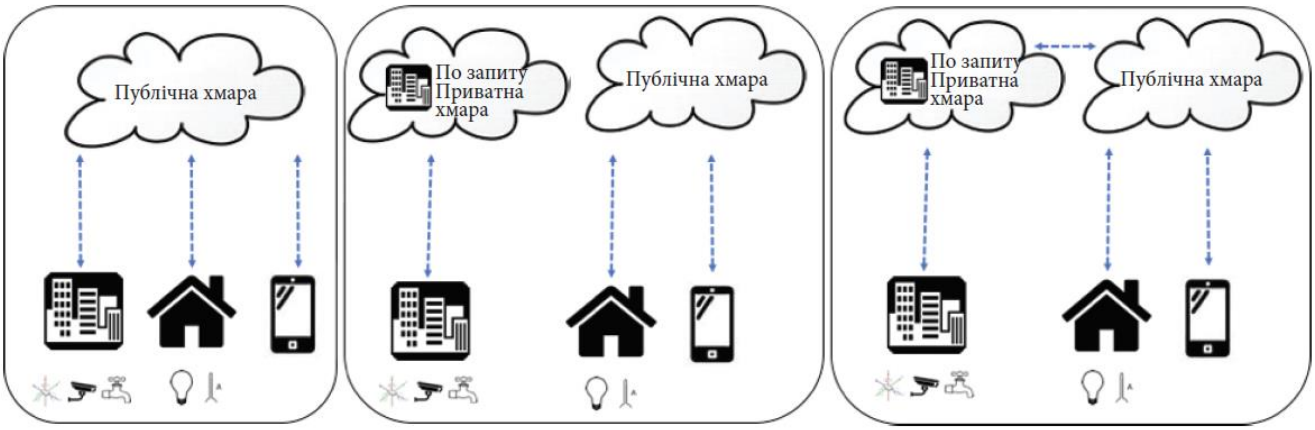


Рисунок 7.2 – Ліворуч: публічна хмара. Посередині: приватна. Праворуч: гібридна хмара

У приватній хмарі інфраструктуру надано одній організації або корпорації. Немає концепції спільного використання ресурсів чи об’єднання поза власною інфраструктурою власника. У приміщеннях спільне використання та розпорядження ресурсами є загальними. Приватна хмара існує з ряду причин, включаючи безпеку та перевіреність якості. Тобто для гарантії, що інформація обробляється виключно системами, керованими клієнтом. Однак, щоб вважатися хмарою, повинні існувати деякі аспекти хмарних сервісів, такі як віртуалізація та балансування навантаження. Приватна хмара може бути локальною або може бути спеціалізованою в обладнання, яке надається третьою стороною виключно для його використання.

Публічна хмара – протилежна ситуація. Тут інфраструктура надається на вимогу для багатьох клієнтів і додатків. Інфраструктура є набором ресурсів, які будь-яка людина може використовувати в будь-який час у рамках своїх угод про рівень обслуговування. Перевага тут явна шкала хмарних

центрів обробки даних дозволяє забезпечити безпрецедентну масштабованість для багатьох клієнтів, які обмежені лише тим, яку частину послуг вони хочуть придбати.

Гібридна архітектурна модель є поєднанням приватних і хмарних технологій. Такими комбінаціями можуть бути множинні публічні хмари, що використовуються одночасно або комбінація суспільної та приватної хмарної інфраструктури. Організації віддають перевагу гібридній моделі, якщо є дані, які потребують унікального підходу, а інтерфейс може використовувати хмара. Іншим варіантом використання є підтримка угоди з хмарними областями для компенсації умов, коли масштабованість краща, ніж у приватної корпорації в цілому. У цьому випадку публічна хмара буде використовуватися як балансувальник навантаження, доки набирання даних та їх використання не повернуться в обмежений простір приватної хмари. Цей варіант використання називається хмарним вибухом і відноситься до використання хмар як умовних ресурсів.

7.3 Хмарна архітектура OpenStack

OpenStack – це сервер Apache 2.0 з відкритим вихідним кодом, який використовується для створення хмарних платформ. Це IaaS розробляється спільнотою розробників з 2010 р. OpenStack Foundation керує програмним забезпеченням та підтримує понад 500 компаній, включаючи Intel, IBM, Red Hat та Ericsson. OpenStack як еталонна архітектура для постачальників хмарних обчислень, оскільки більшість компонентів і термінологія також використовуються в комерційних хмарах. OpenStack починався як спільний проект NASA та Rackspace у 2010 р. Архітектура має всі основні компоненти інших хмарних систем, включаючи обчислення та балансування навантаження; компоненти зберігання, включаючи резервне копіювання та відновлення; мережеві компоненти, інформаційні панелі, системи безпеки та ідентифікації, пакети даних та аналітики, інструменти розгортання, монітори, лічильники та програми. Це ті компоненти, які використовуватиме архітектор при виборі хмарного сервісу. З точки зору архітектури, OpenStack є змішаними шарами компонентів. Основну форму хмари OpenStack показано на рисунку 7.3. Кожен сервіс має певну функцію та унікальне ім’я (наприклад, Nova). Система працює, цілому, надаючи функціональні можливості хмарного класу масштабу корпорації, що масштабуються. Всі комунікації в компонентах OpenStack виконуються через протокол розширеної черги повідомлень (AMQP), зокрема, RabbitMQ або Qpid. Повідомлення можуть бути або неблокуючими, або блокуючими в залежності від того, як надіслано повідомлення. Повідомлення буде надіслано як об’єкт JSON у RabbitMQ, та одержувачі отримають свої повідомлення в одному сервісі. Це метод зв’язку (Remote Procedure Call – RPC) між основними підсистемами. Перевага хмарного середовища полягає в тому, що проблеми клієнта та сервери повністю незалежні один від одного, і це дозволяє серверам динамічно масштабуватися в бік збільшення чи зменшення. Повідомлення не передаються, а спрямовуються, що знижує трафік до мінімуму. AMQP – це стандартний протокол обміну повідомленнями, що використовується в просторі IoT.

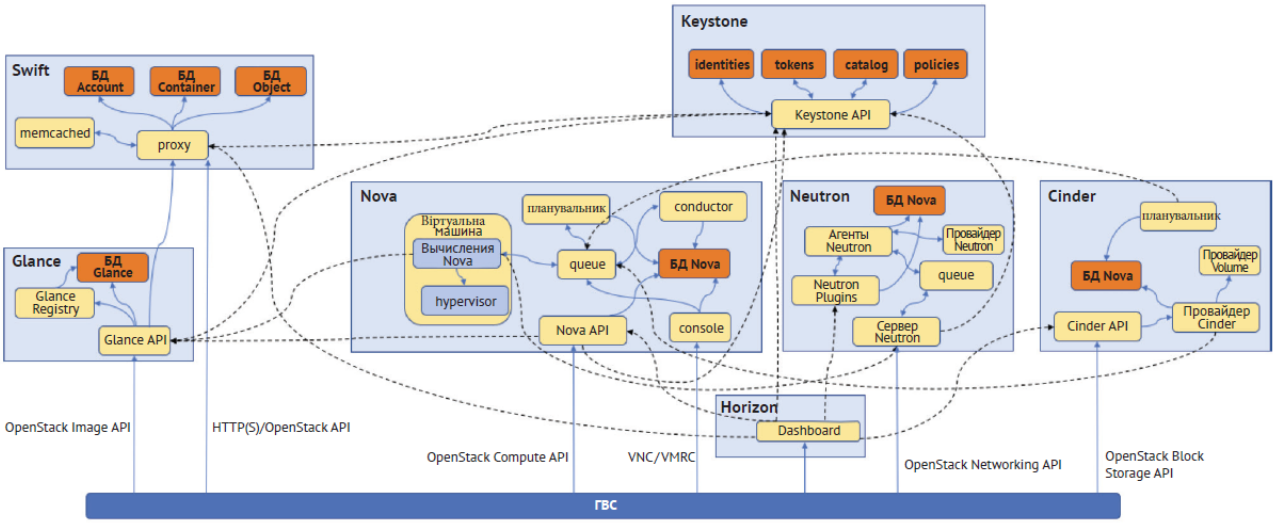


Рисунок 7.3 – Високорівнева архітектурна діаграма OpenStack

7.4 Keystone – управління ідентифікацією та обслуговуванням

Keystone – це служба керування ідентифікаторами хмари OpenStack. Менеджер ідентифікації встановлює облікові дані користувача та авторизацію для входу. Це, по суті, відправна точка або точка входу в хмару. Цей ресурс підтримуватиме центральний каталог користувачів та їх прав доступу. Це найвищий рівень безпеки, що забезпечує незалежність та безпеку середовищ користувача. Keystone може взаємодіяти з такими сервісами, як LDAP на корпоративному рівні. Keystone також підтримує базу даних токенів і надає тимчасові токени користувачам аналогічно тому, як Amazon Web Services (AWS) встановлює облікові дані. Реєстр служб використовується для запиту продуктів або послуг, доступних для користувача програмно.

Glance – це серцевина керування віртуальними машинами для OpenStack. Більшість хмарних сервісів забезпечить певний ступінь віртуалізації та матиме аналоговий ресурс, подібний до Glance. API служби зображень – це служба RESTful, що дозволяє клієнту розробляти шаблони VM, виявляти доступні віртуальні машини, клонувати зображення на інші сервери, реєструвати віртуальні машини та безперешкодно переміщати працюючі віртуальні машини на різні фізичні сервери без перерви в роботи. Glance викликає Swift (сховище об’єктів) для отримання або зберігання різних зображень. Glance підтримує різні стилі віртуальних образів:

– raw – неструктуровані зображення;
– vhd – VMWare, Xen, OracleVirtualBox;
– vmdk – загальний формат диска;
– vdi – зображення емулятора QEMU;
– iso – зображення на оптичному диску (CD-ROM);
– aki / ari / ami – зображення Amazon.

Swift надає резервну систему зберігання для центру обробки даних OpenStack. Swift дозволяє масштабувати кластери шляхом додавання нових серверів. Сховище об’єктів міститиме такі речі, як облікові записи та контейнери. Віртуальна машина користувача може зберігатися або кешуватися в Swift. Обчислювальний вузол Nova може викликати безпосередньо Swift та завантажувати зображення під час першого запуску.

Neutron – це керування мережею OpenStack та служба VLAN. Вся мережа є настроюваною і надає такі послуги, як:

– доменні служби імен;
– DHCP – протокол динамічної конфігурації хостів;
– функції шлюзу;
– управління VLAN;
– з’єднання на другому рівні моделі OSI;
– SDN;
– протоколи з покриттям та тунелюванням;
– VPN;
– NAT (SNAT та DNAT);
– системи виявлення вторгнень;
– балансування навантаження;
– брандмауери.

Cinder забезпечує OpenStack постійними службами зберігання блоків, необхідними для хмар. Він виступає в ролі сховища як служби для використання з базами даних, динамічними файловими системами та в інших випадках, де важливим є захист від витоків даних. Це важливо і для потокових сценаріїв IoT. Як і інші компоненти OpenStack, система зберігання сама по собі динамічна і масштабується в міру потреби. Архітектура побудована на принципах високої доступності та відкритих стандартів.

Функціональність Cinder включає:

– створення, видалення та прив’язку пристроїв зберігання до екземплярам Nova;
– сумісність з декількома сховищами (HP 3 PAR, EMC, IBM, Ceph, CloudByte, Scality);
– підтримку кількох інтерфейсів (Fibre Channel, NFS, Shared SAS, IBM GPFS, iSCSI);
– резервне копіювання та вилучення образів дисків;
– збереження зображень у–певні моменти часу;

– альтернативне сховище для зображень VM.

Horizon – це панель інструментів OpenStack. Це спрощений вигляд OpenStack для клієнта. Він забезпечує веб-представлення різних компонентів, які включають OpenStack (Nova, Cinder, Neutron та інші). Horizon являє собою зображення інтерфейсу хмарної системи в якості альтернативного засобу поверх API. Horizon розширюємо, тому третя сторона може додавати свої віджети чи інструменти до панелі інструментів. Можна додати новий компонент білінгу, і потім для клієнтів може бути створений відповідний елемент панелі Horizon. Більшість систем IoT, які використовують хмарні обчислення, матимуть подібну форму аналогічними функціями.

Heat може запускати кілька складових хмарних програм і керувати хмарною інфраструктурою на основі шаблонів в екземплярі OpenStack. Heat інтегрується з телеметрією для автоматичного налаштування системи відповідно до навантаження. Шаблони в Heat намагаються відповідати форматам AWS CloudFormation, а відносини між ресурсами можуть бути вказані аналогічним чином (наприклад, цей том підключений до даного серверу).

OpenStack надає додатковий сервіс під назвою Ceilometer, який може використовуватися для збору даних телеметрії та обліку ресурсів, що використовуються кожною службою. Вимірювання використовується для збору інформації про використання та перетворення його в рахунки клієнта. Ceilometer також надає інструменти оцінки та виставлення рахунків. Значення виставленої вартості конвертується в еквівалентну валюту, а білінг використовується для початку процесу оплати. Ceilometer контролює та вимірює різні події, такі як запуск служби, додавання тому та зупинка екземпляра. Метрики збираються з використання ЦПУ, кількості ядер, використання пам’яті та переміщення даних. Все це збирається і зберігається у базі даних MongoDB.

Література: [7, 8, 9].

ТЕМА 8
ІНДУСТРІАЛЬНИЙ ІНТЕРНЕТ РЕЧЕЙ (ІІОТ)

8.1 Загальні відомості про Індустріальний Інтернет речей (ІІоТ)

ІІоТ – це індустріальний Інтернет речей. ІІоТ підвищує ефективність виробництва завдяки застосуванню IoT в промисловості. Він дозволяє підвищити свою операційну продуктивність і прибутковість з підключеним заводським прискорювачем рішень. Підключайте і контролюйте промислове обладнання та пристрої в хмарі, включаючи ваші машини, вже працюють на заводі. Проаналізуйте свої дані IoT, щоб отримати інформацію, яка допоможе вам підвищити продуктивність всього заводу.

Скоротити трудомісткий процес доступу до машин заводського рівня дозволяє управління пристроями Azure IoT OPC UA (OPC Twin) та зосередити свій час на створення рішень ІІоТ. Оптимізуйте управління сертифікатами та інтеграцію промислових активів за допомогою управління сертифікатами Azure IoT OPC UA (OPC Vault) і відчуйте впевненість в тому, що з'єднання активів захищено. Ці мікросервіси надають REST-подібний API поверх компонентів IoT Azure Industrial. API сервісу дає вам контроль над функціональністю прикордонного модуля (рис. 8.2).

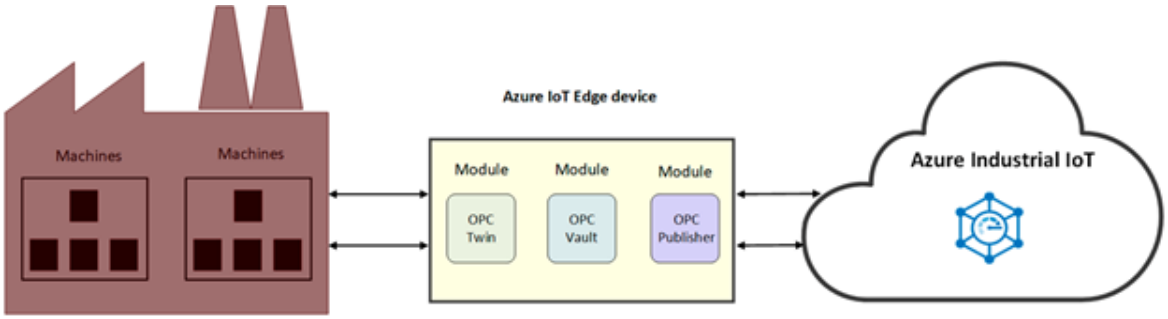


Рисунок 8.1 – Огляд промислового IoT

Connected Factory – це реалізація еталонної архітектури IoT корпорації Microsoft Azure, яку можна налаштувати відповідно до конкретних вимог бізнесу. Повний код рішення з відкритим вихідним кодом і доступний в GitHub-репозиторії прискорювачів рішень Connected Factory. Ви можете використовувати його в якості відправної точки для комерційного продукту і розгорнути готове рішення у своїй підписці Azure за лічені хвилини.

Управління пристроями Azure IoT OPC UA, також відоме як OPC Twin, являє собою компонент ІІоТ, який автоматизує виявлення і реєстрацію пристроїв і пропонує віддалене управління промисловими пристроями через API REST. OPC Twin використовує Azure IoT Edge і IoT Hub для з'єднання хмари і фабричної мережі. OPC Twin дозволяє розробникам ІІоТ зосередитися на створенні додатків ІІоТ, не турбуючись про те, як забезпечити безпечний доступ до локальних комп'ютерів.

Azure IoT OPC UA Управління сертифікатами або OPC Vault – це реалізація OPC UA Global Discovery Server (GDS), яка може налаштовувати, реєструвати і управляти життєвим циклом сертифікатів для сервера OPC UA і клієнтських додатків в хмарі. OPC Vault спрощує впровадження і обслуговування безпечного підключення активів в промисловому просторі. Автоматизуючи управління сертифікатами, OPC Vault звільняє заводських операторів від ручних і складних процесів, пов'язаних з підключенням і управлінням сертифікатами.

Azure IoT Edge переносить хмарну аналітику і настроюється бізнес-логіку на пристрої, щоб ваша організація могла зосередитися на бізнес-аналітиці, а не на управлінні даними. Конфігурація програмного забезпечення IoT, розгортання його на пристроях через стандартні контейнери і відстеження всього цього з хмари.

Примітка. Azure IoT Edge доступний на безкоштовному стандартному рівні IoT Hub. Безкоштовний рівень призначений тільки для тестування і оцінки.

Аналітика підвищує цінність бізнесу в IoT-рішеннях, але не вся аналітика повинна бути в хмарі. Якщо ви хочете, щоб пристрій реагував на надзвичайні ситуації якомога швидше, ви можете виконати виявлення аномалій на самому пристрої. Аналогічним чином, якщо ви хочете скоротити витрати на пропускну здатність і уникнути передачі терабайтів необроблених даних, ви можете

виконати очистку та його узагальнення даних локально. Потім відправте інформацію в хмару для аналізу.

Azure IoT Edge складається з трьох компонентів:

Модулі IoT Edge – це контейнери, в яких виконуються служби Azure, сторонні служби або ваш власний код. Модулі розгортаються на пристроях IoT Edge і виконуються локально на цих пристроях.

Середовище виконання IoT Edge запускається на кожному пристрої IoT Edge і управляє модулями, розгорнутими на кожному пристрої.

Хмарний інтерфейс дозволяє віддалено контролювати і управляти пристроями IoT Edge.

Модулі IoT Edge – це одиниці виконання, реалізовані у вигляді Docker-сумісних контейнерів, які керують вашою бізнес-логікою на межі (кордоні). Кілька модулів можна налаштувати для зв'язку один з одним, створюючи конвеєр обробки даних. Ви можете розробляти власні модулі або упакувати певні служби Azure в модулі, які забезпечують розуміння в автономному режимі і на межі.

Штучний інтелект на межі. Azure IoT Edge дозволяє розгортати складну обробку подій, машинне навчання, розпізнавання зображень і інші високорівневі II без написання їх власними силами. Служби Azure, такі як функції Azure, Azure Stream Analytics і машинне навчання Azure, можна запускати локально через Azure IoT Edge, але ви не обмежені службами Azure. Будь-хто може створювати AI-модулі і робити їх доступними для спільноти для використання через Azure Marketplace.

Принеси свій власний код. Якщо ви хочете розгорнути власний код на своїх пристроях, Azure IoT Edge також підтримує це. Azure IoT Edge підтримує ту ж модель програмування, що і інші служби Azure IoT. Один і той же код можна запустити на пристрої або в хмарі. Azure IoT Edge підтримує як Linux, так і Windows, так що ви можете кодувати на потрібну платформу. Він підтримує Java, .NET Core 2.0, Node.js, C і Python, тому ваші розробники можуть кодувати мовою, який вони вже знають, і використовувати існуючу бізнес-логіку.

Середовище виконання Azure IoT Edge включає настроюється і хмарну логіку на пристроях IoT Edge. Він знаходиться на пристрої IoT Edge і виконує операції управління та зв'язку. Середовище виконання виконує кілька функцій:

Встановіть і обновіть робочі навантаження на пристрої.

Підтримуйте стандарти безпеки Azure IoT Edge на пристрої.

Переконайтеся, що модулі IoT Edge завжди працюють.

Повідомити про працездатність модуля в хмару для віддаленого моніторингу.

Управління зв'язком між нижче стоячими кінцевими пристроями і пристроєм IoT Edge, між модулями на пристрої IoT Edge і між пристроєм IoT Edge і хмарою (рис 8.2).

Рисунок 8.2 – Відправка аналітичних даних і звітів з середовища виконання IoT Edge в Центр Інтернету речей

Як ви використовуєте пристрій Azure IoT Edge, залежить від вас. Середовище виконання часто використовується для розгортання AI на шлюзах, які збирають і обробляють дані з інших локальних пристроїв, проте ця модель розгортання є лише одним з варіантів. Кінцеві пристрої також можуть бути пристроями Azure IoT Edge незалежно від того, підключені вони до шлюзу або безпосередньо до хмари.

Середовище виконання Azure IoT Edge працює на великому наборі пристроїв IoT, що дозволяє використовувати середовище виконання різними способами. Воно підтримує операційні

системи як Linux, так і Windows, а також надає докладну інформацію про обладнання. Використовуйте пристрій менше, ніж Raspberry Pi3, якщо ви не обробляєте багато даних, або використовуйте промисловий сервер для виконання ресурсномістких робочих навантажень.

Управління життєвим циклом програмного забезпечення для корпоративних пристроїв є складним. Управління життєвим циклом програмного забезпечення для мільйонів різнорідних IoT-пристроїв ще складніше. Робочі навантаження повинні бути створені і сконфігуровані для конкретного типу пристроїв, розгорнуті в масштабі на мільйонах пристроїв у вашому рішенні і відслідковані для виявлення будь-яких непрацюючих пристроїв. Ці дії не можуть бути виконані для кожного пристрою і повинні бути виконані в масштабі.

Azure IoT Edge легко інтегрується з прискорювачами рішень Azure IoT, забезпечуючи єдину площину управління для потреб вашого рішення. Хмарні сервіси дозволяють:

– створити і налаштувати робоче навантаження для запуску на пристрої певного типу;

– надіслати навантаження на набір пристроїв.

Моніторинг робочих навантажень на пристроях в польових умовах (рис. 8.3).

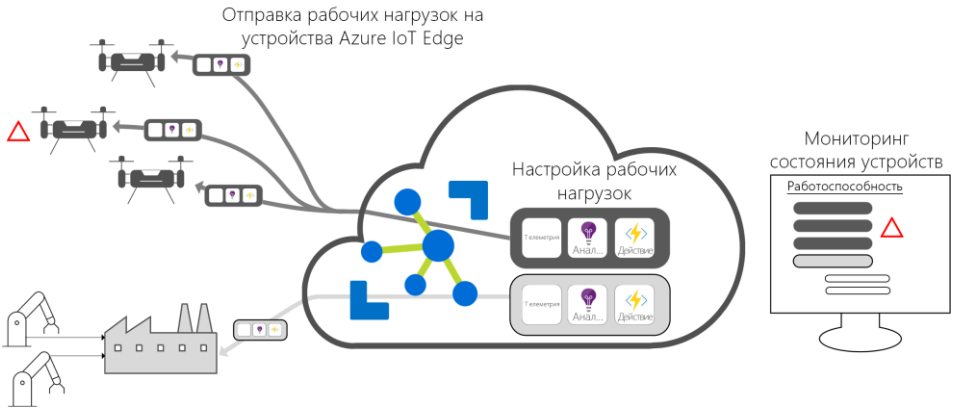


Рисунок 8.3 – Дані телеметрії, аналітики і дії пристроїв координуються з хмарою

8.2 Управління пристроями Azure IoT Open Platform Communications (OPC)

Управління пристроями Azure IoT OPC UA, також відоме як OPC Twin, складається з мікросервісів, які використовують Azure IoT Edge і IoT Hub для з'єднання хмари і фабричної мережі. OPC Twin забезпечує виявлення, реєстрацію та віддалене управління промисловими пристроями через API REST. OPC Twin не вимагає SDK OPC Unified Architecture (OPC UA), не залежить від мови програмування і може бути включений в безсерверний робочий процес.

OPC Twin дозволяє заводським операторам сканувати заводську мережу, щоб сервери OPC UA могли бути виявлені і зареєстровані. В якості альтернативи, фабричні оператори можуть також вручну реєструвати пристрої OPC UA, використовуючи відому URL-адресу виявлення. Наприклад, щоб підключитися до всіх пристроїв OPC UA після того, як на заводському цеху був встановлений шлюз IoT Edge з модулем OPC Twin, оператор фабрики може віддалено запустити сканування мережі та візуально переглянути всі сервери OPC UA.

OPC Twin дозволяє операторам фабрики реагувати на події і перенастроювати свої фабричні машини з хмари автоматично або вручну на льоту. OPC Twin надає API-інтерфейси REST для виклику служб на сервері OPC UA, перегляду його адресного простору, а також для читання / запису змінних і виконання методів, наприклад, бойлер використовує KPI температури для управління виробничою лінією. Датчик температури публікує зміну даних за допомогою OPC Publisher. Заводський оператор отримує попередження про те, що температура досягла порогового значення. Виробнича лінія автоматично охолоджується через OPC Twin. Оператор фабрики повідомляється про охолодження.

OPC Twin використовує аутентифікацію і аудит на основі Azure Active Directory (AAD) від початку до кінця. Наприклад, OPC Twin дозволяє побудувати додаток поверх OPC Twin, щоб визначити, що оператор виконав на машині. На стороні машини - через аудит OPC UA. На хмарній стороні – зберігання незмінного журналу аудиту клієнта і аутентифікація AAD в REST API.

Простий досвід розробника

OPC Twin можна використовувати з додатками, написаними на будь-якій мові програмування через API REST. Оскільки розробники інтегрують клієнта OPC UA в рішення, знання SDC OPC UA не потрібно. OPC Twin може легко інтегруватися в архітектуру без сервера і без сервера. Наприклад, веб-розробник з повним стеком, який розробляє додаток для панелі моніторингу аварійних сигналів і подій, може написати логіку для відповіді на події в JavaScript або TypeScript з використанням OPC Twin без знання C, C # або повної реалізації стека OPC UA.

1. Оператор включає мережеве сканування на модулі або виконує одноразове виявлення з використанням URL-адреси виявлення. Виявлені кінцеві точки і інформація про програму відправляються за допомогою телеметрії агенту для обробки. Агент підключення пристрою OPC UA обробляє події виявлення сервера OPC UA, відправлені модулем OPC Twin IoT Edge в режимі виявлення або сканування. Події виявлення призводять до реєстрації програми та оновлення в реєстрі пристрою OPC UA (рис. 8.4).

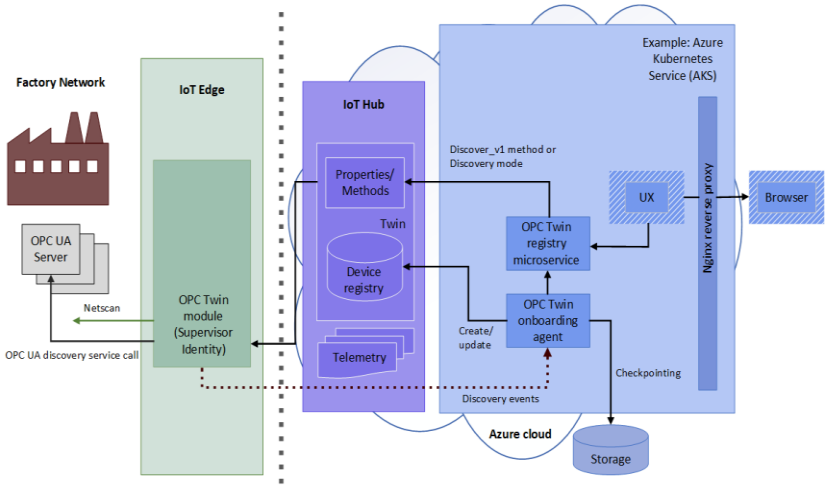


Рисунок 8.4 – Робота OPC Twin (1)

2. Оператор перевіряє сертифікат виявленої кінцевої точки і активує зареєстрованого близнюка кінцевої точки для доступу.

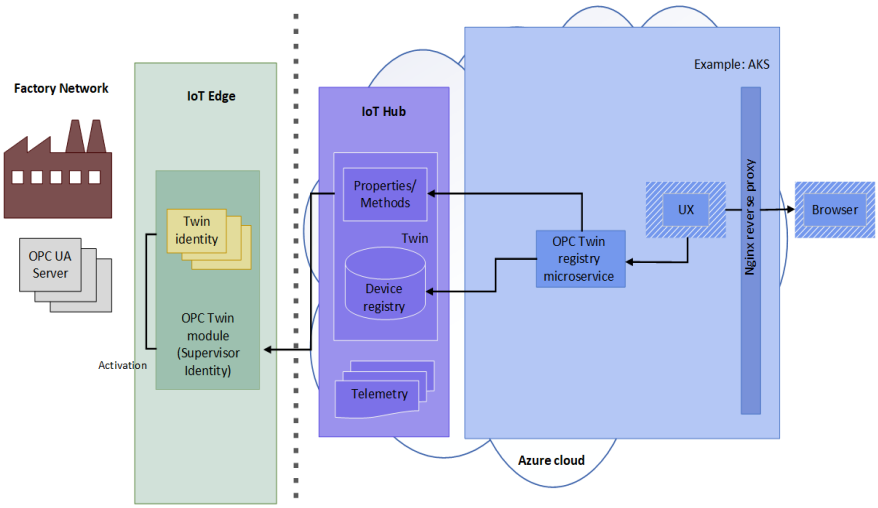


Рисунок 8.5 – Робота OPC Twin (2)

3. Після активації оператор може використовувати REST API служби Twin для перегляду або перевірки інформаційної моделі сервера, читання/запису змінних об’єкта і виклику методів. Користувач використовує спрощений OPC UA API, повністю виражений в HTTP і JSON.

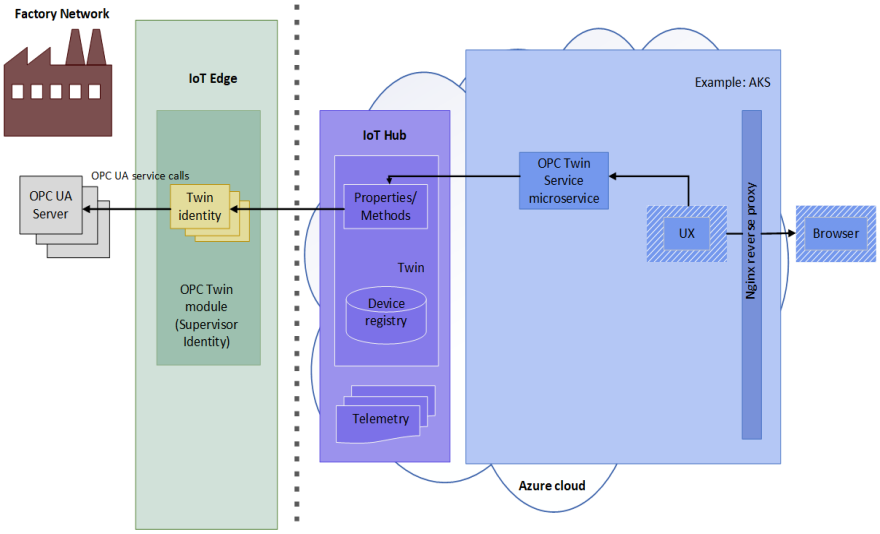


Рисунок 8.6 – Робота OPC Twin (3)

4. Інтерфейс REST з двома службами також можна використовувати для створення відслідковуються елементів і підписок в OPC Publisher. OPC Publisher дозволяє відправляти телеметрію з серверних систем OPC UA в IoT Hub.

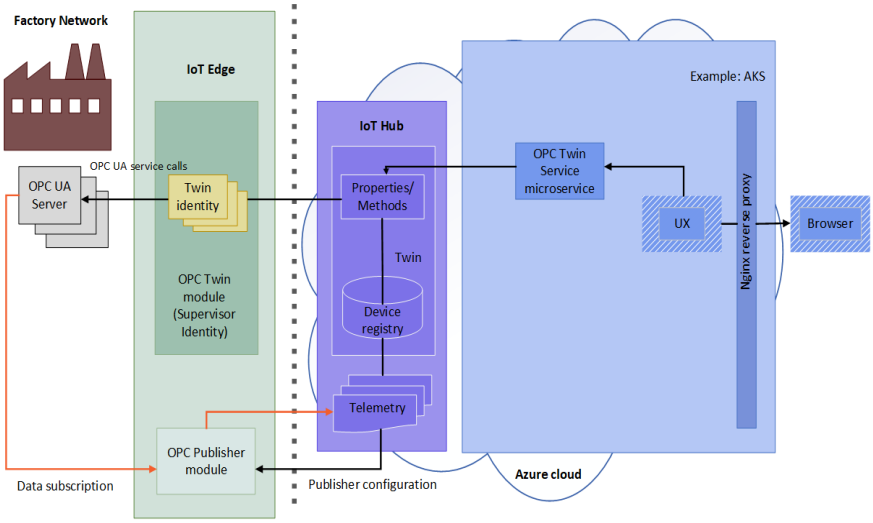


Рисунок 8.7 – Робота OPC Twin (4)

8.3 Управління сертифікатами UA Azure IoT Open Platform Communications (OPC)

Управління сертифікатами Azure IoT OPC UA (сховище OPC), також відоме як **OPC Vault**, являє собою мікросервіс, який може налаштовувати, реєструвати і управляти життєвим циклом сертифікатів для сервера OPC UA і клієнтських додатків в хмарі.

Наприклад, виробнича компанія повинна підключити свій сервер OPC UA до свого нового клієнтського додатку. Коли виробник робить початковий доступ до серверного комп'ютера, на серверному додатку OPC UA негайно відображається повідомлення про помилку, яке вказує, що клієнтське додаток не захищений. Цей механізм вбудований в серверний комп'ютер OPC UA для запобігання несанкціонованому доступу до додатків, що запобігає зловмисний злом в цеху.

Спеціаліст з безпеки використовує мікросервіс OPC Vault, щоб легко підключити сервер OPC UA до будь-якого клієнтського додатку, оскільки в OPC Vault є всі функції для управління реєстром сертифікатів, зберіганням і управлінням життєвим циклом. Тепер сервер OPC UA надійно підключений, він може зв'язуватися з недавно створеним клієнтським додатком.

Наступна діаграма ілюструє повну архітектуру OPC Vault (рис. 8.8).

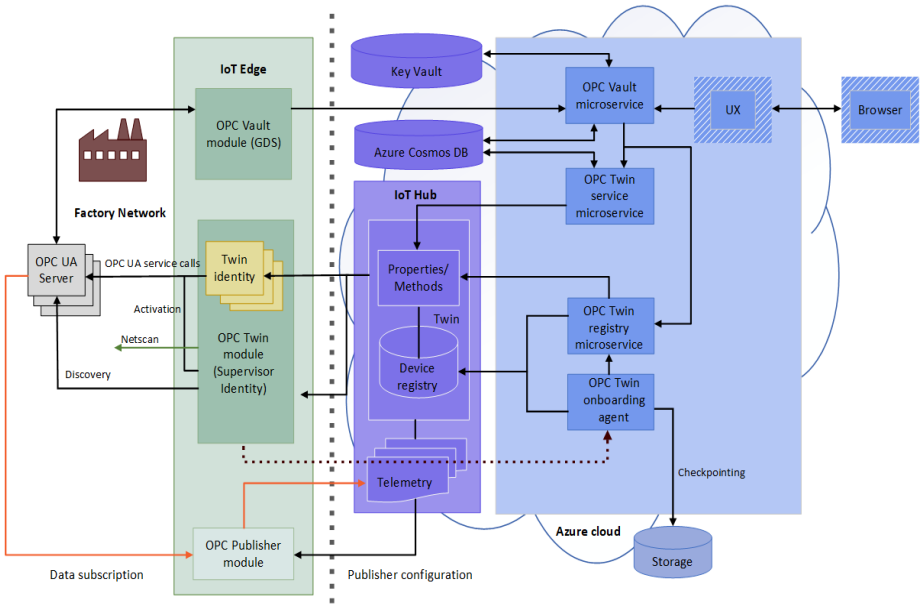


Рисунок 8.8 – Архітектура OPC Vault

8.4 Загальні відомості про акселератор рішень IoT для підключеної фабрики

Підключена фабрика реалізує в одному рішенні з відкритим вихідним кодом еталонну архітектуру Azure для промислового Інтернету речей, розроблену корпорацією Майкрософт. Ви можете використовувати її в якості відправної точки для комерційних продуктів. Готові версії рішення на основі підключеної фабрики ви можете розгорнути в підписці Azure з акселератора рішень Azure IoT (рис. 8.9).

Рисунок 8.9 – Панель моніторингу для вирішення підключеної фабрики

Код акселератора рішень для підключеної фабрики доступний в репозиторії GitHub.

Підключена фабрика реалізує наступні можливості:

– підключення до промислових ресурсів, що підтримує інтерфейс OPC UA;

– використання імітованих виробничих ліній (під управлінням серверів OPC UA в контейнерах Docker) для отримання від них даних телеметрії в режимі реального часу;

– перегляд інформаційної моделі OPC UA для серверів OPC UA через панель моніторингу для хмари;

– налаштування ресурсів OPC UA за допомогою хмарної панелі моніторингу (виклик методів, читання і запису даних);

– публікацію і скасування публікації даних телеметрії, одержуваних від активів OPC UA через хмарну панель моніторингу;

– попередній перегляд даних телеметрії прямо на хмарної панелі моніторингу;

– перегляд тенденцій для даних телеметрії і створення кореляцій за допомогою панелі моніторингу оглядача служби «Аналітика часових рядів»;

– перегляд обчислюється загальної ефективності роботи обладнання (OEE) і ключових показників ефективності (KPI) на хмарної панелі моніторингу;

– перегляд ієрархії промислових активів в топології дерева і на інтерактивній карті;

– перегляд, підтвердження і закриття сповіщень через хмарну панель моніторингу.

Служба «Аналітика часових рядів» розроблена для того, щоб зберігати, візуалізувати і запитувати великі обсяги даних часових рядів. Підключена фабрика використовує цю службу.

Підключена фабрика інтегрується з цією службою, дозволяючи виконувати глибокий аналіз даних пристрою в режимі реального часу.

Налаштування правил для попереджень на основі порогових значень.

Налаштування дозволів безпеки для користувачів за допомогою управління доступом на основі ролей (RBAC).

Наскрізне шифрування реалізується за допомогою аутентифікації OPC UA (з використанням сертифікатів X.509) та маркерів безпеки.

Можливості налаштування

Ви можете налаштувати рішення відповідно до потреб конкретної організації.

Повний вихідний код рішення ви знайдете на сайті GitHub призначеного для виконання таких завдань:

– підключається до віртуальних галузевим пристроїв під управлінням серверів OPC UA на виробничих лініях віртуальної фабрики і до фізичних пристроїв сервера OPC UA;

– показує оперативні ключові показники ефективності і загальну ефективність обладнання цих пристроїв і виробничих ліній;

– демонструє, як можна використовувати хмарний додаток для взаємодії з серверними системами OPC UA;

– дозволяє підключати власні пристрої під управлінням сервера OPC UA;

– дозволяє переглядати і змінювати дані сервера OPC UA;

– інтегрується зі службою Azure Time Series Insights (TSI) для надання користувальницьких подань даних з серверів OPC UA.

Його можна використовувати в якості відправної точки для власної реалізації та налаштувати відповідно до потреб конкретної організації. Знання порядку обміну даними необхідні для того щоб:

– усунути проблеми, що виникли з рішенням;

– спланувати настройку рішення відповідно до певних вимог;

– спроектувати власне рішення IoT, що використовує служби Azure.

На наступній схемі показані логічні компоненти акселератора рішень (рис. 8.10).

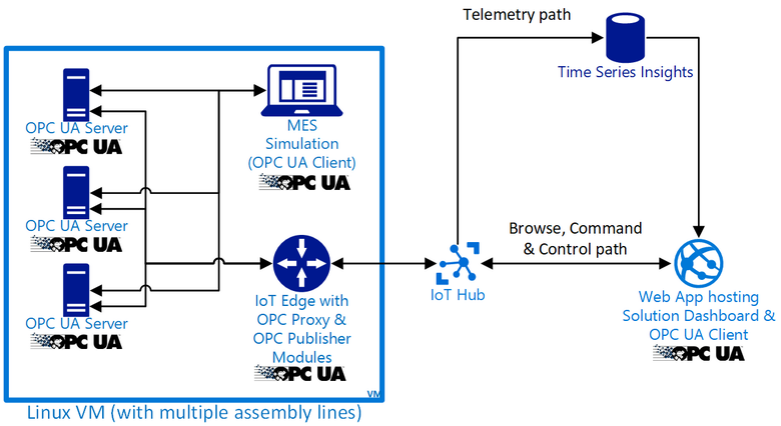


Рисунок 8.10 – Логічна архітектура підключеної фабрики

Рішення використовує специфікацію публікації і підписки OPC UA для відправки даних телеметрії OPC UA в Центр Інтернету речей в форматі JSON. У рішенні для цього використовується модуль IoT Edge видавця OPC.

Рішення також містить клієнт OPC UA, інтегрований в веб-додаток, який може встановлювати з'єднання з локальними серверами OPC UA. Клієнт використовує зворотний проксі-сервер і за допомогою Центру Інтернету речей створює підключення без необхідності відкривати порти в локальному брандмауері. Ця модель зв'язку називається взаємодією з підтримкою служби. У рішенні для цього використовується модуль IoT Edge проксі OPC.

Віртуальні станції і системи управління виробничими процесами (MES) утворюють виробничу лінію фабрики. Віртуальні пристрої і модуль видавця OPC засновані на стандарті OPC UA .NET, опублікованому OPC Foundation.

Проксі-сервер і видавець OPC реалізовані як модулі на основі Edge Інтернету речей Azure. До кожної віртуальної виробничої лінії підключений шлюз.

Всі віртуальні компоненти працюють в контейнерах Docker, розміщених на віртуальній машині Azure під управлінням Linux. За замовчуванням моделювання налаштоване для роботи восьми віртуальних виробничих ліній.

Виробнича лінія виготовляє деталі. Вона складається з кількох станцій: збірка, тестування і упаковка.

Імітація оновлює дані, які відображаються на вузлах OPC UA. Робота всіх станцій віртуальної виробничої лінії координується за допомогою MES через OPC UA.

Система MES відстежує кожну станцію на виробничій лінії через OPC UA для виявлення змін в стані станції. Вона викликає методи OPC UA для управління станціями і передає продукти від однієї станції до іншої до завершення циклу.

Модуль видавця OPC підключається до серверів OPC UA на станції і підписується на вузли OPC, які повинні бути опубліковані. Цей модуль виконує наступне:

– перетворює дані вузла в формат JSON;

– шифрує отриманий код JSON;

– відправляє код JSON в Центр Інтернету речей у вигляді повідомлень публікації і підписки OPC UA.

Модулю видавця OPC потрібно тільки вихідний порт HTTPS (443), і він може працювати з існуючою інфраструктурою підприємства.

Модуль проксі OPC UA шлюзу надає доступ через тунелі двійковим командам і керуючим повідомленнями OPC UA. Для його роботи потрібно тільки вихідний порт HTTPS (443). Він може працювати з існуючою інфраструктурою підприємства, в тому числі з веб-проксі.

Цей модуль використовує методи пристрою Центру Інтернету речей для передачі пакетованих даних TCP / IP на рівні додатку, щоб за допомогою SSL / TLS забезпечити довіру кінцевої точки, шифрування і цілісність даних.

Двійковий протокол OPC UA, який ретранслюється через сам проксі, використовує перевірку автентичності та шифрування UA.

Модуль видавця OPC шлюзу (Gateway OPC Publisher) підписується на вузли сервера OPC UA для виявлення змін в значеннях даних. Якщо буде виявлено зміну даних на одному з вузлів, цей модуль відправляє повідомлення в Центр Інтернету речей Azure.

Центр Інтернету речей передає джерело події в Azure TSI. TSI зберігає дані протягом 30 днів в залежності від міток часу, вкладених в повідомлення. Ці дані включають:

– OPC UA ApplicationUri;

– OPC UA NodeId;

– значення вузла;

– мітка часу джерела;

– OPC UA DisplayName.

Зараз TSI не дозволяє клієнтам налаштовувати тривалість зберігання даних.

TSI відправляє запит до даних вузла з використанням SearchSpan на основі часу і групує результати за значеннями OPC UA ApplicationUri, OPC UA NodeId або OPC UA DisplayName.

Щоб отримати дані для датчиків загальної ефективності обладнання і ключових показників ефективності, а також діаграми часових рядів, це рішення виконує статистичну обробку за кількістю подій, загальній сумі (Sum), середнього (Avg), мінімального (Min) або максимальному (Max) значенням.

Часові ряди створюються за допомогою іншого процесу. Рішення обчислює значення загальної ефективності обладнання і ключових показників ефективності на основі базових даних станції і підсумовує результати по виробничим лініях, фабрикам і підприємству в цілому.

Крім того, часові ряди для топології загальної ефективності обладнання і ключових показників ефективності обчислюються в додатку, коли готовий відображається часовий діапазон. Наприклад, перегляд за день оновлюється щогодини.

Подання часового ряду даних вузла витягується безпосередньо з TSI за допомогою обчислення часового діапазону.

Центр Інтернету речей приймає дані, відправлені з модуля видавця OPC в хмару, і надає до них доступ службі Azure TSI.

Центр Інтернету речей також виконує в рішенні наступні функції.

Підтримує реєстр посвідчень, в якому зберігаються ідентифікатори всіх модулів видавця OPC та модулів проксі OPC.

Використовується в якості каналу транспортування для двостороннього обміну даними модуля проксі OPC.

Рішення використовує сховище BLOB-об’єктів Azure як дискового сховища віртуальної машини і як сховище даних розгортання.

Веб-додаток, розгорнуте як частина акселератора рішень, складається з вбудованого клієнта OPC UA, механізму обробки повідомлень і візуалізації даних телеметрії (рис. 8.11).

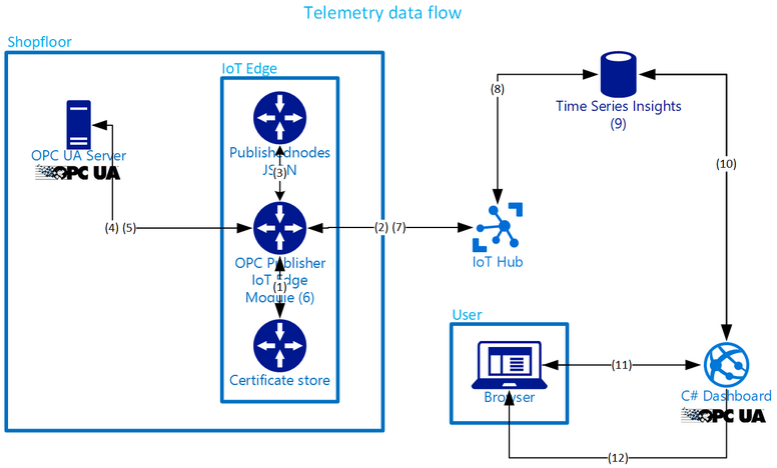


Рисунок 8.11 – Передача даних телеметрії

Етапи передачі.

1. Видавець OPC (OPC Publisher) зчитує необхідні сертифікати X509 OPC UA та облікові дані безпеки Центру Інтернету речей з локального сховища сертифікатів. При необхідності видавець OPC створює і зберігає відсутні сертифікати та облікові дані в сховище сертифікатів.

2. Видавець OPC (OPC Publisher) реєструється в Концентраторі Інтернету речей (IoT Hub). Використовується налаштований протокол. Ви можете використовувати підтримуваний протокол пакета SDK клієнта Центру Інтернету речей. За замовчуванням використовується MQTT. Обмін даними по протоколу захищений протоколом TLS.

3. Видавець OPC (OPC Publisher) зчитує файл конфігурації. 4. Видавець OPC (OPC Publisher) створює сеанс OPC з кожним налаштованим сервером OPC UA. Використовується TCP-з’єднання. Відбувається перевірка справжності між видавцем OPC і сервером OPC UA з використанням сертифіката X509. Подальший трафік OPC UA шифрується з використанням налаштованого механізму шифрування OPC UA.

4. Видавець OPC створює підписки OPC в сеансі OPC для кожного налаштованого інтервалу публікації. Створюються відслідковують елементи OPC для вузлів OPC для публікації в підписці OPC.

5. При зміні значення відслідковуємого вузла OPC сервер OPC UA відправляє поновлення видавцеві OPC.

6. Видавець OPC перекодовує нове значення. Кілька змін поміщаються в пакети, якщо включена пакетна обробка. Створюється повідомлення IoT Hub.

7. Видавець OPC відправляє повідомлення в IoT Hub. Використовується налаштований протокол. Обмін даними захищений протоколом TLS.

8. Аналіз часових рядів Time Series Insights (TSI) зчитує повідомлення з IoT Hub. Використовується AMQP через TCP / TLS. Цей крок є внутрішнім для центру обробки даних.

9. Дані в стані покою в TSI.

10. Connected Factory WebApp в Azure AppService (Підключена фабрика веб-додатки в Службі додатків Azure) запитує необхідні дані в TSI. Використовується безпечний обмін даними через TCP / TLS. Цей крок є внутрішнім для центру обробки даних.

11. Веб-браузер підключається до веб-додатку підключеної фабрики. Відображається панель моніторингу підключеної фабрики. Виконується підключення по протоколу HTTPS. Для доступу до програми підключеної фабрики потрібно виконати аутентифікацію користувача через Azure Active Directory. Всі виклики додатки підключеної фабрики даних в веб-API захищені за допомогою токенів для захисту від підробки.

12. При оновленні веб-додаток підключеної фабрики даних відправляє оновлені дані в веб-браузер. Використовується протокол SignalR. Забезпечується захист через TCP / TLS (рис. 8.12).

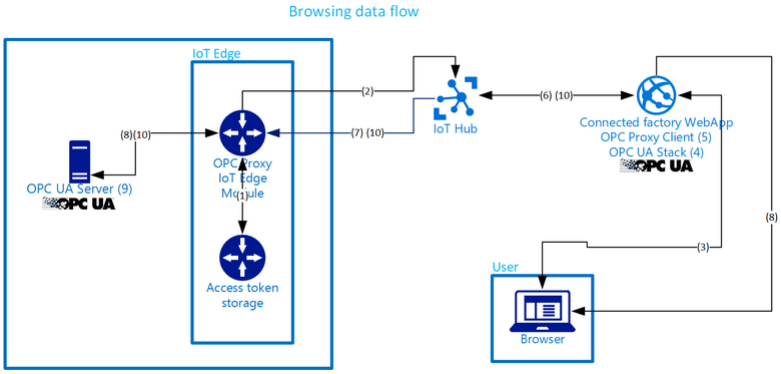


Рисунок 8.12 – Передача даних перегляду

Етапи передачі:

1. Запускається OPC Proxy (проксі-сервер OPC) (компонент сервера). Ключі загального доступу зчитуються з локального сховища. При необхідності відсутні ключі доступу зберігаються в сховищі.

2. Проксі-сервер OPC (серверний компонент) реєструється в IoT Hub. Всі відомі пристрої зчитуються з IoT Hub. Використовується MQTT через TLS через Socket або безпечний Websocket.

3. Веб-браузер підключається до веб-додатку підключеної фабрики і відображає її панель моніторингу. Використовується протокол HTTPS. Користувач вибирає сервер OPC UA для підключення.

4. Connected Factory WebApp (Веб-додаток підключеної фабрики) встановлює сеанс OPC UA на обраному сервері OPC UA. Використовується стек OPC UA.

5. Проксі-сервер OPC отримує з стека OPC UA запит на підключення сокета TCP до сервера OPC UA. Він отримує корисні дані TCP і використовує їх без змін. Цей крок виконується в веб-додатку підключеної фабрики.

6. OPC Proxy (Проксі-сервер OPC) (клієнтський компонент) знаходить пристрій проксі-сервера OPC (серверний компонент) в реєстрі пристроїв IoT Hub. Потім в IoT Hub викликається метод пристрою проксі-сервера OPC (серверний компонент). Для пошуку проксі-сервера OPC використовується протокол HTTPS через TCP / TLS. HTTPS через TCP/TLS використовується також для підключення сокета TCP до сервера OPC UA. Цей крок є внутрішнім для центру обробки даних.

7. IoT Hub викликає метод пристрою проксі-сервера OPC (серверний компонент). Для підключення сокета TCP до сервера OPC UA використовується підключення MQTT через TLS через Socket або Secure Websocket.

8. Проксі-сервер OPC (серверний компонент) відправляє корисні дані TCP в виробничу мережу.

9. Сервер OPC UA обробляє корисні дані і відправляє відповідь.

10. Сокет отримує відповідь проксі-сервера OPC (серверний компонент). Проксі-сервер OPC відправляє дані як повертається значення методу пристрою в Центр Інтернету речей і на проксі-сервер OPC (клієнтський компонент). Ці дані доставляються в стек OPC UA в додатку підключеної фабрики.

11. Веб-додаток підключеної фабрики повертає інтерфейс браузера OPC з відомостями OPC UA, отриманими з сервера OPC UA, в веб-браузер для відображення. Коли користувач переглядає простір адрес OPC і застосовує функції до розміщених в ньому вузлів, клієнт інтерфейсу браузера OPC виконує виклики AJAX через HTTPS, захищені за допомогою токенів захисту від підробки, для отримання даних з веб-додатки підключеної фабрики.

При необхідності клієнт використовує метод обміну, описаний в кроках з 4 по 10, для обміну даними з сервером OPC UA.

Література: [7, 8, 9].

ЛІТЕРАТУРА

1. Douglass Robert et al. IoT for Defense and National Security. Robert Douglass, Keith Gremban, Ananthram Swami, Stephan Gerali. Wiley-IEEE Press, 2023. 516 p.
2. Al-Turjman F., Yadav S.P., Kumar M., Yadav V., Stephan T. (Eds.) Transforming Management with AI, Big-Data, and IoT. Springer, 2022. 315 p.
3. Heins Kersten. NB-IoT Use Cases and Devices: Design Guide. Springer, 2022. 265 p.
4. Lele Chitra. Internet of Things (IoT) A Quick Start Guide: A to Z of IoT Essentials. BPB Publications, 2022. 227 p.
5. Bhardwaj, A., Al-Turjman, F., Kumar, M., Stephan, T., & Mostarda, L. (2020). Capturing-theinvisible (CTI): Behavior-based attacks recognition in IoT-oriented industrial control systems. IEEE Access, 1.
6. Singh, P., Singh, N., Singh, K. K., & Singh, A. (2021). Diagnosing of disease using machine learning. In Machine learning and the internet of medical things in healthcare (pp. 89–111). Academic Press.
7. Chithaluru, P., Al-Turjman, F., Kumar, M., & Stephan, T. (2020). I-AREOR: An energy-balanced clustering protocol for implementing green IoT in smart cities. Sustainable Cities and Society, 102254.
8. Yadav, S. P., Mahato, D. P., & Linh, N. T. D. (Eds.). (2020). Distributed artifcial intelligence: A modern approach. CRC Press.
9. Sheikh, J. A., Cheema, S. M., Ali, M., Amjad, Z., Tariq, J. Z., & Naz, A. (2020). IoT and AI in precision agriculture: Designing smart system to support illiterate farmers. Advances in Intelligent Systems and Computing, 490–496.
10. Kumar, M., Punia, S., Thompson, S., Gopal, D., & Patan, R. (2020). Performance analysis of machine learning algorithms for Big Data classifcation. International Journal of E-Health and Medical Communications (IJEHMC), 12(4), 60–75.
11. Satsyk, V., Cagáňová, D., Reshetylo, O., Zabolotnyi, O., Tkachuk, A. (2023). Increasing the Speed and Performance of the Drupal CMS Server for Industrial IoT Technologies. In: Balog, M., Iakovets, A., Hrehova, S. (eds) EAI International Conference on Automation and Control in Theory and Practice . EAI ARTEP 2023. EAI/Springer Innovations in Communication and Computing. Springer, Cham.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

І 73 **Інтернет Речей в електроніці**: конспект лекцій для здобувачів другого (магістерського) рівня вищої освіти освітньої програми «Електроніка» галузі знань 17 Електроніка, автоматизація та електронні комунікації спеціальності 171 Електроніка денної та заочної форм навчання / уклад. А.А. Ткачук. Луцьк: ЛНТУ, 2025. 64 с.

Конспект лекцій з дисципліни «**Інтернет Речей в електроніці**»: складений відповідно до діючої програми курсу.

Призначений для здобувачів вищої освіти спеціальності 171 Електроніка освітньої програми «Електроніка».

Комп’ютерний набір Анатолій ТКАЧУК

Редактор Анатолій ТКАЧУК

Підп. до друку «___» _____ 2025 р.
Формат 60х84/16. Папір офс. Гарнітура Таймс.
Ум. друк. арк. ____. Тираж 10 прим. Зам. ____

Відділ іміджу та промоцій
Луцького національного технічного університету
43018, м. Луцьк, вул. Львівська, 75