МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Навчально-науковий інститут інноваційних освітніх технологій
Кафедра комп'ютерної інженерії

МУРИН Мар'ян Васильович

Алгоритми ідентифікації об'єктів Інтернет речей на основі технології блокчейн / Algorithms for Internet things objects identifying based on blockchain technology

спеціальність: 123 - Комп’ютерна інженерія
магістерська програма - Комп’ютерна інженерія

Магістерська робота

Виконав студент групи КІм-21
М.В. Мурин
Науковий керівник: д.т.н., професор,
В. В. Яцків

Магістерську роботу допущено до захисту:

ТЕРНОПІЛЬ -2018

РЕЗЮМЕ

Магістерська робота на тему «Алгоритм ідентифікації об’єктів Інтернет речей на основі технологій блокчейн» зі спеціальності 123 «Комп’ютерна інженерія» написана обсягом 88 сторінок і містить 51 ілюстрації, 6 таблиць, 3 додатки та 50 джерела за переліком посилань.

Метою роботи є розроблення алгоритму ідентифікації об’єктів Інтернет речей на основі технологій блокчейн.

Методи досліджень. Для розв’язання поставлених задач у магістерській роботі використано: теорія інформації та кодування, методи теорії чисел, методи шифрування інформації, методи та засоби проектування програмного забезпечення.

Результати дослідження: алгоритм ідентифікації об’єктів Інтернет речей на основі технологій блокчейн.

Результати роботи можуть бути використані при проектуванні безпровідних систем передачі даних а також в навчальному процесі.

Орієнтовні напрямки розвитку досліджень: удосконалення алгоритму ідентифікації об’єктів Інтернет речей за рахунок використання технології блокчейн, що дозволило підвищити надійність ідентифікації.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, БЛОКЧЕЙН, ІНТЕРНЕТ РЕЧІ, ПЕРЕДАЧА ДАНИХ.

RESUME

Master's work on the topic "Algorithm for Internet things objects identifying based on blockchain technology" from the specialty 123 "Computer Engineering" is written in the volume of 88 pages and contains 51 illustrations, 6 tables, 3 annexes and 50 sources under the list of references.

The purpose of the work is to develop an algorithm for identifying objects of Internet solutions based on blocking technologies.

Research methods. For the solution of the tasks set forth in the master's work are used: information theory and coding, methods of theoretical numbers, methods of encryption of information, methods and tools for designing software.

Results of the research: algorithm of identification of objects of Internet solutions based on block technologies.

The results of the work can be used in the design of wireless data transmission systems and in the learning process.

Approximate directions of research development: improvement of the algorithm of identification of objects of Internet things through the use of block technology, which allowed to increase the reliability of identification.

KEYWORDS: ALGORITHM, BLOCKCHINE, INTERNET, DATA TRANSMISSION.

ЗМІСТ

ВСТУП

Інтернет речі називають четвертою індустріальною революцією, яка не лише спростить нам побут, а й дозволить великим підприємствам автоматизувати багато процесів та ухвалювати ефективні рішення на основі аналізу величезних обсягів даних.

Актуальність роботи. Інтернет речей перетворює звичні для нас речі у нові пристрої, створюючи як розумні годинники, так і розумні міста.

Це концепція комунікації об’єктів, які використовують технології для взаємодії між собою та з навколишнім середовищем. Також ця концепція передбачає виконання пристроями певних дій без втручання людини. Таким чином, всі пристрої в будинках, в автомобілях, на користувачеві виконують обробку інформації, її аналіз та обмін між собою та, залежно від результатів, приймають рішення і виконують певні дії.

Сфера Інтернет речі – один із головних світових трендів. Старі- пристрої стають частиною Інтернет мережі і виконують нові функції [49]. Недарма цю галузь вважають рушієм 4-ї індустріальної революції, яка зараз триває у світі.

Швидше буде правильно сказати, що Інтернет речей об'єднує реальні речі в віртуальні системи, здатні вирішувати абсолютно різні завдання. Ключова ідея концепції – з'єднати між собою всі об'єкти, які можна з'єднати, підключити до мережі, і за рахунок цього отримати синергію.

Прилади, котрі можна віднести до інтернету речей, зазвичай мають чотири характерні технології: ідентифікатор, датчики, інструменти для зв'язку з іншими приладами та вбудований комп'ютер.

В якості ідентифікатора використовуються RFID-мітки, QR-коди (ті чорно-білі квадратики, що їх може зчитувати майже кожен сучасний телефон) або інші технології. Мітка дозволяє приладу мати своє «ім'я».

Датчики і сенсори потрібні, щоб отримувати інформацію з середовища. Приміром, так фітнес-трекер зчитує інформацію про пульс людини. Також

прилади можуть отримувати інформацію від інших приладів та з мережі –за допомогою Bluetooth або WiFi.

Для обробки всіх отриманих даних і виконання програм використовується вбудований комп'ютер. Він може бути доволі складним як той, що керує автопілотом в сучасному автомобілі або ж зовсім примітивним.

Блокчейн – розподілена база даних, яка підтримує перелік записів, так званих блоків, що постійно зростає. База захищена від підробки та переробки [18]. Кожен блок містить часову мітку та посилання на попередній блок хеш дерева.

Всі дані в блокчейн накопичуються і формують постійно доповнюється базу даних. З цієї бази даних неможливо нічого видалити або провести заміну/підміну блоку. Це одна з головних особливостей технології блокчейн.

Важливе є те що за допомогою технології блокчейн безпечно передавати дані в мережі для пристроїв Інтернет речей.

Мета і завдання дослідження. Метою роботи є розроблення алгоритму ідентифікації об’єктів Інтернет речей на основі технологій блокчейн.

Досягнення визначеної мети передбачає вирішення таких завдань:

− провести аналіз технології блокчейн та алгоритмів Інтернет речей;

− розробити алгоритм ідентифікації пристроїв на основі технології блокчейн;

− розробити структурну систему ідентифікації;

− розробити алгоритм блокчейн;

− розробити публічний алгоритм технології блокчейн;

− реалізувати на програмній мові C# блокчейн;

− реалізувати публічний блокчейн;

− провести дослідження розроблених алгоритмів.

Об’єкт дослідження – процеси ідентифікації об’єктів Інтернет речей на основі технології блокчейн.

Предмет дослідження – методи та алгоритми ідентифікації об’єктів Інтернет речей на основі технології блокчейн.

Методи досліджень. Для розв’язання поставлених задач у магістерській роботі використано: теорія інформації та кодування, методи теорії чисел, методи шифрування інформації, методи та засоби проектування програмного забезпечення.

Наукова новизна одержаних результатів. Удосконалено алгоритми ідентифікації об’єктів Інтернет речей за рахунок використання технології блокчейн, що дозволило підвищити надійність ідентифікації.

Практичне значення отриманих результатів. Розроблено структуру та реалізовано на мові C# алгоритм ідентифікації на основі технології блокчейн.

Публікації та апробація МР. Мурин М. В., Васильків В. О. Порівняння алгоритмів консенсусу для Інтернет речей . Матеріали семінару комп’ютерні науки та інформаційні технології, CSIT’2018, Тернопіль, 2 червня 2018 р., С.49-50 [37].

Впровадження результатів МР. ТОВ “Апіко Україна”.

Дипломна робота складається із трьох розділів, висновків, списку використаної літератури та додатків. У першому розділі випускної кваліфікаційної роботи зроблений огляд існуючих алгоритмів взаємодії пристроїв Інтренет речей, принцип побудови Інетрент речей і область застосування блокчейн.

В другому розділі розглянуто платформу Microsoft Azure, проведено порівняння, переваги та тестування з іншими платформами. Проведено порівняння алгоритмів консенсусу Інтернет речей IOTA і IOTW та консенсусу технології бокчейн PoW, PoS, PoA. Розроблена структура системи ідентифікації об'єктів на основі технології блокчейн.

У третьому розділі розроблено програмну реалізацію алгоритму ідентифікації об’єктів Інтернет речей на основі технології блокчейн. Реалізовано публічний блокечейн на Microsoft Azure. Протестовано алгоритм роботи блокчейн.

1 АНАЛІЗ ПРИСТРОЇВ ІНТЕРНЕТ РЕЧЕЙ ТА АЛГОРИТМІВ ЇХ ІДЕНТИФІКАЦІЇ

1.1 Принципи побудови Інтернет речей

Інтернет речей - це мережа, що складається із взаємозв’язаних фізичних об’єктів (речей) або пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп’ютерними системами, за допомогою використання стандартних протоколів зв’язку [17]. Крім давачів, мережа може мати виконавчі пристрої, вбудовані у фізичні об’єкти і пов’язані між собою через дротові і бездротові мережі. Ці взаємопов’язані об’єкти (речі) мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

Якщо всі об’єкти (речі) будуть забезпечені мініатюрними радіо мітками, то їх можна буде дистанційно ідентифікувати, а при наявності певного «інтелекту» – і управляти ними. За оцінками експертів компанії Cisco кількість об’єктів, які Інтернет речей зможе з’єднати між собою, буде спів розмірна з кількістю атомів на поверхні Землі.

Інтернет речі є однією з найперспективніших технологій останніх років, що вже сьогодні фактично створює сотні нових продуктів і призводить до появи нових компаній на ринку [25]. Ви можете не помічати, але ви самі, ваші друзі чи колеги вже не перший рік користуються такими пристроями кожен день. Більше того, у чималій кількості домівок вже встановлені системи "розумного будинку", в які інтегровані десятки датчиків. Переваги інтернету речей, які вже доступні і які ще в процесі розробки можна краще продемонструвати на прикладах, тим паче, що сфер використання цієї технології чимало.

Інтернет речі ґрунтуються на трьох базових принципах [24].

По-перше, повсюдно поширену комунікаційну інфраструктуру, по-друге, глобальну ідентифікацію кожного об’єкта і, по-третє, можливість кожного об’єкта відправляти і отримувати дані за допомогою персональної мережі або мережі Інтернет, до якої він підключений.

Найбільш важливими відмінностями Інтернету речей від існуючого інтернету людей є:

– фокус на речах, а не на людині;
– істотно більша кількість підключених об’єктів;
– істотно менші розміри об’єктів і невисокі швидкості передачі даних;
– фокус на зчитуванні інформації, а не на комунікаціях;
– необхідність створення нової інфраструктури і стандартів.

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

– доступ до мережі Інтернет з метою передачі або запиту будь-яких даних;

– має конкретну адресу в глобальній мережі або ідентифікатор, за яким можна здійснити зворотний зв’язок з річчю;

– має інтерфейс для взаємодії з користувачем.

Інтернет-речі мають єдиний протокол взаємодії, згідно з яким будь-який вузол мережі рівноправний в наданні своїх сервісів. На шляху переходу до втілення ідеї Інтернету речей стояла проблема, пов’язана з протоколом IPv4, ресурс вільних мережевих адрес якого вже практично вичерпав себе. Однак підготовка до повсюдного впровадження версії протоколу IPv6 дозволяє вирішити цю проблему і наближає ідею Інтернету речей до реальності [29].

Кожен вузол мережі Інтернет-речей надає свій сервіс, надаючи якусь послугу поставки даних. У той же час вузол такої мережі може приймати команди від будь-якого іншого вузла. Це означає, що всі Інтернет-речі можуть взаємодіяти одна з одною і вирішувати спільні обчислювальні завдання. Інтернет-речі можуть утворювати локальні мережі, об’єднанні якоюсь однією зоною обслуговування або функцією.

Питаннями стандартизації та практичного впровадження окремих складових Інтернету речей (M2M, RFID, все проникні сенсорні мережі та ін.) Займаються багато міжнародних організації, неурядові асоціації, альянси виробників і операторів, партнерські проекти. В цілому для Інтернету речей, як нового напряму розвитку інформаційних комунікацій, в даний час визначені найзагальніші концептуальні і архітектурні рішення. Найближчим часом основною проблемою буде гармонізації різних стандартів з метою формування єдиної і несуперечливої нормативної бази для практичної реалізації Інтернету речей.

В рамках діяльності сектора стандартизації телекомунікацій Міжнародного союзу електрозв'язку (МСЕ-Т) є три глобальні ініціативи GSI (Global Standards Initiative). Під глобальної ініціативою розуміється комплекс робіт, виконуваних паралельно різними дослідницькими комісіями МСЕ відповідно до скоординованого плану роботи. Одна з таких ініціатив присвячена стандартизації Інтернету речей – IoT-GSI (Global Standards Initiative on Internet of Things). Дві інші глобальні ініціативи – по стандартизації мереж наступних поколінь NGN-GSI і систем телебачення на основі протоколу Інтернет IPTV-GSI також базуються на використанні IP-технологій, як і IoT-GSI.

Архітектура IoT має чотири рівні, які зображено на рисунку 1.1.

1. Рівень сенсорів і сенсорних мереж є найнижчий рівень архітектури IoT складається з «розумних» (smart) об'єктів, інтегрованих з сенсорами (датчиками). Сенсори реалізують з'єднання фізичного і віртуального (цифрового) світів, забезпечуючи збір та обробку інформації в реальному масштабі часу [14]. Мініатюризація, яка призвела до зменшення фізичних розмірів апаратних сенсорів, дозволила інтегрувати їх безпосередньо в об'єкти фізичного світу. Існують різні типи сенсорів для відповідних цілей, наприклад, для вимірювання температури, тиску, швидкості руху, місця розташування та ін. Сенсори можуть мати невелику пам'ять, даючи можливість записувати кілька результатів вимірювань. Сенсор може вимірювати фізичні параметри контрольованого об'єкта / явища і перетворювати їх в сигнал, який може бути прийнятий

відповідним пристроєм [48]. Сенсори класифікуються відповідно до їх призначення, наприклад, сенсори навколишнього середовища, сенсори для тіла, сенсори для побутової техніки, сенсори для транспортних засобів і т. д.

Рисунок 1.1 – Архітектура IoT рівнів

Більшість сенсорів вимагає з’єднання з агрегатором сенсорів (шлюзом), які можуть бути реалізовані з використанням локальних обчислювальних мереж (LAN, Local Area Network), таких як Ethernet і Wi-Fi або персональних мереж (PAN, Personal Area Network), таких як ZigBee, Bluetooth і ультраширокополосного бездротового зв’язку на малих відстанях (UWB, Ultra-Wide Band) [23]. Для сенсорів, які не вимагають підключення до агрегатору, їх зв’язок з серверами / додатками може надаватися з використанням глобальних бездротових мереж WAN, таких як GSM, GPRS і LTE [31].

Сенсори, які характеризуються низьким енергоспоживанням і низькою швидкістю передачі даних, утворюють широко відомі бездротові сенсорні мережі (WSN, Wireless Sensor Network). WSN набирають все більшої популярності, оскільки вони можуть містити набагато більше сенсорів з підтримкою роботи від батарей і охоплюють великі площі.

2. Рівень шлюзів і мереж має великий обсяг даних, що створюються на першому рівні IoT численними мініатюрними сенсорами, вимагає надійної та високопродуктивної провідної або бездротової мережевої інфраструктури в якості транспортного середовища. існуючі мережі зв'язку, що використовують різні протоколи, можуть бути використані для підтримки між машинних комунікацій M2M і їх додатків. Для реалізації широкого спектру послуг і додатків в IoT необхідно забезпечити спільну роботу безлічі мереж різних технологій і протоколів доступу в гетерогенній конфігурації. Ці мережі повинні забезпечувати необхідні показники якості передачі інформації, і перш за все по затримці, пропускній спроможності і безпеці. Даний рівень складається з конвергентної мережевої інфраструктури, яка створюється шляхом інтеграції різнорідних мереж в єдину мережеву платформу. Конвергентний абстрактний мережевий рівень в IoT дозволяє через відповідні шлюзи декільком користувачам використовувати ресурси в однієї мережі незалежно і спільно без шкоди для конфіденційності, безпеки і продуктивності.

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

4. На рівні додатків архітектури IoT існують різні типи додатків для відповідних промислових секторів і сфер діяльності (енергетика, транспорт,

торгівля, медицина, освіта та ін.). Додатки можуть бути «вертикальними», коли вони є специфічними для конкретної галузі промисловості, а також «Горизонтальними», (наприклад, управління автопарком, відстеження активів і ін.), які можуть використовуватися в різних секторах економіки.

Складовою частиною Інтернет речей є web of things, яка забезпечує взаємодію різних інтелектуальних об’єктів з використанням стандартів і механізмів Інтернет.

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

Забезпечує синхронну роботу інтелектуальних (смарт) об’єктів через прикладний програмний інтерфейс REST і в цілому відповідає ресурсно-орієнтованій архітектурі ROA (Resource-Oriented Architecture).

Надає асинхронний режим роботи інтелектуальних об’єктів з використанням в значній мірі стандартних Web-технологій, таких як Atom, містить формат для опису ресурсів на веб-сайтах і протокол для їх публікації, або Web-механізмів передачі даних, таких як модель роботи веб-додатку Comet, при якій постійне HTTP-з’єднання дозволяє веб-серверу відправляти дані браузеру без додаткового запиту з боку браузера [50].

Інтернет речей описує нову тенденцію, коли велика кількість вбудованих пристроїв (речей) підключено до Інтернету [26]. Ці підключені пристрої спілкуються з людьми та іншими речами, і часто надають дані датчиків для хмарного зберігання та ресурсів хмарних обчислень, де дані обробляються та аналізуються, щоб отримати важливі статистичні дані. Потужність хмарних обчислень і збільшення підключення пристроїв дають змогу розвивати цю тенденцію [45].

Такі рішення побудовані для багатьох вертикальних додатків, таких як моніторинг та контроль навколишнього середовища, моніторинг здоров'я, моніторинг автопарку, промисловий моніторинг та контроль, а також домашня автоматизація.

На високому рівні, системи можна описати, використовуючи діаграму, яка зображена на рисунку 1.2. На лівій стороні зображено інтелектуальні пристрої , які працюють у мережі [47]. Ці пристрої збирають дані та включають речі, як носії пристроїв, бездротові датчики температури, монітори серцебиття та датчики гідравлічного тиску і т. д. Середина діаграми являє собою хмарне середовище, де дані з багатьох джерел агрегуються та аналізуються в режимі реального часу, часто через платформу IoT. Платформа IoT збирає, обробляє та зберігає дані з розумних пристроїв, які часто географічно розподілені, і може мати можливість аналізу та вжиття заходів щодо вхідних даних.

Рисунок 1.2 – Схема перегляду систем IoT

У правій частині зображено розробку алгоритму, пов'язану з програмою IoT. Витягуються дані з платформи IoT щоб протипувати алгоритми і виконати у хмарних середовищах або на smart-devices.

1.2 Алгоритм взаємодії пристроїв Інтернет речей

Інтернетом речей розуміється сукупність різноманітних приладів, датчиків, пристроїв, об’єднаних в мережу за допомогою будь-яких доступних каналів зв’язку, що використовують різні протоколи взаємодії між собою і єдиний протокол доступу до глобальної мережі. У ролі глобальної мережі для Інтернет-

речей на даний час використовується мережа Інтернет [21]. Спільним протоколом є IP. Поява інтернету речей - це досить очікуваний крок. Алгоритм роботи людини і розумного будинку. Коли ще не було IoT люди робили все власноруч.

Приходили в будинок і включали пристрої які потрібні. З появою IoT стало все простіше, людина налаштовує правила які потрібно зробити в певний час і певну дію. Розглянемо приклад з лампочкою. Приходиш додому включаєш лампочку і йдеш виключаєш. Налаштовуємо правило щоб лампочка включалася і виключалася автоматично в певний час зображено на рисунку 1.3, 1.4.

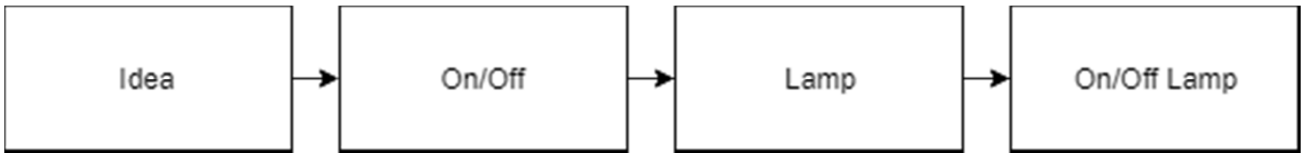


Рисунок 1.3 – Робота без інтернет речей

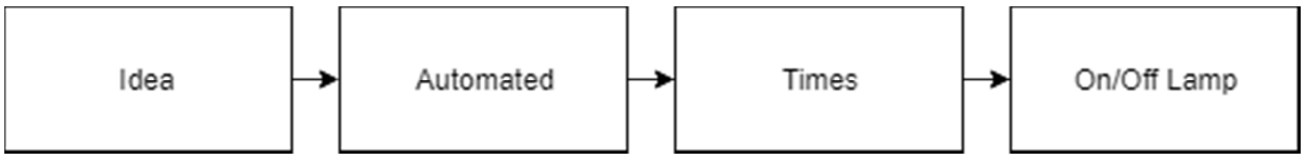


Рисунок 1.4 – Робота з Інтернет речей

Розумний будинок не получився тому що людина як і раніше повинна все контролювати, він центр управління всього. Виходить що це все лише автоматизація. Зображено на рисунку 1.5.

Рисунок 1.5 – Взаємодія людини і Інтернет речей

В розумінні розумних Інтернет речей це є Інтернет речі які дозволяють змінити парадигму досягнення результату.

Розумний Інтернет речей - постійна підтримка людини предметами, які його оточують.

Розумний Інтернет речей - це прозорість процесів, це орієнтація на результат.

Щоб цього досягнути по-перше потрібно мультиагентні технології вони вже по всюди, і Інтернет речей без них неможливий. Кожному учаснику з реального світу (тобто кожній людині і кожному пристрою) ставиться у відповідність програмний агент - об'єкт з певним ступенем інтелектуальності, що представляє його інтереси в світі віртуальному [22]. Віртуальний світ можна назвати певною мірою поліпшеною копією нашого життя: там є ті ж учасники, які найчастіше йдуть заздалегідь встановленим і відомими правилами, надаючи достовірні відповіді на поставлені запитання, чесні і відкриті - альтруїсти, в загальному. При цьому взаємозв'язок реального та віртуального світів двох направлених: рішення з віртуального світу віддаються в реальність для виконання, а всі події реального світу (дуже часто непередбачені) відображаються на світі віртуальному [40].

Життєвий цикл агентів досить простий. Спочатку вони сприймають інформацію із зовнішнього світу. Потім її потрібно обробити, тобто запланувати якісь дії. Ну а дії вже потрібно виконати - віддавши відповідні команди в реальний світ. Зображено на рисунку 1.6 життєвий цикл .

Виходить, що в "розумному" будинку агент людини постійно спілкується зрізними пристроями - віддаючи їм команди і обмінюючись інформацією [44].

Цю ситуацію можна змоделювати з точки зору агентів, пам'ятаючи, що кожен агент знає всю інформацію про свою фізичну сутність.

Рисунок 1.6 – Життєвий цикл

Виходить, що в "розумному" будинку агент людини постійно спілкується зрізними пристроями - віддаючи їм команди і обмінюючись інформацією.

Цю ситуацію можна змоделювати з точки зору агентів, пам'ятаючи, що кожен агент знає всю інформацію про свою фізичну сутність.

Існують онтології - це відносно універсальні способи представлення знань в машинному представленні, причому там можуть бути описані найрізноманітніші знання. В онтології можна описати важливі концепції, описати логічні правила - а інтелектуальні агенти використовувати ці знання для досягнення і взаємодії цілей. Чи можна розробити одну універсальну онтологію, яка буде містити всі потрібні для розумного Інтернету речей знання? Напевне так, але обсяг цієї онтології буде надвеликим. Значно простіше є можливість підтримки предметних областей - і, за необхідності, узгодження між ними зображено на рисунку 1.7.

Найбільш частий спосіб застосування онтологій - це лише спосіб зберігання знань, які жорстко структуровані. При цьому ці знання, як правило, стосуються лише певної сутності фізичного світу. А чому б не піти далі і не зберігати в онтологіях і правила взаємодії, логіку роботи розумного Інтернету речей? На практиці це може виглядати так: при створенні агент дивиться на сутність, до якої він належить. Для коректного розуміння властивостей цієї сутності агент повинен

звернутися до онтології - звідти він почерпне інформацію, що може робити ця сутність, які у неї потреби.

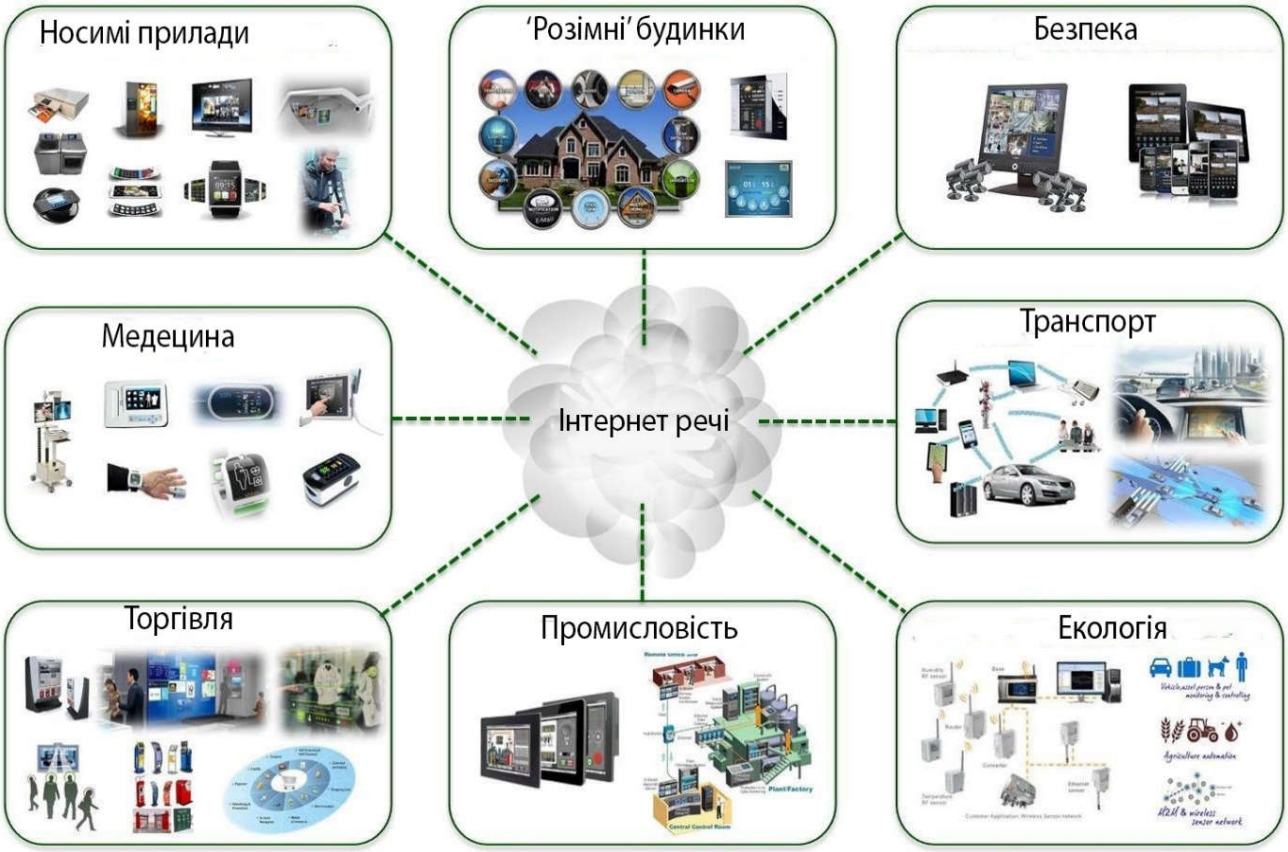


Рисунок 1.7 – Застосування Інтернет речей

MQTT (Message Queue Telemetry Transport) - відкритий протокол обміну даними створений для передачі даних на віддалених локаціях, де потрібно невеликий розмір коду і є обмеження по пропускній здатності каналу (зображено на рисунку 1.8) [27].

Протокол працює за схемою publisher/subscriber. Всі повідомлення від клієнта до клієнта проходять через центрального брокера. Протокол спроектований спеціально для мобільних додатків "Інтернет речей", де часто використовується в умовах повільної і нестабільної мережі. MQTT використовує мінімальну кількість трафіку і мінімальні характеристики до обладнання. Може використовуватися у будь-якому контролері, де може бути використаний протокол TCP/IP.

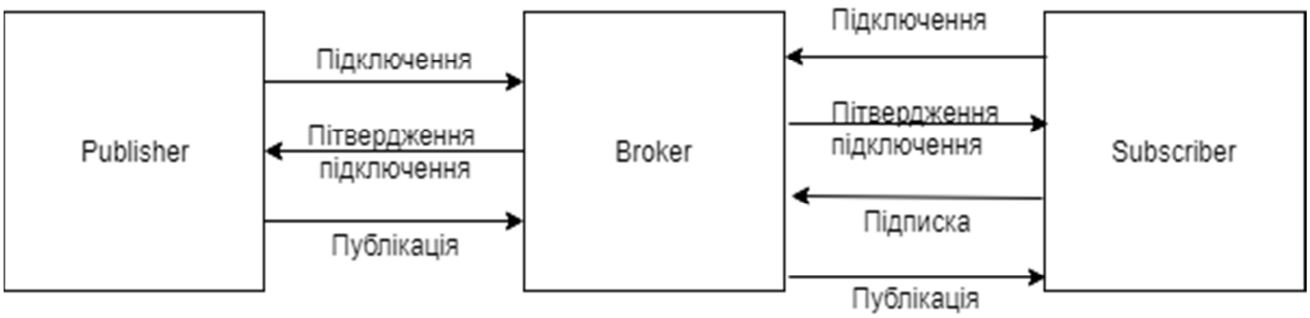


Рисунок 1.8 – Схема роботи MQTT

Publisher відправляє дані на MQTT broker, вказуючи в повідомленні певну тему. Subscribers можуть отримувати різні дані від безлічі publisher залежно від subscriber на відповідні топіки.

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

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

CoAP - це протокол обмеженого застосування з групи IETF CoRE (обмежених ресурсних середовищ).

Архітектура CoAP - це протокол передачі документів. На відміну від HTTP, CoAP призначений для потреб обмежених пристроїв. Пакети CoAP набагато менше потоків HTTP TCP (зображено на рисунку 1.9) [33].

CoAP працює над UDP, а не TCP. Клієнти та сервери обмінюються через бездротові datagrams. Повторні спроби та переналаштування виконуються в стек додатків. Видалення необхідності для TCP може дозволити повне з'єднання IP-адрес у невеликих мікроконтролерів. CoAP дозволяє використовувати UDP широкомовну передачу та багатоадресну передачу для адресації.

CoAP слідує моделі клієнта/сервера. Клієнти роблять запити на сервери, сервери відправляють зворотні відповіді. Клієнти можуть отримати ресурси GET,

PUT, POST і DELETE. Призначений для взаємодії з HTTP і RESTful Інтернетом в цілому за допомогою простих проксі-серверів. Оскільки CoAP заснований на datagram, його можна використовувати на додаток до SMS та інших пакетних протоколів зв'язку.

Безпека у CoAP побудована на вершині UDP, а не TCP, SSL/TLS недоступні для забезпечення безпеки. DTLS, Datagram Transport Layer Security забезпечує гарантії, що і TLS, але для передачі даних через UDP. Як правило, пристрої CoAP, що підтримують DTLS, підтримуватимуть RSA і AES або ECC та AES [41].

CoAP розширює модель HTTP - запиту з можливістю спостерігати за ресурсом. Коли позначка спостереження встановлюється на запит GAP CoAP, сервер може продовжувати відповідати після перенесення початкового документа. Це дозволяє серверам потоково змінювати стан клієнтів, коли вони відбуваються.

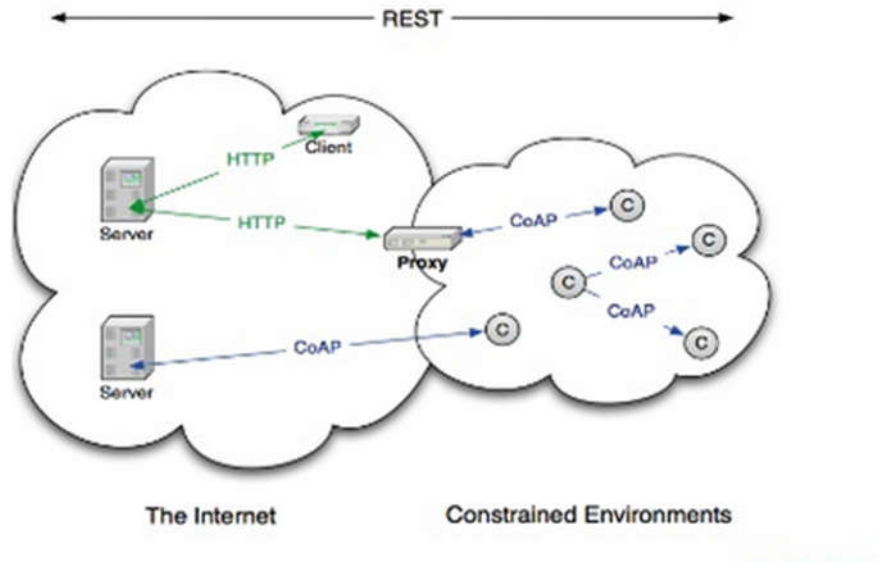


Рисунок 1.9 – Схема роботи CoAP

Виявлення ресурсів визначає стандартний механізм пошуку ресурсів. Сервери надають список своїх ресурсів (разом з метаданими про них) на /.well-known/core. Ці посилання знаходяться в форматі носія типу "додаток / посилання" і дозволяють клієнту з'ясувати, які ресурси надаються, і які типи засобів масової інформації.

Оскільки датчики CoAP є серверами, вони повинні мати можливість отримувати вхідні пакети. Щоб правильно функціонувати поза NAT, пристрій спочатку може надсилати запит на сервер, як це робиться в LWM2M, дозволяючи маршрутизатору пов'язувати ці два. Хоча CoAP не вимагає IPv6, його найлегше використовують в середовищах IP, де пристрої можуть бути безпосередньо інтерактивними.

Архітектура описує структуру рішення Інтернет речей, включаючи фізичні аспекти та віртуальні аспекти [46]. Прийняття багатоярусної архітектури дозволяє зосередити увагу на покращенні розуміння того, як всі аспекти архітектури працюють незалежно перед тим, як інтегрувати їх у програму Інтернет речей. Модульний підхід допомагає керувати складністю рішень IoT.

Для прикладних Інтернет речей - додатків, які використовують edge, основна трирівнева архітектура, фіксує потік інформації з пристроїв, до edge служб, а потім до хмарних служб. Більш детальна архітектура IoT також включатиме вертикальні шари, які проходять через інші шари, такі як управління ідентифікацією або безпека даних (зображено на рисунку 1.10).

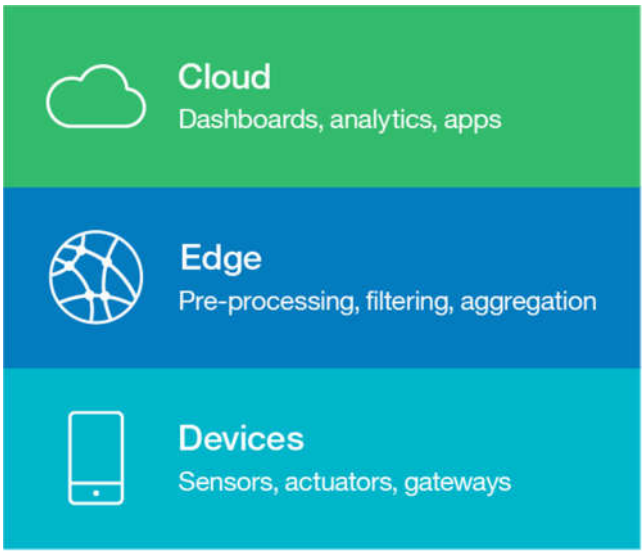


Рисунок 1.10 – Трирівнева архітектура IoT

Компоненти на рівні пристрою (рисунок 1.9) включають фізичні датчики та виконавчі пристрої, які підключені до пристроїв IoT та самих пристроїв IoT. Датчики та виконавчі пристрої зазвичай не розглядаються як smart пристрої, але

датчики та виконавчі пристрої часто з'єднуються як безконтактно, так і без провідна з використанням технологій, таких як Bluetooth LE або ZigBee, для пристроїв IoT, що мають більше можливостей обробки.

Деякі пристрої IoT спілкуються безпосередньо з пов'язаними хмарними службами та додатками. Тим не менш, для пристроїв IoT звичайно з'єднуються провідні шлюзи, які є проміжними пристроями, що мають трохи більше обсягу обробки, ніж основні пристрої IoT. Хоча вони не завжди мають датчики безпосередньо, шлюзові пристрої відіграють важливу роль у процесі збору даних. Вони можуть виконувати основні перетворення аналогового-цифрового сигналу, масштабування та іншу нормалізацію показників даних вихідних датчиків.

Другий рівень відноситься до служб аналітики та попередньої обробки, які знаходяться в мережі. Аналіз Edge відбувається в реальному часі, обробляючи потік даних у точці, де дані збираються, як це відбувається з датчиків. Основна задача попередньої обробки, така як фільтрація та агрегування даних, виконуються. Дані попередньо обробляються, передаються у хмарні службами та додатками для подальшої обробки та аналітики.

Після підготовки даних передаються у хмарні служби. Хмарні служби, що виконують обробку даних, часто доповнюються мобільними додатками та веб-програмами клієнта, які надають дані кінцевим користувачам і забезпечують доступ до інструментів для подальшого вивчення та аналізу через інформаційні панелі та візуалізації.

Безпека в рамках рішення IoT, повинна бути пріоритетом для всіх шарів архітектури IoT. При підключені багато пристроїв, цілісність системи в цілому повинна бути збережена, навіть якщо окремі пристрої чи шлюзи скомпрометовані. Перевірте чи ваша архітектура підтримує кілька рівнів захисту. Крім того потрібно переконайтеся, що рішення IoT може ідентифікувати та нейтралізувати пристрої, які стають скомпрометованими, наприклад, використовуючи шлюзи для виявлення вразливих пристроїв та моніторингу повідомлень та моделей використання для виявлення аномалій.

Потрібно прийняти стандарти для цих аспектів інфраструктури IoT:

– пристрій, додаток та ідентифікатор користувача, автентифікація, авторизація та контроль доступу;

– ключ управління;

– захист даних;

– безпечні канали зв'язку та цілісність повідомлень (за допомогою шифрування);

– аудит;

– безпечний розвиток та доставка.

Архітектура мікросервісів на Edge або Cloud Layer може бути реалізована за допомогою контейнерних технологій. Функції створюють такі операції, як настроювання нового пристрою чи шлюзу або розгортання нового примірника додатка для хмар, щоб обробляти дані пристрою. Уникнення ручної настройки гарантує, що операції є повторюваними, що є важливим для того, щоб бути в змозі розширити можливості IoT - рішень, що містять багато під'єднаних пристроїв.

Розробка рішень IoT, керованих даними, складна через масштаб і неоднорідність пристроїв та пов'язані з ними зв'язки.

1.3 Область застосування блокчейн

Блокчейн - база даних, що складається з ланцюжків транзакцій, що має певні правила і надає доступ до інформації. Все це виключає шахрайські дії, крадіжку персональних даних, захищає майнові права.

Публічний блокчейн - відкрита база даних. Такий вид блокчейна використовується в криптовалютах. Кожен учасник може записувати і читати дані.

Приватний блокчейн має обмеження по запису і читанні даних. Можуть встановлюватися пріоритетні вузли [1].

Рисунок 1.11 – Область застосування Blockchain

Швидкі і зручні фінансові транзакції за допомогою блокчейна можна спростити роботу з усіма видами фінансових операцій: робота зі звітами про витрати, кредити і мікрофінансування, ICO, транзакції з акціями та фондами зображено на рисунку 1.11.

Облік матеріальних активів і загальнодоступних документів в журнал записів можна додавати дані про різні документах, свідченнях, посвідченнях, завдяки технології можна завіряти ці документи[16].

Державні реєстри на блокчейні в сфері державного управління дозволить скоротити рівень бюрократизації.

Щоб захистити авторські права, що до недавнього часу було досить складним завданням, так як довести, що це саме ваша інтелектуальна власність, без гарантій дуже важко.

Щоб брати участь у голосуванні з допомогою відкритого реєстру. Завдяки блокчейну, підробити результати виборів буде неможливо.

Технологію блокчейн можна використовувати в юриспруденції та управлінської діяльності. Зокрема, зберігати так відомості про розподіл бюджетних коштів.

В області нерухомості, щоб прискорити угоди купівлі-продажу, підтвердити право власності на майно.

На біржах і в секторі послуг і навіть у благодійності.

Блокчейн використовують не тільки у фінансових але інших:

У банківській сфері. Банки застосовують блокчейн у своїй роботі. Було розроблено платформу Ripple.

Ascribe допомагає художникам і іншим творчим людям підтвердити своє авторство за допомогою блокчейна. Ascribe дозволяє створити цифровий унікальний ID і цифровий сертифікат справжності, що дозволяє однозначно довести авторство і справжність твори. Це також спосіб приймати цифрові роботи від художників і передавати їх покупцям з дотриманням всіх прав і законів [3].

Factom займається застосуванням розподіленого реєстру на фінансовому ринку - вона займаються управлінням даними. Компанія використовує засновані на блокчейне реєстри для управління базами даних і аналізу інформації, що необхідно для різних застосувань. Бізнесмени і уряду можуть використовувати Factom для того, щоб зробити простіше управління своїми записами, зберігаючи бізнес процеси і адреси без можливого шкоди для безпеки. Factom веде постійну, з часовою позначкою запис даних в блокчейн, що дозволяє компаніям скоротити витрати і складність при проведенні аудиту, управлінням записами і дотриманні вимог регуляторів [7].

Everledger забезпечує не змінюваний реєстр для ідентифікації алмазів і перевірки транзакцій між учасниками ринку, якими також можуть бути страхові компанії, правоохоронні органи. Оскаржуючи право власності Everledger створила «цифровий паспорт», який є унікальним [6].

Grid Singularity децентралізована платформа для обміну енергією, яка спрощує аналіз даних в області енергетики і дозволяє розумне управління мережами, торгівлю сертифікатами на зелену енергетику, інвестиційні рішення і торгівлю електрикою [9].

Follow My Vote пропонує безпечне і прозоре рішення для онлайн голосування, яке використовує технологію блокчейна і криптографію на еліптичних кривих для того, щоб переконатися в чесності і точності підрахунку результатів виборів. Розроблено платформу для голосування з відкритим вихідним кодом, яка забезпечує прозорість виборчого процесу [8].

Блокчейн можна застосовувати для забезпечення прозорості та цілісності політичної системи, він також використовується в проекті BITNATION, це перша в світі віртуальна нація.

Платформа Borderless коаліція сервісів управління через розумні контракти, побудована на блокчейні Expanse [4].

Verbatim заснований на блокчейн протоколі, який дозволяє людям надавати свої документи без необхідності в додатковій перевірці третьою стороною [20].

Appii використовує технологію розподіленого реєстру для безпечного зберігання та перевірки інформації, яка стосується освіти, акредитації, нагород, дипломів і трудового стажу. Така технологія дозволяє отримати швидкий доступ до трудових даними професіоналів [2].

Ethereum платформа і мова програмування, яка робить можливим для будь-якого розробника побудувати і опублікувати розподілене додаток наступного покоління. Ethereum може використовуватися для кодування, децентралізації, безпеки та здійснення операцій практично в будь-якій області: голосування, доменні імена, обмін грошей, спільне фінансування, управління, контракти, угоди, інтелектуальна власність і навіть розумна власність завдяки інтеграції з апаратним забезпеченням [5].

UBITQUITY пропонує користувачам зручний спосіб безпечного внесення даних, їх відстеження, а також передачі майнових прав через блокчейн платформу SaaS. Це допомагає фірмам, які займаються нерухомістю та іпотекою, так як управління записами стає зручніше, скорочується час пошуку, збільшується конфіденційність і прозорість [19].

Переваги технології блокчейн [13] показано нижче:

1. Прозорість блокчейн є публічною базою всіх транзакцій, здійснених в системі.

2. Незалежна перевірка шифрування (хешування) виконується на різних комп'ютерах однієї мережі, якщо в результаті розрахунків усі учасники отримують однаковий результат, блоку присвоюється цифрова сигнатура.

3. Неможливість підробки - після присвоєння блоку сигнатури і додавання блоку в блокчейн, реєстр оновлюється одночасно у всіх користувачів, змінити або видалити блок в ланцюжку неможливо. Можна тільки додавати нові блоки в ланцюжок.

4. Децентралізованість реєстру одночасно зберігається у всіх користувачів мережі блокчейна, будь-який користувач мережі має доступ до актуальної версії реєстру.

5. Неможливо отримати доступ одночасно до всіх копій блокчейна на всіх комп'ютерах мережі.

Недоліки технології блокчейн показано нижче [30]:

1. Транзакції в мережі блокчейн не можна скасувати або повернути.

2. У системи блокчейн не висока продуктивність, тому використання її в глобальному масштабі не вийде. Система вразлива, тому що обробляє занадто мало транзакцій в секунду, не порівнянну з кількістю транзакцій.

3. Незважаючи на високу незалежність ланцюга, якщо один користувач буде володіти 51% всіх блоків, про її децентралізації доведеться забути, і ймовірність атаки значно зросте.

На основі технології блокчейн можна створювати будь-які реєстри, дані в яких, в силу неможливості будь-якої фальсифікації, можуть використовуватися в якості юридичних документів. Крім того, обслуговування таких реєстрів очікує набагато менше ресурсів, адже реєстр розподілений між усіма учасниками системи, автоматичне оновлення відбувається відразу після внесення змін.

Блокчейн-технологія допомагає ефективно організувати спільну роботу.

Розглянуто основні поняття технології блокчейн та Інтернет речей.

Проаналізовано алгоритми взаємодії пристроїв з Інтернет речами, протоколи з’єднання MQTT і CoAP та область застосування блокчейн.

Розглянувши існуючі алгоритми взаємодію пристроїв з Інтернет речами, можна зробити висновок, що всі алгоритми є складними через масштабність і неоднорідність пристроїв та пов'язані з ними зв'язки.

Зроблено постановку задачі для подальшої роботи для досягнення поставленої мети.

2 АЛГОРИТМИ ІДЕНТИФІКАЦІЇ ОБ’ЄКТІВ НА ОСНОВІ ТЕХНОЛОГІЇ БЛОКЧЕЙН

2.1 Платформа Microsoft Azure

Microsoft Azure – це хмарна платформа та інфраструктура корпорації Microsoft, призначена для розробників застосунків хмарних обчислень і покликана спростити процес створення онлайнових додатків [15].

Розробка хмарних додатків зазвичай займає набагато менше часу, ніж традиційних, з двох причин. По-перше, немає необхідності налаштовувати і підтримувати обчислювальну і мережеву інфраструктуру, також інфраструктуру зберігання, на якій будуть працювати додатки [43]. Замість цього можна використовувати ресурси, що надаються постачальником хмарного хостингу. Тут все аналогічно висвітлення будинку. Якщо вам, крім покупки лампочок, доведеться тягнути проводку, встановлювати вимикачі, купувати і монтувати генератор, а ще й купувати бензин для нього, то справа неабияк затягнеться. А якщо лампочки будуть підключатися до проведення, яку провели будівельники, і працювати від електрики, яке виробляє місцева підстанція, то організувати освітлення не складе труднощів. Друга причина, по якій хмарні додатки розгортаються швидше звичайних, пов'язана з процесом розробки. У типовому корпоративному оточенні розробники створюють і тестують додатки в тестовому середовищі, яка не в повній мірі відповідає реальній. Наприклад, додаток може розроблятися і тестуватися на некластеризованних вузлах, а потім працювати на кластеризованних вузлах. Подібні розбіжності між середовищем розробки і реальним оточенням уповільнюють процес розробки бізнес-додатків, так як при тестуванні не завжди вдається виявити проблеми, які стають очевидними тільки в реальних умовах експлуатації, що часто призводить до додаткових циклам розробки і тестування. В хмарі можна розробляти і тестувати програми в тих же середовищах, в яких вони будуть розгортатися, тобто на обчислювальних і мережних ресурсах, а також на ресурсах зберігання вашого постачальника

хмарного хостингу. Це спрощує тестування додатків і робить його більш надійним, що, в свою чергу, прискорює розробку.

Microsoft Azure є хмарної платформою і дозволяє працювати з бізнес-додатками, службами та завданнями в хмарі. Є кілька ключових слів, які характеризують платформу [38]:

1. Відкрита надає ряд хмарних служб, за допомогою
яких можна створювати і розгортати Хмарний додатки засобами практично будь-якої мови програмування, структури або інструменту.

2. Гнучка включає в себе безліч хмарних служб будь-яких завдань - від хостингу веб-сайту компанії в роботи з великими базами даних SQL в хмарі. Крім того, в цій платформі є функції, Які забезпечують Високу продуктивність і низьку затримку в мережі для хмарних додатків.

3. Під управлінням служби Microsoft Azure розміщені в декількох центрах обробки даних в США, Європі та Азії. Ці центри управляються корпорацією Майкрософт і забезпечують цілодобову професійну підтримку.

4. Сумісна з хмарними додатками можна легко інтегрувати з локальним IT-середовищами, які працюють на базі Microsoft Windows Server.

Надає компаніям хмарні служби чотирьох основних типів:

- обчислювальні служби;
- мережеві служби;
- служби обробки даних;
- служби додатків.

Обчислювальні служби Microsoft Azure надають комп'ютерні ресурси, на яких працюють хмарні додатки. Актуальна версія Microsoft Azure підтримує чотири обчислювальні служби:

1. Віртуальні машини. Надає універсальну обчислювальну середу, в якій можна створювати, розгортати і керувати віртуальними машинами в хмарі Microsoft Azure.

2. Веб сайти. Надає керовану хмарну веб-середовище, в якій можна як створювати нові веб-сайти, так і переносити в неї існуючі.

3. Хмарні служби. Дозволяє створювати і розгортати майже необмежено масштабовані додатки високої доступності практично на будь-якій мові програмування і з мінімальними витратами на адміністрування.

4. Мобільні служби. Є готовим до використання рішенням для створення і розгортання додатків, а також для зберігання даних для мобільних пристроїв.

Мережеві служби Microsoft Azure дозволяють надавати хмарні додатки користувачам і центрів обробки даних різними способами. Актуальна версія Microsoft Azure підтримує дві мережеві служби:

1. Віртуальна мережа. Дозволяє використовувати загальнодоступне хмара Microsoft Azure в якості розширення локального центру обробки даних.

2. Диспетчер трафіку. Дозволяє маршрутизувати трафік додатків для користувачів центрів обробки даних Microsoft Azure трьома способами: з максимальною продуктивністю, за принципом циклічного обслуговування або на основі відмовостійкої активно-пасивної конфігурації.

Служби обробки даних Microsoft Azure дозволяють різними способами зберігати, управляти, захищати і аналізувати бізнес-дані, а також складати звіти по ним. Актуальна версія Microsoft Azure підтримує п'ять служб обробки даних:

1. Управління даними. Дозволяє зберігати бізнес-інформацію в базах даних SQL різними способами - на виділених віртуальних машинах Microsoft SQL Server, в базі даних Microsoft Azure SQL, в таблицях NoSQL через REST або в BLOB-сховище.

2. Бізнес-аналітика полегшує вивчення і підвищує інформативність даних за допомогою служб Microsoft SQL Server Reporting and Analysis або Microsoft SharePoint Server, що працюють на віртуальній машині, Microsoft Azure SQL Reporting, Microsoft Azure Marketplace або HDInsight – реалізації «Великих даних» на базі Hadoop.

3. HDInsight. Розроблена Майкрософт на основі Hadoop, ця служба повністю повторює функціональність Apache Hadoop в хмарі.

4. Кеш. Включає в себе розподілене рішення для кешування, що прискорює роботу хмарних додатків і знижує навантаження на базу даних.

5. Резервне копіювання. Допомагає автономно захистити дані на сервері, дозволяючи створювати як автоматичні, так і ручні копії в Microsoft Azure.

Антивірус для Microsoft Azure забезпечує захист в реальному часі або за розкладом, а так само надає збір інформації про атаки вашого профілю через Azure Diagnostics. Побудований на тій же платформі, що і Microsoft Security Essentials, Microsoft Forefront Endpoint Protection, Microsoft System Center Endpoint Protection, Windows Intune, і Windows Defender для Windows, 8.0 і вище. І призначений для роботи у фоновому режимі. Розробник може налаштувати антивірус, грунтуючись на потребах свого рішення, вибрати роботу захисту в режимі реального часу або сканування за розкладом, запустити оновлення засобів захисту або збір інформації про шкідливі пригодах. Антивірус встановлюється за умовчанням на кожному примірнику Cloud Services.

Можемо включити службу захисту, використовуючи базову конфігурацію або налаштувати її за власним бажанням. Налаштування служби захисту за замовчуванням вже оптимізовані для роботи в середовищі Microsoft Azure. Можемо так само налаштувати їх відповідно до специфіки вашого рішення. На рисунку 2.1 зображено схему роботи антивірусу Microsoft Azure .

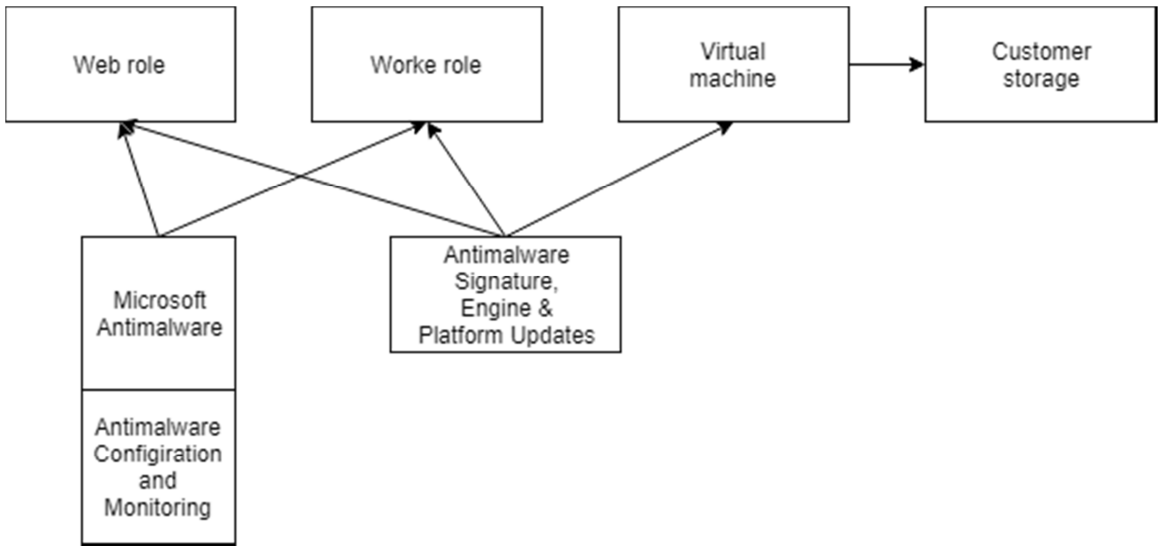


Рисунок 2.1 – Схема роботи антивірусу Microsoft Azure

Розглянемо принцип рольової моделі на основі типового Web-програми, написаної з використанням MVC (Model-View-Controller). Зображено на рисунку 2.2 Стандартна схема MVC:

– Web-інтерфейсу користувача;
– Контролер;
– Шар доступу до джерела даних (БД).

Рисунок 2.2 – Стандартна схема MVC

Доступ до програми здійснюється по кінцевій точці доступу (HTTP або HTTPS).

– Подання змінює свою назву на Web-роль;
– Контролер - на Worker-роль;
– Доступ до джерела даних може бути реалізований всередині Worker.

Для отримання даних від Web-ролі оброблювачем-контролером (Worker-роллю) традиційно використовується Middleware у вигляді сервісу черг. Використання Middleware є прекрасним прикладом розробки відмовостійких розподілених додатків. Замість розробки тісно пов'язаної системи, для якої втрата одного з компонентів (наприклад, Web-ролі) буде означати непрацездатність всієї системи і втрату даних розробник може використовувати Middleware в режимі брокера (Brokered Messaging).

Це означає, що Web-роль просто кладе повідомлення в сервіс Middleware (в чергу), де вони зберігаються до тих пір, поки ними не займе збирач сміття, або поки вони не оброблені екземплярами Worker-ролі зображено на рисунку 2.3.

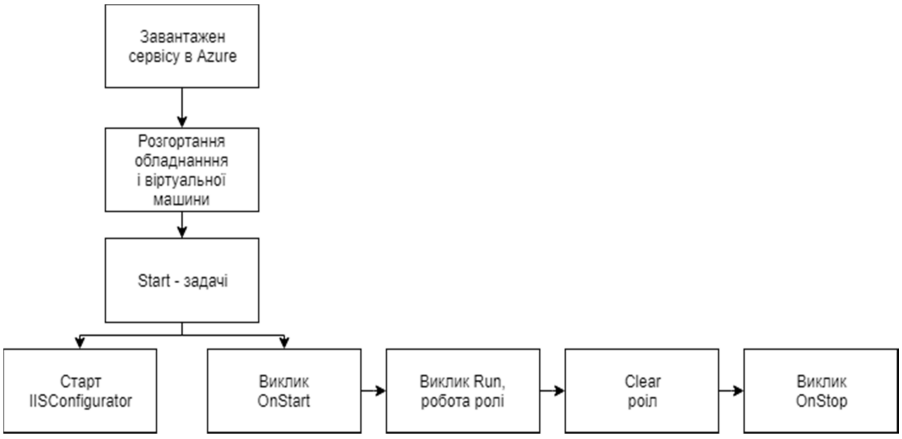


Рисунок 2.3 – Web - ролі

У той же час Web-роль не має поняття про те, скільки оброблювачів займаються приходять з неї повідомленнями, існують ці обробники, оффлайн вони або онлайн, і не знає про статус обробки цих повідомлень (якщо це спеціально не передбачено). Очевидно, що зв'язність системи значно знижується, адже тепер вихід з ладу частини системи не призведе до збою системи в цілому.

Як видно на малюнку архітектури Microsoft Azure зображено на рисунку 2.4, кожна з ролей існує в певній кількості примірників, які виконують одну й ту ж функцію ролі.

Це означає, що, перейшовши за посиланням на Web-сайт, користувач може потрапити як на перший екземпляр Web-ролі, так і на будь-який інший, що знаходиться в ротації балансування навантаження.

Рисунок 2.4 – Архітектуру Microsoft Azure

Технологія блокчейн в Azure це децентралізована (відкрита та криптографічна) яка дозволяє довіряти один одному та здійснювати взаємодію.

Оскільки автономні системи та пристрої взаємодіють один з одним, транзакції IoT піддаються потенційному ризику для безпеки. Технологія Blockchain дає простий, економічно ефективний та постійний реєстр рішень, прийнятих та переданих.

Відстеження – дані транзакції відбуваються між кількома мережами, що належать і управляються кількома організаціями. Блокчейн може забезпечити постійний, незмінний запис так, що зберігання може бути відстежено, коли дані або фізичні товари, переміщаються між точками в ланцюжку вартості. Записи блокчейн за своїм характером є прозорими - активність може відслідковуватись та проаналізувати будь-хто, уповноважений для підключення до мережі.

Безпека – мереж Інтернет речей є однією з найбільших проблем. Якщо дані обмінюються мережею блокчейн, загальна безпека мережі Інтернет речей значно покращується.

Розумні контракти - мають розумні програми або контракти, які автоматично виконуються, коли умови виконуються. Використовуючи інтелектуальні контракти, дії можуть виконуватися різними організаціями в ланцюжку поставок, автоматично, незмінно, не хвилюючись про суперечки.

Обчислювальний хмара Amazon Elastic Compute Cloud (Amazon EC2) - це веб-сервіс, що надає безпечні масштабовані обчислювальні ресурси в хмарі. Він допомагає розробникам, спрощуючи проведення обчислень в хмарі в масштабі всього Інтернету.

Простий веб-інтерфейс сервісу Amazon EC2 дозволяє отримати доступ до обчислювальних ресурсів і налаштувати їх з мінімальними зусиллями. Він надає користувачам повний контроль над обчислювальними ресурсами, а також перевірену обчислювальну середу Amazon для роботи. Amazon EC2 дозволяє скоротити час, необхідний для створення і завантаження нових instance сервера, до декількох хвилин, і забезпечує можливість швидко масштабувати в будь-якому напрямку з урахуванням мінливих вимог до обчислювальних ресурсів. Amazon EC2 змінює економічну складову процесу обчислень, надаючи можливість платити тільки за використовувані ресурси. Amazon EC2 дозволяє розробникам уникати поширених хибних сценаріїв і створювати відмовостійкі додатки.

Google Cloud Platform служить для збірки, тестування і розгортання додатків в надійної масштабованій інфраструктурі і пропонує обчислювальні потужності, зберігання і обслуговування програмних рішень.

В таблиці 2.1 зображено параметри продуктивності віртуальних машин.

Таблиця 2.1 – Продуктивність віртуальних машин

Cloud	Vm Size	Cores	Ram	Price/Hour на windows
Azure	Standard D3 v2	4	14GB	$0.422
AWS	m4.xlarge	4	16GB	$0.406
Google	n1-standard-4	4	15GB	$0.306

Для тестування було використано платформи:

– GeekBench;

– CrystalDiskMark;

– Hardinfo.

У GeekBench результати багатоядерного тестування продемонстрували такі результати зображені у таблиці 2.2. AWS і Google Cloud видають таку картину: Intel Xeon GHz 1 processor, 2 cores, 4 threads, в той час як Azure Intel Xeon E5-2673 v3 GHz 1 processor, 4 cores. Найбільшу кількість балі набрав Azure з розривом майже у півтора рази.

Таблиця 2.2 – Результати тестування GeekBench

Cloud	GeekBench Score
Azure	9508.4
AWS	6568.6
Google	6188.2

Результат у CrystalDiskMark самі неоднозначні, але швидше за все така різниця зумовлена тим, що платформи Google Cloud / AWS необхідно додатково конфігурувати для досягнення оптимальної продуктивності, в той час як Azure в

конфігурації за замовчуванням показують хорошу продуктивність зображено на рисунку 2.5.

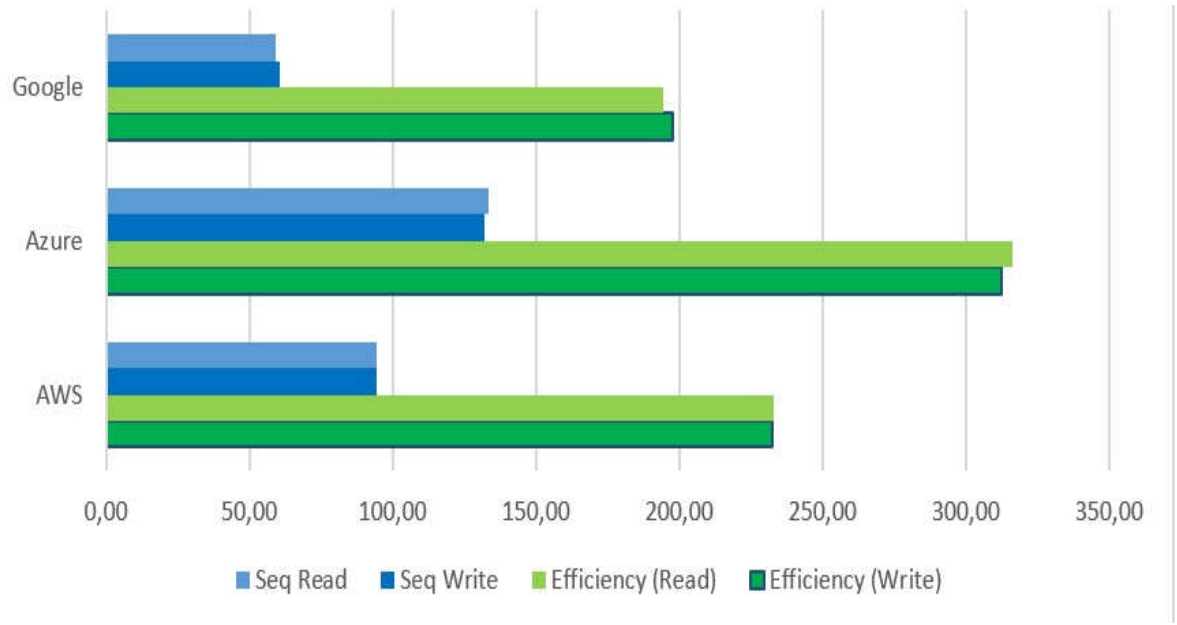


Рисунок 2.5 – Результат CrystalDiskMark

У даній конфігурації Azure набрав найбільше балів за всіма показниками

послідовне читання / запис, 4к читання / запис, 4к читання / запис з глибинної черзі рівній 32.

Google має відмінну систему дисконтування, яка включається автоматично і, на перший погляд, дійсно дає великий бонус.

Amazon є дуже сильний з точки зору довіри до бренду і найбільш популярна платформа.

Azure має перевагу в продуктивності за рахунок продажу справжніх ядер, а не віртуальних hyper-threading threads (vCPU).

Тести є синтетичними і для кожного окремого рішення або додатки потрібно проводити спеціалізовані навантаження, але це може бути відправною базовою точкою при виборі.

2.2 Алгоритми ідентифікації пристроїв на основі блокчейн.

Блокчейн – ланцюжок блоків транзакцій розподілена база даних, яка підтримує перелік записів, так званих блоків, що постійно зростає. База захищена від підробки та переробки. Кожен блок містить часову мітку та посилання на попередній блок хеш дерева. Це означає, що можемо подивитися історію транзакції і шлях, по якому вона здійснювалась. Інформація про розмір угоди теж відкрита. При цьому особистість адресата і адресанта не розкриваються. У цьому полягає прозорість блокчейна.

Транзакції рухають значення від входів до виходів. Кожен вхід визначає джерело вартості - або випуск нових одиниць, або вихід з попередньої транзакції. Кожен вивід визначає призначення, а саме - програму керування, яка визначає правила витрачання цього випуску в майбутньому. Кожен вхід і вихід визначають кількість одиничного ідентифікатора ресурсу. Транзакція може змішувати входи та виходи різних ідентифікаторів активів або об'єднувати або розділяти входи на виходи з різними сумами, поки загальні значення входів і загальні значення результатів балансують. Вхід повинен відповідати програмі видачі (якщо джерелом значення є нова видача одиниць) або програмою керування (якщо джерелом значення є попередня невикористана транзакція). Issuer або spender може передавати аргументи до програми через поле свідків.

Наприклад, програма видачі або керування може здійснювати перевірку авторизації шляхом визначення відкритого ключа та вимагання криптографічного підпису транзакції за допомогою відповідного приватного ключа. Кожна транзакція та кожний окремий вхід і вихід включають поле довідкових даних для довільного використання на рівні програми. Протокол ланцюга визначає, як посилання дані використовуються для блочного шаблону, але не передбачає жодної структури чи семантики для її вмісту.

Після того як транзакція витратить певний вихід, жодна інша транзакція не може витрачати той самий вихід. Для запобігання подібних "подвійних витрат"

мережа підтримує унікальне і незмінне замовлення всіх транзакцій разом із станом блокчейну, що складається з невитрачених транзакційних виходів (UTXO). Вхідні дані транзакції дозволяють відсилати лише невитрачені виходи з набору UTXO. Як тільки транзакція застосовується, витрачені витрати видаляються з набору, а до нього додаються нові виходи.

Блок - це структура даних, яка складається з декількох транзакцій, які повинні бути виконані. Кожен заголовок блоку містить хеш попереднього блоку, перетворюючи серію блоків у незмінний блочний блок зображено на рисунку 2.6.

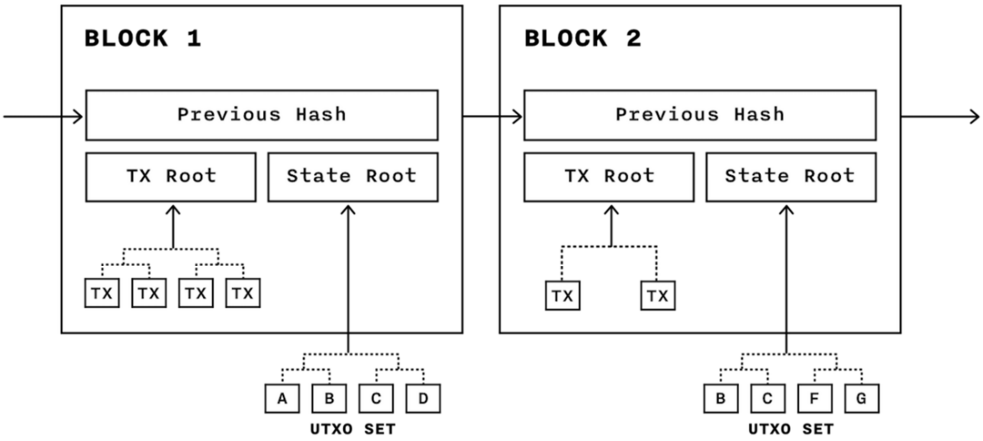


Рисунок 2.6 – Структурна схема Blockchain

Блок містить хеш усіх своїх транзакцій та хешу поточного стану, тобто набір поточних невитрачених виходів. Стан blockchain дозволяє учаснику мережі легко приєднатися до поточної мережі без необхідності повторювати всю історію блокчейну. Хеші являють собою коріння двох Merkle дерев, які дозволяють мати компактні докази існування будь-якої конкретної транзакції або невитраченого виходу і дозволяють одночасно завантажувати та перевіряти їхні елементи. Щоб запобігти неавторизованим учасникам створення нових блоків, кожен новий блок повинен відповідати консенсусній програмі, яка вказана в заголовку попереднього блоку. Програма консенсусу - так само, як програма видачі чи контролю - отримує аргументи від поля свідків нового блоку. Наприклад, консенсусна програма може вказувати відкритий ключ і вимагати, щоб свідок наступного

блоку містив підпис відповідним приватним ключем, використовуючи хеш блокування як повідомлення. Це є основою консенсусних програм, що використовуються в протоколі консенсусу. Гнучкість і сила мови програмування, проте означає, що протокол теоретично може підтримувати довільні алгоритми консенсусу - навіть складні, засновані на “proof-of-work” або “proof-of-stake”.

Алгоритм Proof of Work застосовується в блокчейні Bitcoin. PoW в блокчейні перевіряє обчислення, які генеруються в процесі створення нового блоку. Тут використовується наступна модель: блок визнається вірним і закритим, за умови, якщо значення його хешу менше ніж шуканий майнерами підпис. Тобто, певний криптографічний шифр показує справжність блоку. В якості «ревізорів», котрі перевіряють справжність блоку, виступають ноди. Зараз в мережі Bitcoin блок створюється протягом 10 хвилин. У цей момент і відбувається пошук підпису. А вже перевірка відбувається миттєво. Алгоритм часто критикується через те, що для його роботи необхідні великі обчислювальні потужності.

Інтернет речей – одна з найпопулярніших концепцій в сучасній футурології. І більш того, одна з тих небагатьох, що вже перестають бути концепціями і втілюються в життя. Згідно з найбільш поширеним формулюванням, Інтернет речей – це концепція обчислювальної мережі фізичних предметів які оснащені технологіями для взаємодії один з одним.

Proof of Stake – алгоритм не такий ресурсномісткий, і в цілому, дешевший PoW. Блокчейн криптовалюти Ethereum здійснює перехід з PoW на PoS. Якщо в Proof of Work на перший план виходить обчислювальна потужність, то в Proof of Stake роль відіграє баланс [32]. Здійснення і підтвердження транзакцій відбуватиметься без активної участі обчислювальної техніки. В ідеалі, всі власники криптовалюти на блокчейні з PoS виступатимуть в ролі інвесторів. Однак у алгоритму є істотні недоліки – можливе проведення дублюючих транзакцій. В таблиці 2.3 перераховано недоліки і переваги розвитку блокчейн.

Таблиця 2.3 – Недоліки і переваги Proof of Stake

Переваги	Недоліки
Децентралізація – учасники мережі рівні між собою і можуть обмінюватися даними безпосередньо;	Масштабованість
Надійність – виключена підміна даних і хакерські атаки, так як використовуються спеціальні зашифровані ключі;	Шахрайство – передача даних блокчейн відбувається необоротно.
Прозорість – всі блоки доступні для публічного перегляду. Можна перевірити пройдений шлях для будь-якої транзакції;	Атака 51% – якщо в блокчейні біткоіна 51% обчислювальних потужностей будуть належати одному пристрою, то цілісність порушиться.

IOTA не схожа на біткоін або ефір, так як вона фактично не використовує блокчейн [11]. Ця платформа використовує спеціальний журнал Tangle, що працює на основі DAG - спрямованого ациклічного графу. У блокчейн Bitcoin або Ethereum все тримається на блоках, куди і записується інформацію про транзакції. У Tangle IOTA блоків немає, а транзакції там пов'язані за своєю особливою схемою: кожна нова виникла транзакція (назвемо її A) підтверджує дві старі (B і C). Верифікація також може відбуватися побічно - виникла транзакція D, яка підтвердила A і умовну Z. Але при цьому вона побічно підтвердила B і C. Зображено на рисунку 2.7 схема роботи IOTA.

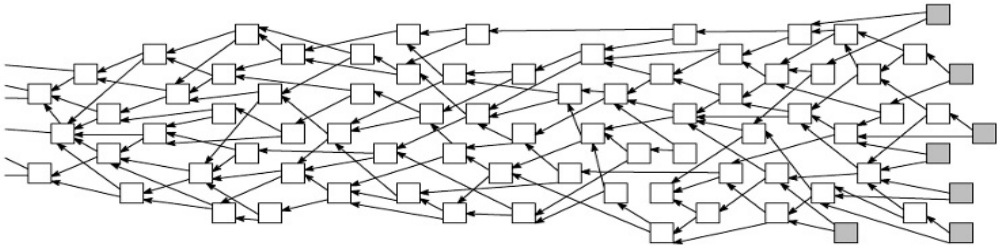


Рисунок 2.7 – Схема роботи IOTA

Така система формує цілу павутину верифікації, що захищає мережу від подвійного витрачання і змушує працювати з більш високою ефективністю. Тобто, чим більше транзакцій в мережі, тим швидше вони будуть оброблятися.

IOTA як криптовалюта має свої переваги. Головні з них - це відсутність комісій, можливість проводити мікротранзакції і висока швидкість операцій. Всі ці переваги забезпечені Tangle, який багато негласно охрестили блокчейном 3.0. Хоча фактично він і не є блокчейном, так як блоки в ньому не формуються. Переваги та недоліки зображені у таблиці 2.4.

Таблиця 2.4 – Переваги та недоліки IOTA

Переваги	Недоліки
Немає проблем з масштабністю - чим більше транзакцій, тим краще для мережі	Слабка підтримка на біржах і обмежена функція монети
Відсутність комісій за транзакції	Відсутність надійного гаманця для зберігання IOTA
Нема майнингу	Можлива власна атака 51% в мережі
Унікальність платформи.	–

На відміну від блокчейн Tangle може містити конфліктуючі транзакції, а всіх вузлів мережі не обов'язково мати однакову копію Tangle. Головні завдання повного вузла мережі наступні:

1. Транслювати транзакції з коректним PoW і електронним підписом.

2. Коректно вибирати транзакції для підтвердження при створенні нової (tip selection algorithm).

3. Визначати підтверджена чи транзакція (підтверджена якщо її продовжують побічно підтверджувати нові транзакції), це перевіряється шляхом повторного запуску tip selection algorithm.

Поточна реалізація Tangle в IOTA вимагає наявності довіреної сторони - координатора (the coordinator), для забезпечення надійності. Оскільки,

передбачається, що поточна версія не здатна працювати у відкритому середовищі, утворюючи повністю розподілену однорангові з'єднання [42]. Плани з розвитку IOTA включають позбавлення від необхідності довіреної сторони (координатора), передбачається така робота мережі, що жоден вузол не має уявлення про повну стані Tangle. У поточній бета-версії The Coordinator робить повний огляд і контроль всіх транзакцій в мережі, а інші вузли орієнтуються на нього. Таким чином The Coordinator є тимчасовим центральним органом необхідним для захисту від атак, який підтримується спільнотою IOTA.

IOTW приймає унікальний підхід до вирішення проблеми, використовуючи програмне і апаратне забезпечення [12].

Програмна частина включає в себе як перехід від систем консенсусу докази роботи (PoW), так і з біткойн і ефіріум. Варіанти докази частки (PoS), що використовуються так званими блокчейн третього покоління, такими як EOS і Tron.

Базові принципи IoT. Інтернет речей грунтується на трьох базових принципах. По-перше, повсюдно поширену комунікаційну інфраструктуру, по-друге, глобальну ідентифікацію кожного об'єкта і, по-третє, можливість кожного об'єкта відправляти і отримувати дані за допомогою персональної мережі або мережі Інтернет, до якої він підключений.

Однак отримати оптимальний баланс між безпекою та швидкістю транзакцій важко. Йоти проекту, наприклад, ці питання вирішуються шляхом фактично відмовившись від блокчейн, деякі стверджують, для власного плутати систему, засновану на спрямований ациклічний граф, або Tangle, де операції повинні бути затверджені в вузол, що і підтверджує попередні угоди.

Мережеве програмне забезпечення розроблене таким чином, що для підключення інтелектуальних пристроїв до мережі не потрібно ніякого спеціального устаткування.

В таблиці 2.5 приведено порівняння алгоритмів консенсусу для мереж Інтернет речей.

Таблиця 2.5 – Порівняння алгоритмів консенсусу для Інтернет речей

	IOTW	IOTA
Спеціальне обладнання / CPU необхідний	Не потрібно	Повинні працювати на власних CPU / платах
Mining	+	–
Миттєві транзакція	+	–
Гнучкість системи винагороди	+	–
Витрати на будівництво системи	1/100c	Дорого
Безпека	Висока	Низька
Масове партнерство по розгортання в напівпровідникові, IoT апаратній індустрії	+	–
Застосування	Децентралізована платформа електронної комерції, Платіжна система, Великий збір даних, Інше використання підприємств	Мікро-транзакція
Token Utility	Оплата, мікро-транзакція, купівля великих даних	Мікро-транзакція

Проте, для того, щоб підвищити ефективність видобутку, IOTW цифрова система (DPS) - оптимізований для виконання на низькій потужності для запуску обчислень, щоб знайти хеші. Пристрої інтелектуального аналізу даних повинні

знайти хеш зашифрованого шаблону BL blockchain, переданого йому довіреними вузлами мережі IOTW.

Мережа здатна здійснювати десятки тисяч транзакцій в секунду (tps) і націлена на досягнення одного мільйона tps. Будь-прилад встановлений з більш дешевим і більш з низьким енергоспоживанням уламок буде автоматично на мережу IOTW. Пристроєм можна управляти, контролювати і обмінюватися даними.

Proof-of-assignment – це голодний, неефективний вільний для всіх, де всі вузли конкурують в масовому розтраті ресурсів, в той час як PoS винагороджує тих, хто з найбільшою часткою токенів в можливу шкоду для більш дрібних власників. Доказ присвоєння і стимулюючі ефекти об'єднані, щоб забезпечити ліквідність, заохочувати, скасувати відходи і зупинити потенційне спотворення мережі на користь найбільших власників акцій.

Переваги алгоритму «Proof of Assignment». По-перше, побутові пристрої можуть бути використані для майнингу, пропонуючи реалістичне вирішення проблем масштабованості і відкладеної обробки транзакцій, з якими стикаються сучасні популярні криптовалютні мережі.

По-друге, власники пристроїв можуть самі планувати, коли їх пристрої будуть брати участь в майнингу.

По-третє, власник пристрою може охоче ділитися або продавати дані, згенеровані і оброблені його пристроєм для майнингу криптовалют, тому що дані можуть бути корисні організаціям, які займаються, наприклад, дослідженнями ринку.

2.3 Структура системи ідентифікації об'єктів

У структурній схемі зображено технології працюють в тандемі один з одним. Діаграма є спрощеною, оскільки процеси, які не відбуваються паралельно весь час, а також будуть дублювання. Сенсори отримують шар над платформою Інтернет речей, тоді як дані з зовнішніх джерел беруться через платформу блокчейн. Навіть обмін даними в мережі Інтернет речей може статися через блокчейн, щоб забезпечити відстеження та запис усіх транзакцій. Багато мереж Інтернет речей можуть обмінюватися даними, тоді як потужність буде покращуватися за допомогою більшої кількості даних (рисунок 2.8). Це не тільки допоможе підвищити ефективність, але й допоможе компаніям забезпечити краще обслуговування клієнтів.

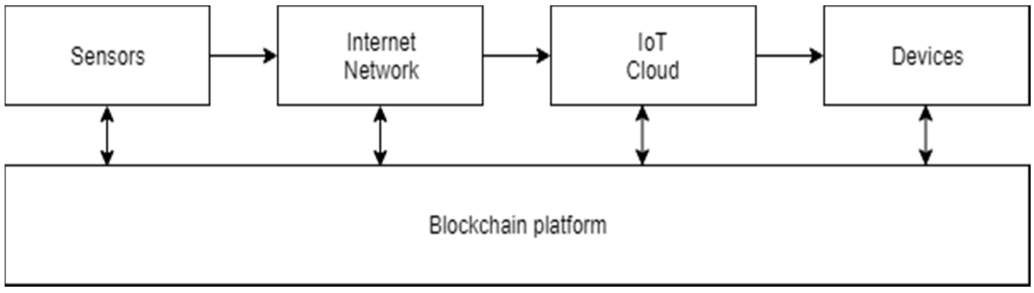


Рисунок 2.8 – Структурна схема ідентифікації об'єктів

Спільне використання цих трьох технологій призведе до вищої складності, тоді як проблеми конфіденційності продовжують переслідувати ряд бізнес та споживчих додатків. Існує проблеми під час сумісності через відсутність протоколів та стандартів. Деякі програми, ймовірно, будуть оформлені на неперевірених територіях, і вони можуть мати юридичні чи юридичні проблеми.

Інтернет речі та блокчейн доповнюють один одного і потенційно можуть усунути деякі недоліки цих технологій, коли вони виконуються окремо. Ідея цих технологій, що працюють в тандемі, не нова. Конвергенція відбудеться швидше, ніж передбачалося.

Неможлива підміна сенсора, вона полягає у тому що кожен сенсор має свій власний ідентифікаційний номер, який записується у базу даних блокчейн зображено на рисунку 2.9. Якщо використовує сенсор для передачі даних і хтось підмінив його то дані не будуть зчитуватися. При підключені цього датчика було записано його ідентифікаційний номер і при підміні він не співпадає з ідентифікаційним номером, який був записаний у базу даних.

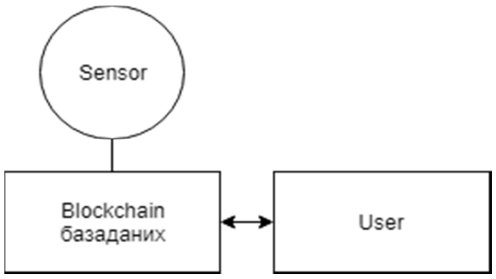


Рисунок 2.9 – Структурна схема сенсора

Архітектура Інтернету речей передбачає наявність таких функціональних рівнів: мережа датчиків, шлюз, управління, додаток. Оскільки нижній рівень складається з датчиків, сенсорів, то виникає необхідність в "особливих" протоколах для забезпечення взаємодії цих пристроїв один з одним і верхніми рівнями [28]. Стандартні прикладні протоколи не підходять через їх непристосованість до умов мережі Інтернет речей. Датчик з невеликою пам'яттю, вимірює фізичні параметри в режимі реального часу, найчастіше в умовах низького енергозабезпечення. Результати вимірювань обробляються сенсорним вузлом і передаються на сервер. Обсяг інформації, що формується одним сенсорним вузлом, порівняно невеликий, проте більшість сервісів Інтернету речей побудовано на принципі обробки інформації від безлічі вузлів, що принципово відрізняється від архітектури, прийнятих в класичних мережах, типу абонент - вузол зв'язку для телефонії, клієнт-сервер для передачі даних (зображено на рисунку 2.10).

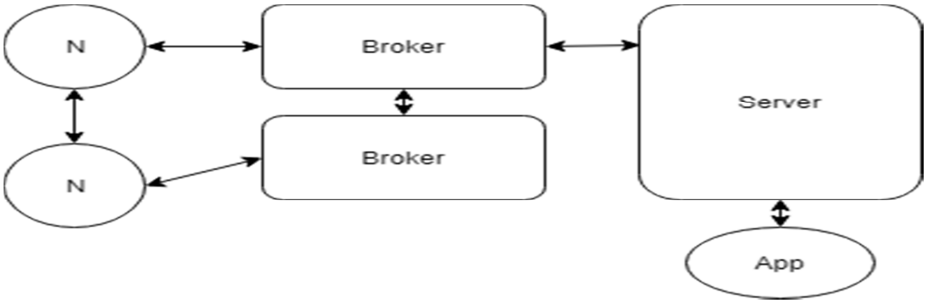


Рисунок 2.10 – Топологія мережі

Реалізовується реєстрація сенсорного вузла; конфігурація і налаштування вузлів; передача і розподіл інформації і т. д. На цьому сегменті мережі можуть використовуватися два наступних протоколу (зображено на рисунку 2.11).

XMPP - розширений протокол обміну повідомленнями та інформацією про присутність. XMPP в Інтернет речах забезпечує простий спосіб адресації пристроїв. Для ідентифікації користувачів використовуються запам'ятовуються ідентифікатори JID, за форматом схожі на адреси електронної пошти. У протоколі XMPP застосовується текстовий формат XML. В якості транспорту використовується протокол TCP. XMPP підтримує різні комунікаційні моделі (запит-відповідь, публікація-підписка і інші).

Адресація XMPP особливо зручна у випадках, коли дані передаються між віддаленими, найчастіше незалежними точками, наприклад в разі взаємодії двох абонентів. За допомогою XMPP, можливе підключення домашнього термостата до Web-серверу для отримання до нього доступу з телефону. Перевагами протоколу є також безпека і масштабованість, що робить його ідеальним для додатків Інтернету речей з орієнтацією на споживача.

Для мереж з обмеженими ресурсами, низьким енергоспоживанням більше підійде протокол COAP.

Повідомлень, які використовуються протоколом COAP, не так багато, деякі мають на увазі запити-відповіді: GET, PUT, HEAD, POST, DELETE, CONNECT. Додатки користувача використовують повідомлення для управління і спостереження за ресурсом. За запитом встановлюється прапор спостереження, і сервер продовжує відповідати після того, як первинне повідомлення було

передано. Це дозволяє серверам організовувати потокову передачу змін станів датчиків.

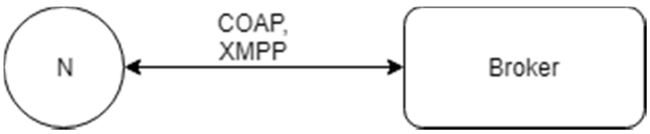


Рисунок 2.11 – Сегмент мережі від вузла до брокера

На ділянці мережі між сенсорним вузлом і брокером для забезпечення їх зв'язку, з метою реєстрації та конфігурації вузлів, а також для передачі інформації найчастіше застосовуються два протоколи - XMPP і COAP. Вибір конкретного протоколу залежить від умов реалізованої мережі. XMPP застосовується в системах освітлення і клімату, а також використовується для адресації пристроїв в невеликих персональних мережах. COAP застосовується для пристроїв з обмеженими ресурсами і для мереж з низьким енергоспоживанням. Відомо застосування протоколу в системах датчиків температури та інших датчиків розумного будинку.

Наступним відрізком топології є ділянку мережі, де ключовим елементом є брокер. З огляду на описане раніше призначення брокера в мережі Інтернету речей, можна виділити завдання, які повинні вирішуватися на даній ділянці: збір та агрегація даних; організація черг повідомлень; розподіл і зберігання інформації "до запитання" (зображено на рисунку 2.12).

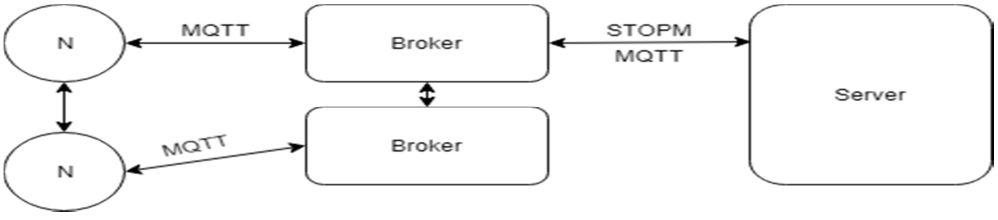


Рисунок 2.12 – Сегмент мережі Broker

Для завантажених мереж з великою кількістю пристроїв раціональніше застосовувати протокол, що знижує навантаження на канал за рахунок організації черг, - протокол MQTT.

Протокол MQTT - призначений для телеметрії і дистанційного моніторингу. Використовується для обміну сполучення між пристроями за принципом "видавець-передплатник", дає їм змогу надсилати і отримувати дані при виникненні деякої події. MQTT - бінарний протокол обміну повідомленнями, що має на увазі публікацію / підписку, що працює з використанням транспорту TCP.

STOMP - простий протокол обміну повідомленнями, що передбачає широку взаємодію з багатьма мовами, платформами і брокерами. Даний протокол підходить під шаблон "видавець-передплатник" і за допомогою повідомлень SEND, SUBSCRIBE, UNSUBSCRIBE, BEGIN, COMMIT, ABORT, ACK, NACK, DISCONNECT організовує зв'язок з брокером за методом "запит-відповідь" [35].

Протокол схожий на HTTP, використовує транспорт TCP, є простим текстовим протоколом, що дозволяє клієнтам STOMP спілкуватися з будь-яким брокером повідомлень, що підтримує даний протокол. Таким чином, це спосіб взаємодії, розроблений для обміну повідомленнями між платформою, описуваної на одній мові програмування, і клієнтом, програмне забезпечення якого розроблено на іншій мові. Підтримує велику кількість сумісних клієнтських бібліотек, пов'язаних мов.

Для забезпечення роботи брокера в мережі Інтернет речей можливе використання обох протоколів: MQTT і STOMP. Протокол MQTT забезпечує "наскрізну" зв'язок, як від брокера до сенсорних вузлів, так і від брокера до сервера, тоді як протокол STOMP орієнтований тільки на взаємодію брокера з сервером [34].

Розглянуто платформу Microsoft Azure, проведено порівняння, переваги та тестування [39] з іншими платформами. Проведено порівняння алгоритмів консенсусу Інтернет речей IOTA і IOTW та консенсусу технології бокчейн PoW, PoS, PoA.

Розроблена структура системи ідентифікації об'єктів на основі технології блокчейн.

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ АЛГОРИТМИ ІДЕНТИФІКАЦІЇ ІНТЕРНЕТ РЕЧЕЙ

3.1 Програмна реалізація алгоритму ідентифікації об’єктів Інтернет речей на основі технології блокчейн

Blockchain - це база даних. База даних - це організована колекція даних, структура даних, яка зберігає дані (рисунок 3.1).

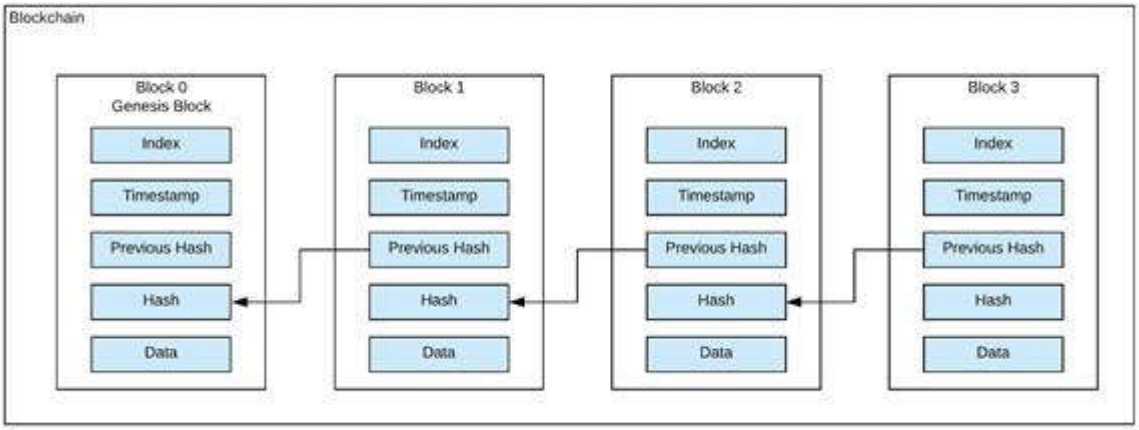


Рисунок 3.1 – Структура blockchain

Перший блок - це спеціальний блок: блок генезу. Блок Genesis - це єдиний блок, який не має попередніх блоків і не містить даних. блоків.

Створюємо функцію Block) у якій оголошуємо параметри і передаємо їхні параметри (рисунок 3.2:

– Index;
– TimeStamp;
– PreviousHash;
– Hash;
– Data.

```
public class Block {
    public int Index { get; set; }
    public DateTime TimeStamp { get; set; }
    public string PreviousHash { get; set; }
    public string Hash { get; set; }
    public string Data { get; set; }
    public Block(DateTime timeStamp, string previousHash, string data) {
        Index = 0;
        TimeStamp = timeStamp;
        PreviousHash = previousHash;
        Data = data;
        Hash = CalculateHash(); }
    public string CalculateHash() {
        SHA256 sha256 = SHA256.Create();
        byte[] inputBytes = Encoding.ASCII.GetBytes($"{TimeStamp}-{PreviousHash ?? ""}-{Data}");
        byte[] outputBytes = sha256.ComputeHash(inputBytes);
        return Convert.ToBase64String(outputBytes); }
}
```

Рисунок 3.2 – Створення класу Block

Клас Block містить метод CalculateHash у якому обчислюється hash даних та конвертується до Base64.

Оголошуємо клас Blockchain який містить список об’єктів класу блока. Цей клас містить методи:

– InitializeChain - ініціалізує список блоків;
– AddGenesisBlock - добавляє новий block у список;
– GetLatestBlock - повертає остальній блок у списку;
– CreateGenesisBlock - створює новий block у список;
– AddBlock - додавання нового блоку у ланцюжок.

Створюємо екземпляр нового блочного шаблону, який зображено на рисунку 3.4.

```
public class Blockchain {
    public IList<Block> Chain { set;  get; }
    public Blockchain() {
        InitializeChain();
        AddGenesisBlock(); }
    public void InitializeChain(){
        Chain = new List<Block>();
    }
    public Block CreateGenesisBlock() {
        return new Block(DateTime.Now, null, "{}");
    }
    public void AddGenesisBlock() {
        Chain.Add(CreateGenesisBlock());
    }
    public Block GetLatestBlock() {
        return Chain[Chain.Count - 1];
    }
    public void AddBlock(Block block) {
        Block latestBlock = GetLatestBlock();
        block.Index = latestBlock.Index + 1;
        block.PreviousHash = latestBlock.Hash;
        block.Hash = block.CalculateHash();
        Chain.Add(block); }
}
```

Рисунок 3.3 – Створення класу Blockchain

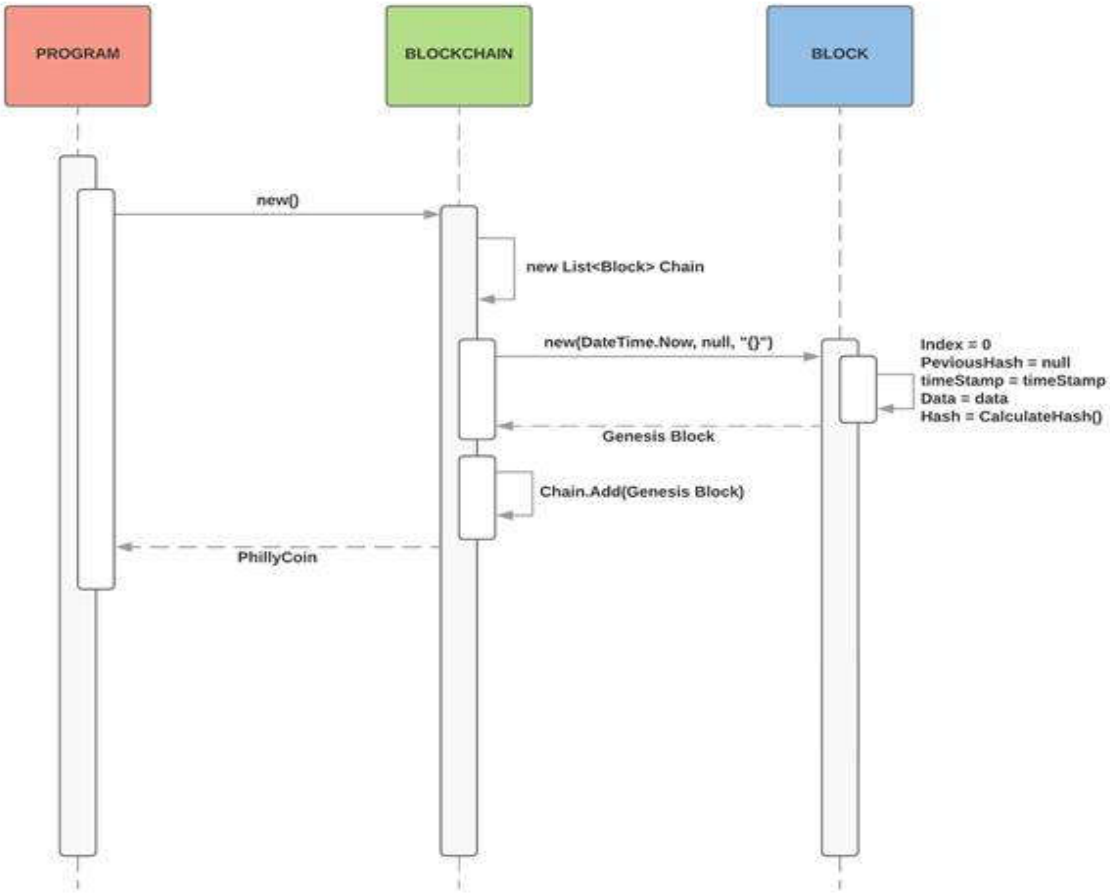


Рисунок 3.4 – Діаграма послідовності

Можемо додати до екземпляру блочного шаблону нові блоки, які зображено на рисунку 3.5.

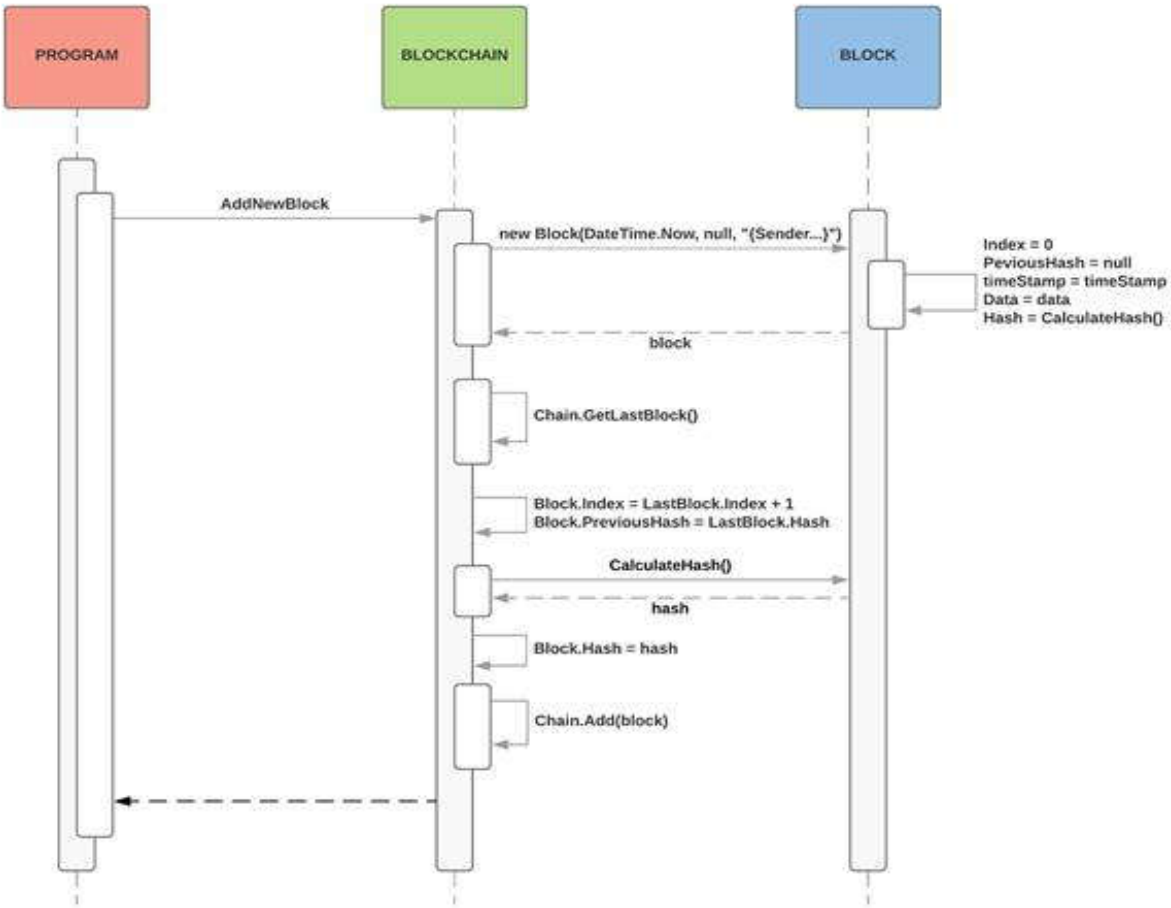


Рисунок 3.5 – Діаграма послідовності з додаванням блоків

Однією з переваг використання технології блокчейн є захист даних. Захист даних означає, що втручання у старі дані та зміни методу захисту нових даних забезпечується як криптографічним методом, так і не централізованим зберіганням самих даних. Однак блокчейн - це просто структура даних, в якій дані можна легко змінити подібно.

Спосіб підтвердження даних описано у функції IsValid зображено на рисунку 3.6.

```
public bool IsValid() {
    for (int i = 1; i < Chain.Count; i++) {
        Block currentBlock = Chain[i];
        Block previousBlock = Chain[i - 1];
        if (currentBlock.Hash != currentBlock.CalculateHash()) {
            return false;
        }
        if (currentBlock.PreviousHash != previousBlock.Hash) {
            return false;
        }
    }
    return true;
}
```

Рисунок 3.6 – Функція IsValid

Створюємо функцію IsValid як перевіряє валідність Blockchain. У даній функції перевіряється еквівалентність hash`s.

Метод IsValid перевіряє кожен блок хеш, щоб побачити, чи блок змінений. Хеш попереднього блоку, щоб побачити, чи блок змінений і перерахований. Викликаємо IsValid перед тим, як змінити дані та після того, як дані змінилися. Якщо дані змінилися виводимо повідомлення із зміною даних. Якщо перерахувати усі хеші всіх поточних блоків та наступних. Тоді як усі блоки будуть передані, перевірка буде передана. Проте вона передана лише на одному вузлі оскільки блокчейн - це децентралізована система.

Виконуємо та тестуємо функцію Main (зображено на рисунках 3.7, 3.8). У ній створюємо об’єкт Blockchain.

```
static void Main(string[] args) {
    Blockchain phillyCoin = new Blockchain();
    phillyCoin.AddBlock(new Block(DateTime.Now, null, "{sender:Henry,receiver:MaHesh,amount:10}"));
    phillyCoin.AddBlock(new Block(DateTime.Now, null, "{sender:MaHesh,receiver:Henry,amount:5}"));
    phillyCoin.AddBlock(new Block(DateTime.Now, null, "{sender:Mahesh,receiver:Henry,amount:5}"));
    Console.WriteLine(JsonConvert.SerializeObject(phillyCoin, Formatting.Indented));
    Console.WriteLine($"Is Chain Valid: {phillyCoin.IsValid()}");
    Console.WriteLine($"Update amount to 1000");
    phillyCoin.Chain[1].Data = "{sender:Henry,receiver:MaHesh,amount:1000}";
    Console.WriteLine($"Is Chain Valid: {phillyCoin.IsValid()}");
    Console.WriteLine($"Update hash");
    phillyCoin.Chain[1].Hash = phillyCoin.Chain[1].CalculateHash();
    Console.WriteLine($"Is Chain Valid: {phillyCoin.IsValid()}");
    Console.WriteLine($"Update the entire chain");
    phillyCoin.Chain[2].PreviousHash = phillyCoin.Chain[1].Hash;
    phillyCoin.Chain[2].Hash = phillyCoin.Chain[2].CalculateHash();
    phillyCoin.Chain[3].PreviousHash = phillyCoin.Chain[2].Hash;
    phillyCoin.Chain[3].Hash = phillyCoin.Chain[3].CalculateHash();
    Console.WriteLine($"Is Chain Valid: {phillyCoin.IsValid()}");
    Console.ReadKey();
}
```

Рисунок 3.7 – Виконання функції Blockchain

Рисунок 3.8 – Тестування Blockchain

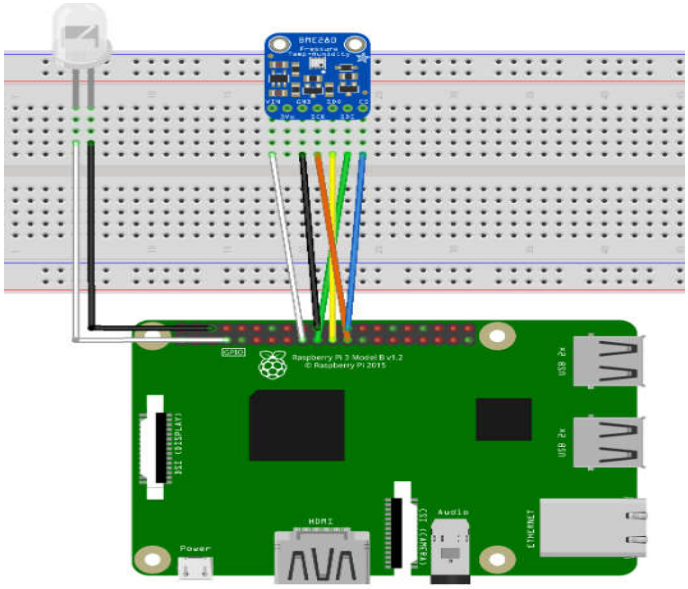
Додаємо нові блоки і перевіряємо їх на валідність і виводимо їх дані у форматі JSON. При додаванні першого блоку IsValid успішна. Оновлюємо суму 1000. При додаванні другого блоку IsValid видає значення false. Оновлюємо hash і додаєм третій блок і перевіряєм на IsValid і видає значення false. Після оновлення усіх Chain перевіряєм на валідність IsValid і отримуємо значення true.

3.2 Реалізація публічного блокчейну

Розгортання IoT-концентратора та клієнта Raspberry Pi. Використано службу для читання даних з обладнання Thunderboard React (рисунок 3.9). Датчик Thunder має суперконтролер датчиків, що використовуються у цьому зразку, і це відображається в схемі обміну повідомленнями, яку він передає в концентратор IoT. Єдиними властивостями, які використовуються з корисної завантаженості, є пристрій, температура та вологість. Можемо мати своїх клієнтів (реальних або імітаційних) додати ту саму схему або доставити іншу схему, яка включає ці значення.

Таблиця 3.1 – Схема підключення датчика Thunder

Thunder	Raspberry Pi
LED VDD (Pin 5G)	GPIO 4 (Pin 7)
LED GND (Pin 6G)	GND (Pin 6)
VDD (Pin 18F)	3.3V PWR (Pin 17)
GND (Pin 20F)	GND (Pin 20)
SCK (Pin 21F)	SPI0 SCLK (Pin 23)
SDO (Pin 22F)	SPI0 MISO (Pin 21)
SDI (Pin 23F)	SPI0 MOSI (Pin 19)
CS (Pin 24F)	SPI0 CS (Pin 24)



Зображення 3.9 – Підключення датчика до Raspberry Pi

На порталі Azure відкриваємо концентратор IoT і натисніть «Endpoints». На рисунку 3.10 зображенно IoT-hub.

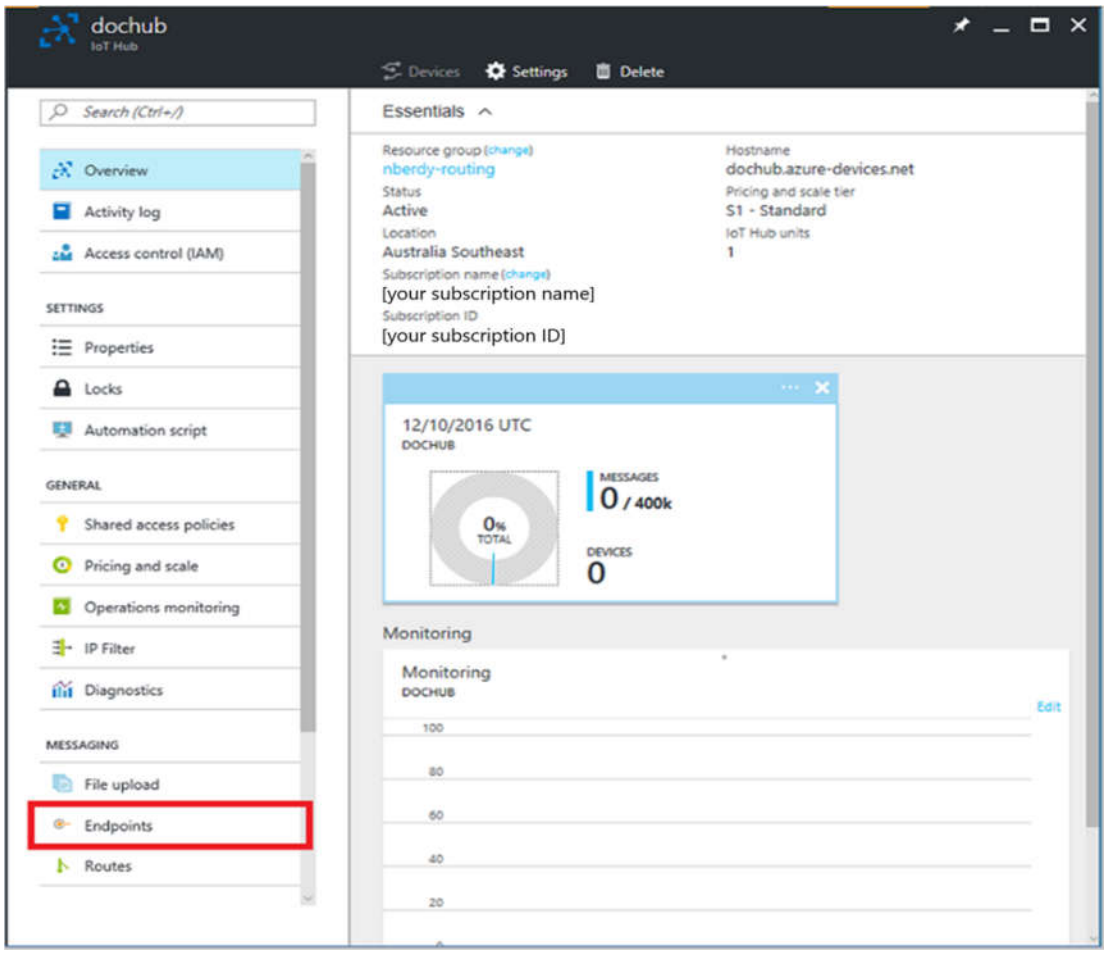


Рисунок 3.10 – Портал Azure IoT hub

На панелі кінцевих точок натисніть Add у верхній частині, щоб додати чергу до свого концентратора IoT. Назвіть кінцеву точку iotingest і скористайтеся випадаючими списками, щоб вибрати чергу Service Bus, простір імен Service Bus, в якому знаходиться ваша черга, та ім'я вашої черги. Коли ви закінчите, натисніть "Зберегти" унизу. Зображено на рисунку 3.11.

Тепер натисніть маршрути у вашому IoT-концентраторі. Клацніть Додати у верхній частині леза, щоб створити правило маршрутизації, яке направляє повідомлення до черги, яку ви щойно додали. Виберіть DeviceTelemetry як джерело даних. Введіть умову, яка є релевантною для вашого розумного контракту, і виберіть чергу, яку ви щойно додали як спеціальну кінцеву точку, як кінцеву точку правила маршрутизації.

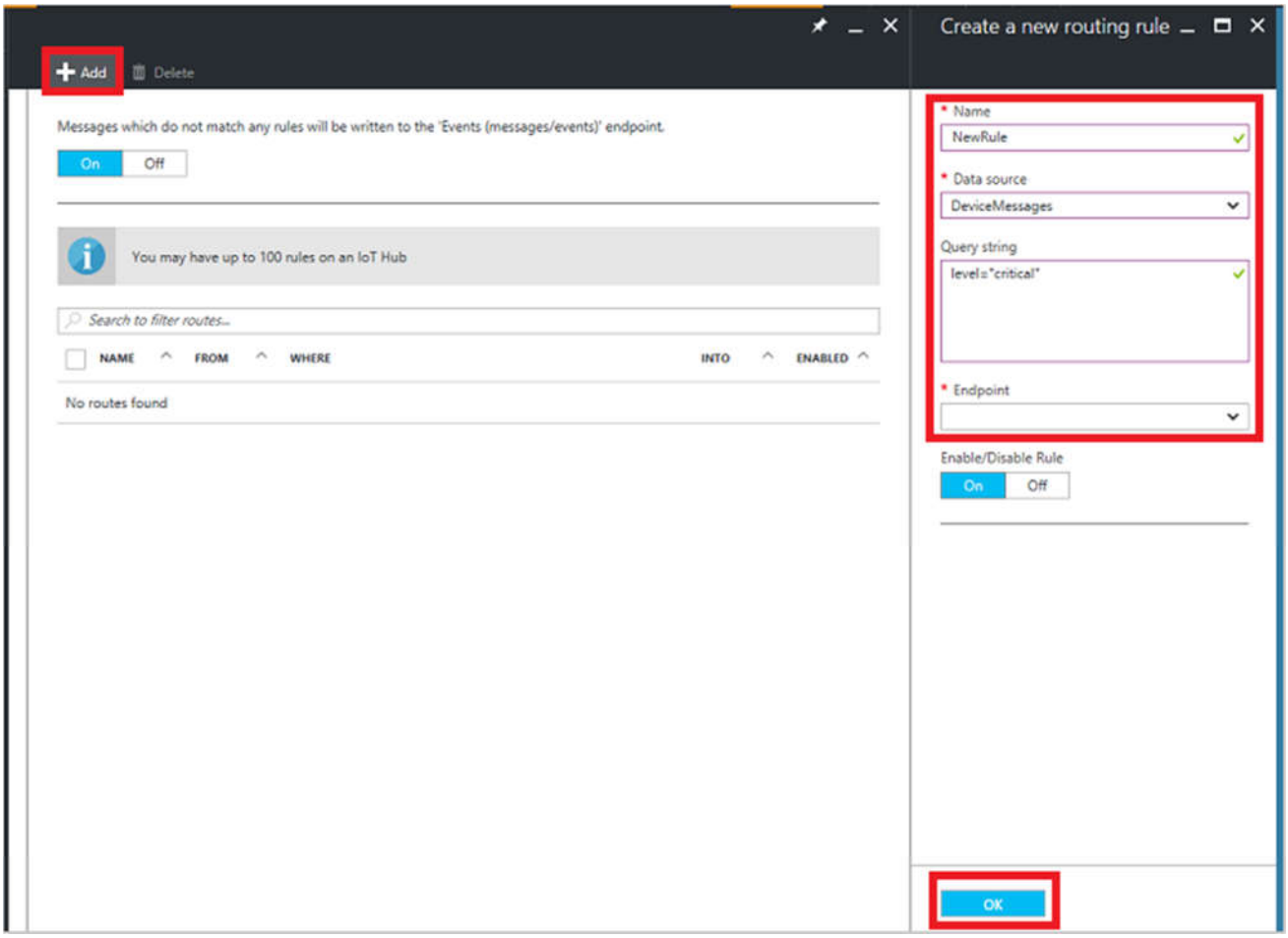


Рисунок 3.11 – Додавання кінцевої точки

Перевіряємо, що альтернативний маршрут встановлений у положення ON. Це значення є конфігурацією за замовчуванням для концентратора IoT. Зображено на рисунку 3.12.

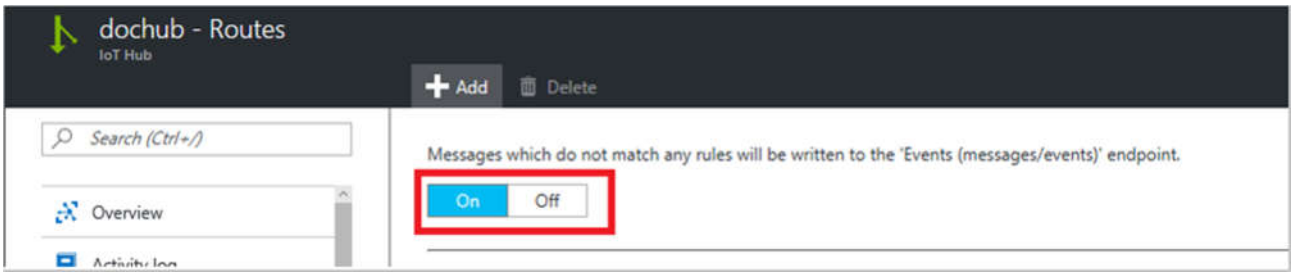


Рисунок 3.12 – Додавання маршруту

IoT-концентратор тепер налаштований на ідентифікацію повідомлень пристрою, що відповідає критеріям, що відповідають вашому розумному контракту, і доставляти їх до сервісної шини з ім'ям iotingest.

Logic App програма, яка виконує дві дії: Перша пов'язано з створенням користувача, а інше пов'язано з доставкою телеметричних даних. Логічна програма вказана на Service Bus, який розміщений IoT-концентратором. За замовчуванням логічна програма налаштована так, щоб запускати кожну хвилину для обробки будь-яких нових повідомлень, які доставляються концентратором IoT. Коли з'являється нове повідомлення, логічна програма створює повідомлення відповідно до створення або створення функції "IngestTelemetry".

Логічне додаток створює нове повідомлення про дію, відповідне кожному отриманому від концентратора IoT. Перед додаванням новоствореного повідомлення до Service Bus програма Logic спершу робить виклик SQL до процедури збереження Get Contract Instance Info For DeviceId в Workbench db. Ця збережена процедура приймає DeviceID як вхідний сигнал (знову взятий з повідомлення службової шини IoT) і повертає дані для конкретного примірника контракту, до якого пристрій було додано за допомогою ролі "Device". (Кожен пристрій може бути зіставлений лише з одним елементом із цією роллю). Після того, як це повідомлення було розміщено на Azure Blockchain Workbench Service

Bus, робота збирається, а відповідні дії виконуються для виконання запиту на блокчейн (додаток А).

У середовищі Azure Blockchain бази даних виберіть редактор запитів. Зображено на рисунку 3.13.

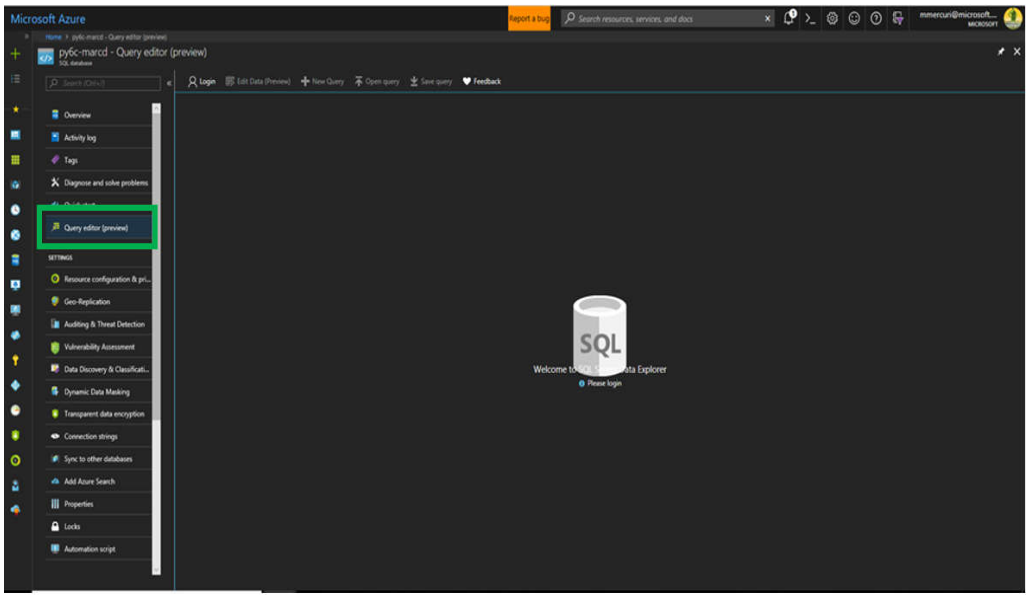


Рисунок 3.13 – Середовище Azure Blockchain

Натисніть login і ведіть свої облікові дані бази даних зображено на рисунку 3.14. Ім'я користувача буде 'dbadmin', а пароль - той, який ви надали під час встановлення Azure Blockchain Workbench.

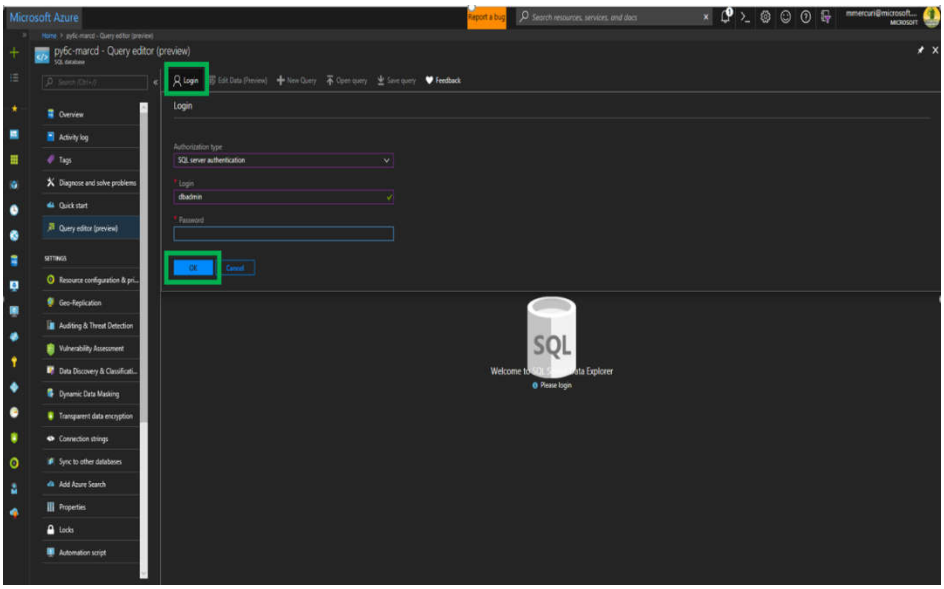


Рисунок 3.14 – Azure Blockchain Login

Зовнішній ідентифікатор таблиці User використовується для зберігання ідентифікатора пристрою. Визначте ідентифікатор пристрою свого пристрою, який буде надіслано з повідомленням телеметрії. У вікні запиту введіть та виконайте наступний SQL.

Оновити [User] Set External Id = '<your device id here>' where EmailAddress = '<insert email address here>.

Для того, щоб оцінювався вміст, він повинен бути перетворений з рядка Base64 string. Натисніть у полі змісту, виберіть Expression і введіть - json (base64ToString (triggerBody ()? ['ContentData'])) зображено на рисунку 3.15.

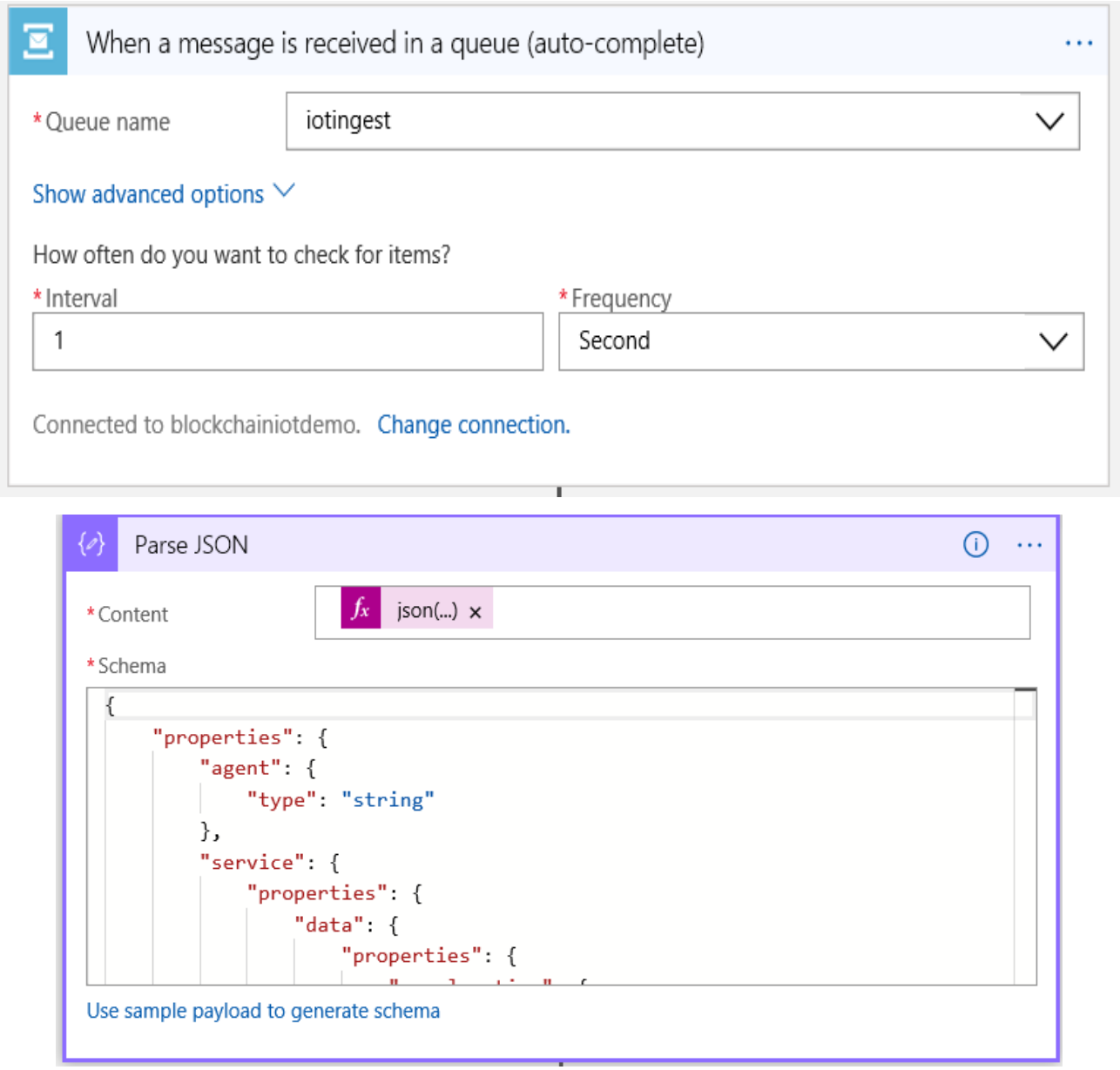


Рисунок 3.15 – Parse JSON

Наступний текст являє собою схему, що використовується клієнтом і відображає дані датчиків, доступні з пристрою React Thunderboard .

Натисніть "More" та додайте нову дію. Виберіть " SQL Stored Procedure". Виберіть збережену процедуру під назвою "GetContractInfoForDeviceID". Використовуючи динамічні властивості, виберіть "deviceId (зображено на рисунку 3.16).

Рисунок 3.16 – Додавання нової дії

Натисніть посилання "More" та виберіть Додати дію. Виберіть дію Parse JSON. У полі Зміст виберіть елемент ResultSets для вмісту. Натисніть посилання " Add step" та оберіть " Initialize Variable".

Ця нова змінна буде використовуватися для вказування унікального ідентифікатора для запиту, який можна відстежити в Insights програми.

Встановіть властивість Name на RequestId. Встановлюємо властивість типу у рядку. Зображено на рисунку 3.17.

У полі Value використовуйте вікно Dynamic Content, оберіть Expression і введіть guid ().

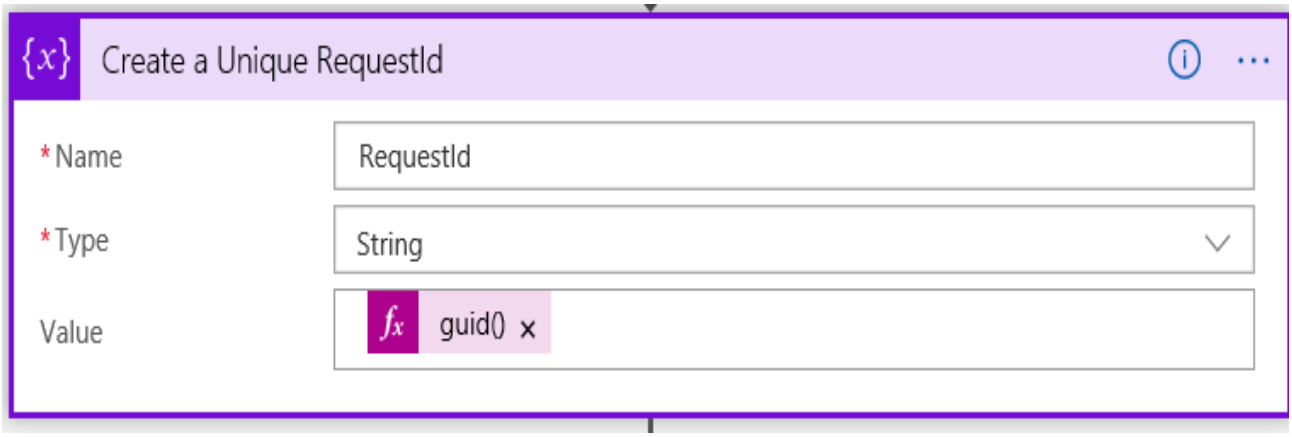


Рисунок 3.17 – Create a Unique RequestId

Далі ви хочете визначити час Unix Epoch, щоб включити його як мітку часу для повідомлення. Натисніть посилання "Add step" та оберіть " Initialize Variable". Встановіть властивість Name на TicksNow. Встановіть властивість Type у ціле число. Виберіть поле властивості Value, а потім у вікні Dynamic Content введіть наступне на вкладці Expression - ticks (utcNow ()) зображено на рисунку 3.18.

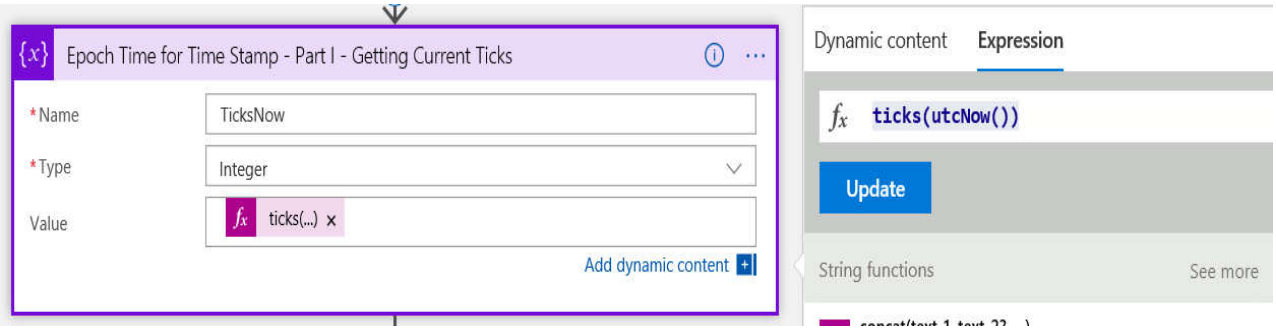


Рисунок 3.18 – Встановлення ticks(utcNow())

Натисніть посилання "Add step" та оберіть " Initialize Variable". Встановіть властивість Name на TicksNow. Встановіть властивість Type у ціле число. Клацніть поле властивості Value, а потім у вікні Dynamic Content введіть наступне на вкладці Expression - ticks ('1970-01-01') зображено на рисунку 3.19.

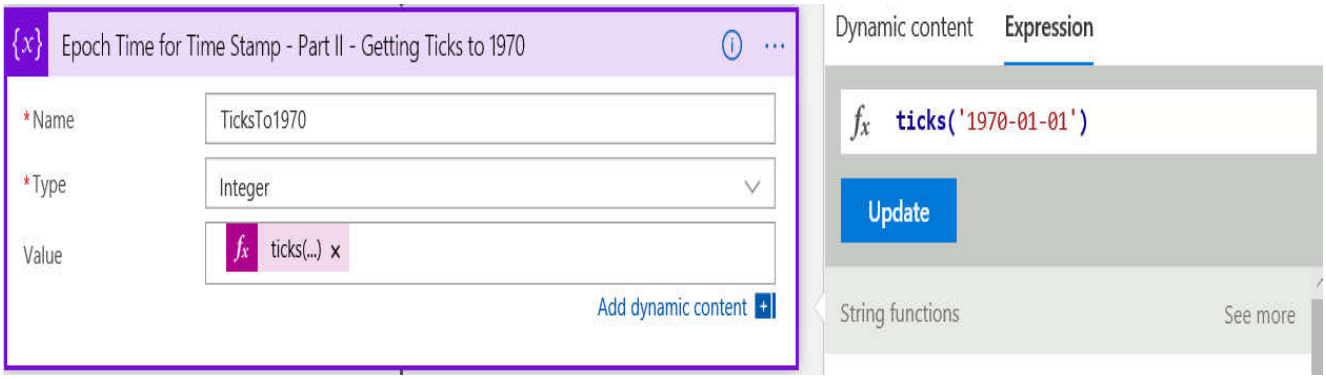


Рисунок 3.19 – Встановлення ticks ('1970-01-01')

Натисніть посилання "Додати крок" та оберіть "Ініціалізація змінної". Встановіть властивість Name на Timestamp. Встановіть властивість Type у ціле число. Клацніть поле властивості Value, а потім у вікні Dynamic Content введіть наступне на вкладці Expression - div (sub (змінні ('TicksNow'), змінні ('TicksTo1970')), 10000000) зображено на рисунку 3.20.

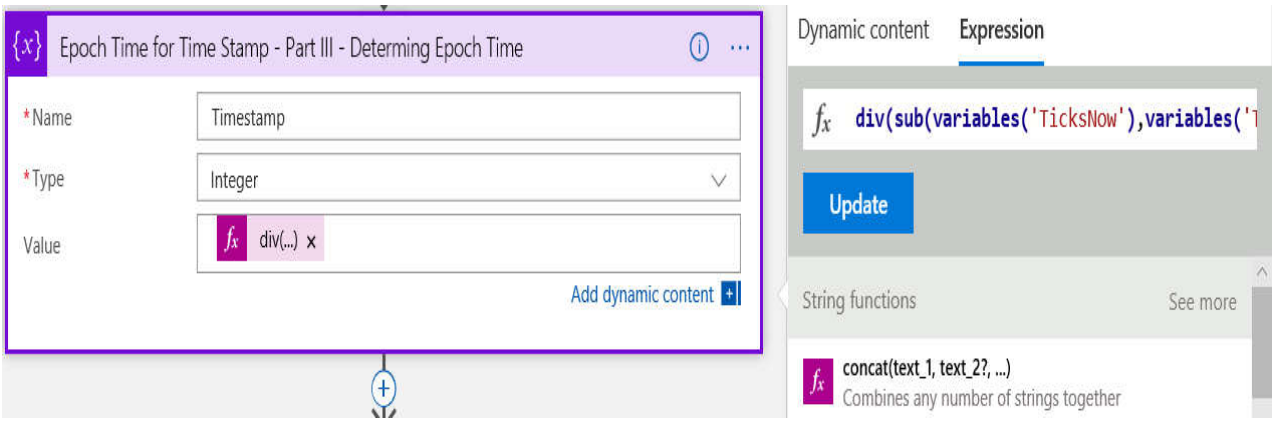


Рисунок 3.20 – Встановлення ('TicksTo1970')), 10000000)

Натисніть посилання "More" та виберіть " add a for each". Для вибору результату з попередніх властивостей кроків натисніть у полі та виберіть таблицю 1 зі списку динамічного вмісту.

У розділі "for each" натисніть посилання "Add an action" та виберіть "Service Bus – Send a message" зображено на рисунку 3.21.

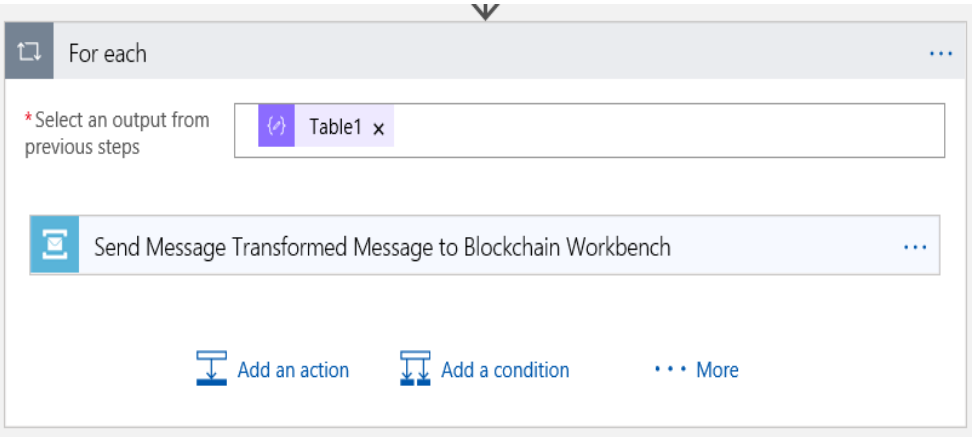


Рисунок 3.21 – Вікно додавання

Налаштуйте дію службового автобуса, щоб використовувати підключення, вказане на Сервісний транспорт, в розгортанні Workschar Blockchain зображено на рисунку 3.22. Встановіть властивість "Queue/Topic" для активності. Встановіть властивість SessionId на RequestId.


```
{
    "ContractActionId": null,
    "ConnectionId":,
    "UserChainIdentifier": "",
    "ContractCodeArtifactBlobStorageURL": "",
    "OperationName": "CreateContractAction",
    "ContractLedgerIdentifier": "",
    "WorkflowFunctionName": "IngestTelemetry",
    "WorkflowName": "RefrigeratedTransportation",
    "ContractActionParameters": [{
            "name": "humidity",
            "value": ""
        },{
            "name": "temperature",
            "value": ""
        },{
            "name": "timestamp",
            "value":
        }
    ],
    "RequestId": ""
}
```


Рисунок 3.22 – Параметри Services Bus

Нижче наведено шаблон повідомлення, яке потрібно додати до ресурсу Content зображено на рисунку 3.23.

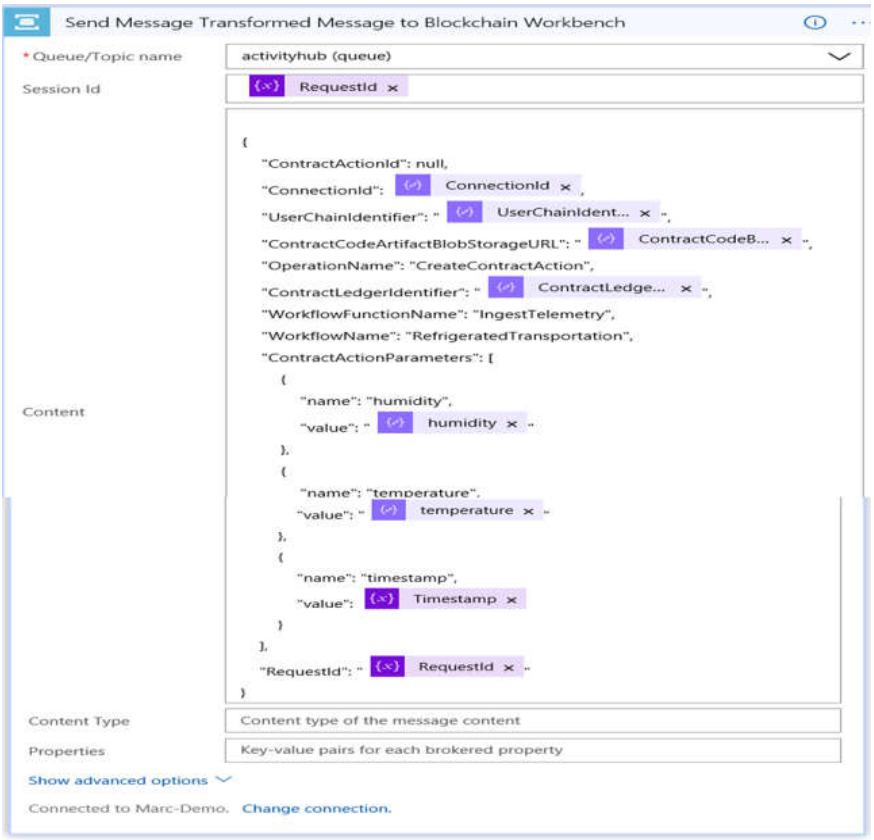


Рисунок 3.23 – Відправлення повідомлення

Розробляємо та налаштовує концентратор Інтернет речей, декілька Services Bus та додаток, яке передаватиме повідомлення від пристрою до розумного контракту в програмі Azure Blockchain.

3.3 Тестування алгоритму роботи блокчейн

Модульне тестування – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестованою. У процедурному програмуванні модулем вважають окрему функцію або процедуру.

В об'єктно-орієнтованому програмуванні – інтерфейс, клас. Модульні тести, або unit-тести, розробляються в процесі розробки програмістами та, іноді тестувальниками білої скриньки (white-box testers).

Зазвичай unit-тести застосовують для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку.

Розглянемо метод передавання повідомлення на сервер

```
[FunctionName("HttpTrigger")]
Public async static Task <IActionResult> RunAsync(
    [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)]
    HttpRequest req, TraceWriter log)
{
  log.Info("C# HTTP trigger function processed a request.");
  string name = req.Query["name"];
  string requestBody = new StreamReader(req.Body).ReadToEnd();
  dynamic data = JsonConvert.DeserializeObject(requestBody);
  name = name ?? data?.name;
  return name != null
    ? (ActionResult)new OkObjectResult($"Hello, {name}")
: new BadRequestObjectResult("Please pass a name on the query string or in the request body");}
```

Рисунок 3.24 – Функція HttpTrigger

Опис функції HttpTrigger який зображено на рисунку 3.24.

string name = req.Query[“name”]; – дістаємо параметр з полем name.

string requestBody = new StreamReader(req.Body).ReadToEnd(); – зчитуємо модель тілі запиту.

dynamic data = JsonConvert.DeserializeObject(requestBody); – десеріалізація тіла запиту.

return name != null ? (ActionResult) new OkObjectResult($”Hello, {name}”) : new BadRequestObjectResult(“Please pass a name on the query string or in the request body”); - робимо перевірку на існування значення в полі name. Якщо поле містить значення повертає його з статус кодом 200. Який свідчить про успішне виконання запиту.

В іншому випадку повертаємо статус код 400 з повідомлення про некоректно передане тіло запиту.

Тестуємо метод передавання повідомлення на сервер зображено на рисунку 3.25.

```
[TestClass]
public class HttpTriggerTest : FunctionTestHelper.FunctionTest
{
  [TestMethod]
  public async Task Request_With_Query()
  {
    var query = new Dictionary<String, StringValues>();
    query.TryAdd(“name”, “ushio”);
    var body = “”;
    var result = await HttpTrigger.RunAsync(req: HttpRequestSetup(query, body), log: log);
    var resultObject = (OkObjectResult)result;
    Assert.AreEqual(“Hello, ushio”, resultObject.Value);}
```

Рисунок 3.25 – Функція HttpTriggerTest

var query = new Dictionary<String, StringValues>(); – створюємо словник з ключе якого буде тип даних string а значенням тип даних StringValues.

query.TryAdd(“name”, “ushio”); – пробуємо додати значення у наш словник

var result = await HttpTrigger.RunAsync(req: HttpRequestSetup(query, body), log: log); – надсилаємо запит на сервер.

var resultObject = (OkObjectResult)result; – сереалізуємо відповідь сервера до OkObjectResult який свідчить про успішне завершення запиту.

За допомогою бібліотеки Assert перевіряємо на рівність надісланих даних на сервер і повернутих даних з сервера.

Метод помічника HttpRequestSetup створює макет об'єкта за допомогою Moq. Я написав допоміжний метод для полегшення тестування. Ви можете передати весь тест тут. Одиночний тест - це просто спотворення параметрів та встановлення значень. Тоді стверджуй висновок.

Azure Функції параметр спеціального класу. Іноді функції Azure використовують спеціальний клас, наприклад IAsyncCollector <T>.

```
[FunctionName(“Function1”)]
    public async static Task RunAsync(
        [QueueTrigger(“myqueue-items”, Connection = “connectionString”)]
        string myQueueItem, [Table(“MyTable”, Connection = “connectionString”)]
        IAsyncCollector<Message> messages, TraceWriter log) {
    log.Info($”C# Queue trigger function processed: {myQueueItem}”);
    var message = new Message();
    message.PartitionKey = “1”;
    message.RowKey = “2”;
    message.Text = myQueueItem;
    await messages.AddAsync(message); }
```

Рисунок 3.26 – Функція Function1

У даній функції методу створюємо об’єкт класу message. Додаємо його в нашу колекцію

Також створюємо простий об'єкт Mock на FunctionTest, який є просто помічником тестування одиниці зображено на рисунку 3.26.

```
[TestClass]
    public class QueueTriggerTest : FunctionTestHelper.FunctionTest
{
  [TestMethod]
  public async Task Recieve_Queue_And_Emit_To_Table()  {
    var col = new AsyncCollector<Message>();
    var json = “{\”name\”: \”ushio\”}”;
    await DotNet.QueueTrigger.RunAsync(json, col, log);
    Assert.AreEqual(json, col.Items[0].Text); }
}
```

Рисунок 3.27 – Функцію QueusTriggerTest

Створюємо колекцію (зображено на рисунку 3.27) яку передаємо у метод RunAsync з повідомленням у форматі JSON .

Для тестування довговічних функцій ви можете зіткнутися з проблемою знущання. Класи параметрів - це герметичний клас. Це означає, що ви не можете знущатися з неї. Тоді, як насміхатися з цими?

```
[FunctionName(“DurableFunctions”)]
public static async Task<List<string>> RunOrchestrator([OrchestrationTrigger]
  DurableOrchestrationContextBase context) {
  var outputs = new List<string>();
  outputs.Add(await context.CallActivityAsync<string>(“DurableFunctions_Hello”, “Tokyo”));
  outputs.Add(await context.CallActivityAsync<string>(“DurableFunctions_Hello”, “Seattle”));
  outputs.Add(await context.CallActivityAsync<string>(“DurableFunctions_Hello”, “London”));
  return outputs;}
```

Рисунок 3.28 – Функця DurableFunctions

Створюємо функцію DurableFunctions (зображено на рисунку 3.28) у якій створюємо список наших вихідних значень. Поступово добавляємо в цей список значень та повертаємо його.

Тестуємо метод Run_Orchestrator який зображено на рисунку 3.29.

```
[TestClass]
    public class DurableFunctionTest : FunctionTestHelper.FunctionTest {
[TestMethod]
    public async Task Run_Orchectrator(){
  var contextMock = new Mock<DurableOrchestrationContextBase>();
  contextMock.Setup(context=>context.CallActivityAsync<string>(“DurableFunctions_Hello”, “Tokyo”)).
  Returns(Task.FromResult<string>(“Hello Tokyo!”));
  contextMock.Setup(context=>context.CallActivityAsync<string>(“DurableFunctions_Hello”,
  “Seattle”)).Returns(Task.FromResult<string>(“Hello Seattle!”));
  contextMock.Setup(context=>context.CallActivityAsync<string>(“DurableFunctions_Hello”, “London”)).
  Returns(Task.FromResult<string>(“Hello London!”));
  var result = await DotNet.DurableFunctions.RunOrchestrator(contextMock.Object);
  Assert.AreEqual(“Hello Tokyo!”, result[0]);
  Assert.AreEqual(“Hello Seattle!”, result[1]);
  Assert.AreEqual(“Hello London!”, result[2]);}
```

Рисунок 3.28 – Функція Run_Orchestrator

Створюємо об'єкт Mock в який встановлюємо тестові дані. Викликаємо функцію та перевіряємо значення яке повертаємо з параметрами які передавали.

Розроблено програмну реалізацію алгоритму ідентифікації об’єктів Інтернет речей на основі технології блокчейн. Реалізовано публічний блокечейн на Microsoft Azure. Протестовано алгоритм роботи блокчейн.

ВИСНОВКИ

В роботі розв’язано актуальну задачу розробки алгоритмів ідентифікації об’єктів Інтернет речей на основі технології блокчейн. При цьому отримано наступні результати:

1. Розкрито теоретичні основи Інтернет речей та технології блокчейн, зокрема алгоритми взаємодії пристроїв Інтернет речей. Показано переваги та недоліки технології блокчейн.

2. Проаналізовано протоколи з’єднання MQTT та CoAP та область застосування.

3. Розглянуто платформу Microsoft Azure та проведено порівняння і тестування з іншими платформам.

4. Проведено порівняння алгоритмів консенсусу Інтернет речей IOTA, IOTW та технології блокчейн Proof of Work, Poof of State, Proof of Assignment.

5. Розроблено структурну схему ідентифікації об'єктів на основі технології блокчейн.

6. Розроблено структуру та реалізовано на мові C# алгоритм ідентифікації на основі технології блокчейн.

7. Удосконалено алгоритми ідентифікації об’єктів Інтернет речей за рахунок використання технології блокчейн, що дозволило підвищити надійність ідентифікації.

8. Реалізовано публічний блокчейн на основі платформи Microsoft Azure.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Andreas M. Antonopoulos. Mastering Bitcoin: unlocking digital cryptocurrencies. "O'Reilly Media, Inc.", 2014, 298 p.

2. Appii проведення перевірки підтвердження CV технологією блокчейн [Електронний ресурс]. – Режим доступу: https://appii.io/.

3. Ascribe підтвердження авторського права митцям [Електронний ресурс]. – Режим доступу: https://www.ascribe.io/.

4. Borderless управління розумними контрактами [Електронний ресурс]. – Режим доступу: http://www.borderless.tech/.

5. Ethereum платформа для створення децентралізованих онлайн-сервісів на базі блокчейн [Електронний ресурс]. – Режим доступу: https://www.ethereum.org/.

6. Everledger забезпечує реєстр для алмазів [Електронний ресурс]. – Режим доступу: https://www.everledger.io/.

7. Factom розподілення реєстру на фінансовому ринку [Електронний ресурс]. – Режим доступу: https://www.factom.com/.

8. Follow My Vote онлайн голосування [Електронний ресурс]. – Режим доступу: https://followmyvote.com/.

9. Grid Singularity децентралізована платформа для обміну енергією [Електронний ресурс]. – Режим доступу: https://gridsingularity.com/.

10. IEEE Standard Verilog Hardware Description Language, IEEE Computer Society, IEEE, New York, NY, 1364 – 2001 с.

11. IOTA [Електронний ресурс]. – Режим доступу https://www.iota.org/.

12. IOTW [Електронний ресурс]. – Режим доступу https://www.iotw.org/

13. Klint Finley, The Wired guide to the blockchain [Electronic resource]. - Access mode: https://www.wired.com/story/guide-blockchain.

14. Liao Junsong. FPGA based wireless sensor node with customizable event-driven architecture. / Liao Junsong // EURASIP Journal on Embedded Systems 2013.1, 2013. – 1-11 с.

15. Microsoft Azure для образования [Електронний ресурс] // Microsoft. – 2014. – Режим доступа : https://msdn.microsoft.com/ukua/dn133768.aspx.

16. Nolan Bauerle, What is Blockchain Technology [Electronic resource]. - Access mode: https://www.coindesk.com/information/what-is-blockchaintechnology.

17. Novo, O. Blockchain meets IoT: An architecture for scalable access management in IoT. IEEE Int. Things J. 2018, 5, pp.1184– 1195.

1.8 Samaniego, M.,Deters, R. Blockchain as a Service for IoT. In Proceedings of the 9th IEEE International Conference on Internet of Things, Chengdu, China, 15–18 December 2016; pp. 433–436.

19. Ubitquity Enterprise-Ready Blockchain-Secured платформа для ведення обліку нерухомості [Електронний ресурс]. – Режим доступу: https://www.ubitquity.io.

20. Verbatim компанія виробник носіїв для зберігання інформації [Електронний ресурс]. – Режим доступу: https://www.verbatim-europe.nl/nl/security/.

21. Акушский И.Я. Проблемы безопасности интернет вещей / И.Я.Акушский, И.Т.Пак // Вопросы кибернетики. – 1977, Т.28. – С.36-56.

22. Барсов В.И. Методология параллельной обработки информации в модулярной системе счисления. / В.И.Барсов, Л.С.Сорока, В.А.Краснобаев: 23. Монография.- Х.: МОН, УИПА, 2009. – 268 с.

23. Варгаузин В. Помехоустойчивое кодирование в пакетных сетях. / В.Варгаузин // ТелеМультиМедиа. – 2005. – №9. – С.10-16.

Віллам. Я., Internet of Things: The Simple Guide.– Тернопіль «Книгарня Є», 2016. – 69с.

24. Ворнер. Д., Архітектура Інтернет речей.– «ДМК Пресс», 2012. – 656с.

25. Грінгард. С., Інтернет речі – «Клуб Сімейного Дозвілля», 2018. – 176с.

26. Девід. Р., Дивовижні технології. Дизайн та інтернет речей – «Клуб Сімейного Дозвілля», 2018. – 246с.

27. Дунець Р.Б. Мережевий протокол MQTT . / Р.Б.Дунець, Д.Я. Тиранський // Радіоелектронні і комп’ютерні системи. – 2010. – №7(48). – С.200 – 204.

28.Казимир В. В. Проектування комп'ютерних систем на основі мікросхем програмованої логіки: монографія / С. А. Іванець, Ю. О. Зубань, В. В. Казимир, В. В. Литвинов. – Суми : Сумський державний університет, 2013. – 313 с.

29. Колліер. М., Основи архітектури Інтерне речей.– Тернопіль «Книгарня Є», 2016. – 238с.

30. Конфіденційність-обізнаність на основі Blockchain [Електронний ресурс] / Режим доступу до ресурсу http://goo.gl/3Nv2oK.

31. Краснобаев В. А. Метод обработки данных / В. А.Краснобаев, А. С.Янко, С. А Кошман., С. А.Сомов, Ю. П.Бендес // Збірник наукових праць Харківського університету Повітряних сил. – 2014. –№2 – С.121-126.

32. Краснобаєв В.А. Метод підвищення достовірності контролю даних / В.А.Краснобаєв, С.О.Кошман, М.О.Мавріна // Кибернетика и системний аналіз. – 2014. – Том 50, №6 .– С.167 –175.

33. Мережевий протокол CoAP [Електронний ресурс] / Режим доступу до ресурсу http://lib.tssonline.ru/articles2/internet-of-things/protokol-interneta-veschey-coap.

34. Мережевий протокол MQTT [Електронний ресурс] / Режим доступу до ресурсу https://1sheeld.com/mqtt-protocol/.

35. Мережевий протокол STOMP [Електронний ресурс] / Режим доступу до ресурсу https://stomp.github.io/.

36. Методичні рекомендації до виконання магістерської роботи з освітнього ступеня “Магістр”. Спеціальність: 123 - Комп’ютерна інженерія. Магістерська програма - Комп’ютерна інженерія" / О.М. Березький, Л.О. Дубчак, Г.М. Мельник / Під ред. О.М. Березького – Тернопіль: ТНЕУ, 2018.– 41 с.

37. Мурин М. В. Порівняння алгоритмів консенсусу для Інтернет речей / Мар’ян Васильович Мурин // Матеріали семінару комп’ютерні науки та інформаційні технології, CSIT’2018 / Тернопіль, 2018. – С. 49–50.

38. Нечаев В. И. Элементы криптографии. Основы теории защиты информации. / В. И. Нечаев – М.: Высшая школа, 1999. – 278 с.

39. Основні види тестування програмного забезпечення [Електронний ресурс]. – Режим доступу: http://pohnews.org/5293-osnovni-vidi-testuvannyaprogramnogo-zabezpechennya.

40. Палагин А.В. Проектирование реконфигурируемых цифровых систем: монография / А.В. Палагин, А.А. Баркалов, В.Н. Опанасенко, Л.А. Титаренко. – Луганск: изд-во ВНУ им. В. Даля, 2011. – 432 с.

41. Петров А. А. Компьютерная безопасность. Криптографические методы защиты. / А. А. Петров – М.: ДМК, 2000. – 448 с.

42. Протоколи з'єднання з IoT [Електронний ресурс] / Режим доступу до ресурсу http://www.rtsoft.ru/press/articles/detail.php?ID=2718.

43. Розробка додатків на платформі Windows Azure [Електронний ресурс] / Режим доступу до ресурсу https://azure.microsoft.com/ru-ru/campaigns/cloud-application-architecture-guide/.

44. Стейнер. К., Тотальна автоматизація. Як комп'ютерні алгоритми змінюють світ – Тернопіль «Книгарня Є», 2018. – 343с.

45. Суханов Є., Алгоритми Інтернет речей – Тернопіль «Книгарня Є», 2017. – 546с.

46. Тепскоттом. А., Internet of Things Revolution.– Тернопіль «Книгарня Є», 2016. – 324с.

47. Томашевський В.М. Моделювання систем / В.М. Томашевський. – К.: Видавнича група BHV, 2005. – 352с.

48. Что такое Интернет вещи? Расскажем простыми словами [Електронний ресурс]. – Режим доступу: https://coinspot.io/beginners/ chto-takoe-blokchejn.

49. Яцків Н.Г., Яцків С.В. Перспективи використання технології блокчейн в мережі Інтернет речей. Науковий вісник НЛТУ України. - 2016. - Вип. 26.8. – С. 381-387.

50. Яцків Н.Г., Яцків С.В. Типи атак на Інтернет речей. Захист інформації і безпека інформаційних систем: матеріали VI Міжнародної наукової технічної конференції. – Львів: Видавництво Львівської політехніки, 2017 р. – С.162-163.