КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

До захисту допущено

завідувач кафедри ІСТ

Олександр КУЧАНСЬКИЙ

(підпис) (ім'я, ПРІЗВИЩЕ).

« » червня 2022р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

спеціальності 126 «Інформаційні системи та технології»
освітньої програми «Програмні технології інтернет речей»

на тему: «Розробка системи аналізу даних датчиків IoT комерційної будівлі та інтеграція у Blockchain технологію»

Виконав студент 4 курсу, групи ІР-41

Костянтин БУТЕНКО _____

(ім'я, ПРІЗВИЩЕ) (підпис)

Керівник к.т.н., доцент Ростислав ЛІСНЕВСЬКИЙ _____

(посада, науковий ступінь, вчене звання, ім'я, ПРІЗВИЩЕ) (підпис)

Консультант нормо контроль к.т.н., доцент Ростислав ЛІСНЕВСЬКИЙ _____

(назва розділу) (посада, науковий ступінь, вчене звання, ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент директор ТОВ «МІОРА» Юрий БОДЕЛАН _____

(посада, науковий ступінь, вчене звання, ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що кваліфікаційна робота не має запозичень з праць інших авторів без відповідних посилань.

Здобувач освіти _____

(підпис)

Київ – 2022 року

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

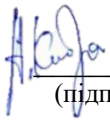
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційні системи та технології
Освітній рівень Бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітня програма Програмні технології інтернет речей

ЗАТВЕРДЖУЮ

завідувач кафедри ІСТ



Олександр КУЧАНСЬКИЙ
(підпис) (ім'я, ПРІЗВИЩЕ).

« » червня 2022р.

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Здобувач освіти: Костянтин БУТЕНКО

Група: ІР-41

1. **Тема кваліфікаційної роботи бакалавра:** «Розробка системи аналізу даних датчиків IoT комерційної будівлі та інтеграція у Blockchain технологію».

Затверджена протоколом засідання кафедри ІСТ №05/21_22 від 03.12.2021 року.

2. **Строк подання студентом готової роботи** − «22» червня 2022 р.

3. **Вихідні дані до роботи:** дослідження можливості використання технології Blockchain для зберігання даних з датчиків. Проектування системи збереження та аналізу даних з датчиків IoT. Нормативні документи та вимоги до побудови та експлуатації комерційних приміщень. Системи та технології передачі даних IoT. Аналіз датчиків, що використовуються у IoT. Програмна реалізація системи для збереження та аналізу даних з датчиків IoT за допомогою технології Blockchain.

4. **Зміст роботи:** РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ТА АНАЛІЗ РІШЕННЯ (постановка задачі, аналіз існуючих у світі систем аналізу датчиків IoT, розробки Blockchain в IoT, аналіз існуючих датчиків IoT, вхідні дані датчиків,

вимоги нормативних документів для комерційних будівель); РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ ТА МЕТОДІВ ОБРОБКИ ДАНИХ ІОТ РІШЕННЯ (аналіз методів обробки даних з IoT датчиків, огляд протоколів IoT, вибір технологій та протоколів, архітектура (модель) аналізу даних комерційних будівель); РОЗДІЛ 3. РЕАЛІЗАЦІЯ АРХІТЕКТУРИ ІОТ РІШЕННЯ (реалізація датчика, реалізація хабу, реалізація обкладинки для блокчейну, реалізація блокчейну, реалізація адміністративної панелі з візуалізацією даних).

5. Перелік графічного матеріалу: зображення веб-інтерфейсу програм проаналізованих аналогів; схема алгоритму роботи системи аналізу та збору даних; зображення моделі з'єднання та використання допоміжних сервісів; архітектура системи збору та обробки інформації з датчиків; зображення датчиків різних типів; знімки екрану з веб-інтерфейсами допоміжних сервісів, що було використано; знімки екрану командного рядка з результатами виконання команд; знімки екрану з середовищем розробки під час створення програми; графічний інтерфейс веб застосунку.

6. Календарний план виконання роботи:

Етапи виконання кваліфікаційної роботи бакалавра	**Термін виконання**	**Результат виконання**
1. Вибір тематики кваліфікаційної роботи бакалавра	01.09.2021-01.10.2021	виконано
2. Наказ про затвердження тем кваліфікаційної роботи бакалавра та призначення керівників	03.12.2021	виконано
3. Розробка плану кваліфікаційної роботи бакалавра і його погодження з керівником	25.12.2021	виконано
4. Написання І розділу кваліфікаційної роботи	19.03.2022	виконано
5. Написання ІІ розділу кваліфікаційної роботи	25.04.2022	виконано
6. Написання ІІІ розділу кваліфікаційної роботи	29.04.2022	виконано
8. Підготовка висновків і пропозицій	30.04.2022	виконано
9. Попередній захист кваліфікаційної роботи	12.05.2022	виконано
10. Перевірка на плагіат	13.05.2022-15.06.2022	виконано
11. Нормоконтроль	02.06.2022-	виконано

	06.06.2022	
12. Рецензування кваліфікаційної роботи бакалавра і представлення роботи на кафедру в друкованому вигляді	15.06.2022	виконано
13. Захист кваліфікаційної роботи бакалавра	23.06.2021-24.06.2021	

Дата видачі завдання «__» _____ 2022 р.

Керівник роботи: к. т. н., доцент Ростислав ЛІСНЕВСЬКИЙ_____ (підпис)

Завдання прийняв до виконання:

Здобувач освіти на освітньому рівні «бакалавр» 4-го курсу групи ІР-41

Костянтин БУТЕНКО _____

(Власне Ім’я, ПРІЗВИЩЕ) (підпис)

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційних систем та технологій

Освітня програма «Програмні технології інтернет речей»

Кваліфікаційна робота бакалавра Костянтина БУТЕНКО

Тема роботи: «Розробка системи аналізу даних датчиків IoT комерційної будівлі та інтеграція у Blockchain технологію»

Мета кваліфікаційної роботи бакалавра – аналіз систем збереження та обробки даних IoT, проектування та створення IoT системи збереження даних, що генерують датчики IoT, за допомогою технології Blockchain з ціллю їх наступного аналізу за допомогою візуалізації результатів на основі проаналізованих даних.

Об'єкт дослідження – технологія Blockchain, яка використовується для збереження даних з можливістю їх наступного аналізу.

Предмет дослідження – система збереження та аналізу інформації, отриманої від датчиків IoT за допомогою технології Blockchain.

Апробація результатів. Було подано тези доповідей на Міжнародну науково-практичну конференцію «Прикладні системи та технології в інформаційному суспільстві», що проходила 30 вересня 2021 року та на VIII Міжнародну науково-технічну Internet-конференцію «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами» (НУХТ), що проходила 25–26 листопада 2021 р.

Кваліфікаційна робота бакалавра складається зі змісту, вступу, основної частини, яка включає три розділи, висновків та списку використаних джерел. Всього 86 сторінок.

КЛЮЧОВІ СЛОВА: Internet of Things, IoT, PHP, Blockchain, MQTT, MySQL, OOP, RabbitMQ, Symfony, Web-application, JavaScript, CMS, Visualization.

ABSTRACT

TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Information Technologies

Department of Information Systems and Technologies

Educational Program "Software Technologies of the Internet of Things"

Qualification work of bachelor Konstantin BUTENKO

Work topic: "Development of a data analysis system for IoT sensors in a commercial building and integration into Blockchain technology"

The purpose of the bachelor's qualification work is to analyze existing IoT data storage and processing systems, and design and create IoT data storage systems generated by IoT sensors using Blockchain technology for further analysis by visualizing the results based on the analyzed data.

The object of research is to save data with the help of Blockchain with the possibility of their analysis.

The subject of research is the system of storage and analysis of information received from IoT sensors using Blockchain technology.

Approbation of results.

Abstracts were submitted to the International Scientific and Practical Conference "Applied Systems and Technologies in the Information Society" held on September 30, 2021, and the VIII International Scientific and Technical Internet Conference "Modern Methods, Information, Software and Technical Support of Organizational Management Systems". technical and technological complexes "(NUHT), which took place on November 25-26, 2021.

The bachelor's thesis consists of the content, introduction, and main part, which includes three sections, conclusions, and a list of sources used. Total 86 pages.

KEYWORDS: Internet of Things, IoT, PHP, Blockchain, MQTT, MySQL, OOP, RabbitMQ, Symfony, Web-application, JavaScript, CMS, Visualization.

ЗМІСТ

ВСТУП

Актуальність теми. Нещодавнє поширення пристроїв і датчиків Інтернету речей (IoT) створило потребу в нових способах зберігання та аналізу отриманих даних. Згідно деяких досліджень у 2025 пристрої IoT зможуть згенерувати приблизно 73 Зетабайт даних (1,099,511,627,776 Гігабайт)[52]. Це дуже велика кількість даних які потрібно десь зберігати та для яких існує ще одна проблема, а саме проблема із їх безпечним зберіганням та аналізом. Дані часто є конфіденційними та особистими, і якщо вони потрапляють у чужі руки, вони можуть бути використані в підступних цілях.

Одним з перспективних рішень є використання технології Blockchain, яка може забезпечити безпечний і децентралізований спосіб зберігання та обробки даних.

Blockchain – це розподілена база даних, що складається з ланцюжка блоків, кожен з яких містить запис даних. Технологію Blockchain можна використовувати для створення безпечної, децентралізованої та захищеної від несанкціонованого доступу системи для зберігання та обробки даних.

Технологія Blockchain пропонує ряд переваг, які роблять її добре придатною для використання з пристроями та датчиками IoT. По-перше, Blockchain є безпечним і стійким, що важливо для забезпечення конфіденційності та безпеки даних IoT. По-друге, Blockchain децентралізований, а це означає, що дані можуть зберігатися та оброблятися в розподіленій мережі комп’ютерів. Це може покращити масштабованість та надійність обробки даних IoT.

Хоча технологія Blockchain пропонує багато потенційних переваг для Інтернету речей, існують також деякі проблеми, які необхідно вирішити. По-перше, Blockchain є відносно новою технологією, і вона все ще розвивається. Це означає, що є ще деякі технічні проблеми, які потребують вирішення. По-друге, Blockchain вимагає певної обчислювальної потужності для роботи, що може бути проблемою для пристроїв IoT з обмеженими ресурсами.

Незважаючи на труднощі, вже є кілька перспективних варіантів використання Blockchain в Інтернеті речей. Одним із прикладів є смарт-контракти, які можна використовувати для автоматичного виконання транзакцій на основі даних із

датчиків Інтернету речей. Інший приклад – у сфері обміну даними, де Blockchain можна використовувати для створення безпечного та децентралізованого способу обміну даними пристроїв IoT.

Технологія Blockchain може змінити спосіб зберігання та обробки даних із пристроїв і датчиків Інтернету речей. Однак є деякі проблеми, які потребують вирішення, наприклад, масштабованість Blockchain та обчислювальна потужність, необхідна для його запуску. Тим не менш, вже є кілька перспективних варіантів використання Blockchain в Інтернеті речей, і ця технологія, ймовірно, продовжить розвиватися та отримувати все більш широке поширення в майбутньому.

Отож, виходячи з описаної актуальності проблеми та взявши до уваги усю зазначену інформацію було сформульовано ***мету роботи***: аналіз існуючих систем збереження та обробки даних IoT, проектування та створення IoT системи збереження та аналізу даних, що генерують датчики IoT, за допомогою технології Blockchain з ціллю їх наступного аналізу за допомогою візуалізації результатів на основі проаналізованих даних.

Предметом дослідження є система збереження та аналізу інформації, отриманої від датчиків IoT за допомогою технології Blockchain.

Об'єктом дослідження є технологія Blockchain, яка використовується для збереження даних з можливістю їх наступного аналізу.

Методами дослідження є аналіз та вивчення інформації про аналогічні системи збереження чи/та обробки інформації з IoT. Вивчення документації існуючих технологічних рішень для обробки та передачі інформації. Вивчення нормативних документів. Проектування архітектури системи та пошук моделей аналогічних систем.

Для виконання мети дослідження необхідно:

- Спроектувати та реалізувати систему збереження та аналізу інформації за допомогою технології Blockchain.
- Спроектувати та реалізувати модель передачі даних між усіма компонентами системи.
- Реалізувати адміністративну панель з можливістю реєструвати датчики та візуалізувати інформацію отриману від них.

РОЗДІЛ 1. ПОСТАНОВКА ЗАДАЧІ ТА АНАЛІЗ РІШЕННЯ

1.1. Постановка задачі

Зберігання та аналіз інформації з датчиків Інтернету речей є критично важливим завданням для будь-якої організації, яка використовує на дані з цих пристроїв. При розробці такої системи слід враховувати багато факторів, включаючи тип даних, які збираються, частоту збору даних, необхідну ємність зберігання та потужність обробки, необхідну для аналізу даних.

Розробка систем зберігання та аналізу інформації від датчиків IoT є складним завданням, яке вимагає використання різноманітних технологій. Однією з найбільш перспективних технологій для цієї мети є Blockchain.

Використання Blockchain при розробці систем для зберігання та аналізу інформації від датчиків IoT має ряд переваг. По-перше, це гарантує, що дані захищені та їх неможливо підробити. По-друге, це забезпечує прозорість і простежуваність даних. І по-третє, це дозволяє розробляти децентралізовані програми, які можуть самостійно працювати в мережі.

Використання технології Blockchain в Інтернеті речей є природним, оскільки технологія розподіленої мережі може забезпечити безпечний, захищений від несанкціонованого доступу спосіб зберігання даних із датчиків Інтернету речей.

Щоб розробити успішну систему IoT на основі блокчейну, необхідно вирішити кілька проблем, зокрема масштабованість, сумісність та керування даними. Однак потенційні переваги використання блокчейну в Інтернеті речей є значними, і ця технологія вже пілотується рядом компаній і організацій.

Визначено наступні цілі роботи:

- Аналіз можливості застосування технології Blockchain для збереження даних згенерованих IoT.
- Проектування системи збереження та аналізу інформації за допомогою технології Blockchain.
- Розробка системи Blockchain, яка може зберігати та оброблювати вхідну інформацію з датчиків.

- Спроектувати та реалізувати модель передачі даних між усіма компонентами системи.
- Реалізувати адміністративну панель з можливістю реєструвати датчики та візуалізувати інформацію отриману від них.

1.2. Аналіз існуючих у світі систем аналізу датчиків IoT.

Аналітика даних IoT – це процес збору, аналізу та отримання інформації з даних, створених підключеними пристроями та датчиками. Розуміючи закономірності та взаємозв'язки в цих даних, підприємства можуть оптимізувати свою діяльність, покращити взаємодію з клієнтами та створити нові потоки доходу.

Аналітику даних IoT можна використовувати для різних цілей, зокрема:

- Моніторинг та керування пристроями та парками.
- Виявлення та діагностика проблем.
- Підвищення якості продукції.
- Створення нового досвіду клієнтів.
- Розробка нових бізнес-моделей.

Щоб бути ефективним, аналіз даних IoT повинен мати можливість обробляти великі обсяги даних з різних джерел. Він також повинен мати можливість ідентифікувати та відстежувати зв'язки між точками даних.

Платформи для аналізу даних IoT зазвичай пропонують різноманітні функції, зокрема:

- Збір і зберігання даних.
- Обробка та аналітика даних.
- Візуалізація та звітність.
- Машинне навчання та штучний інтелект[1].

Великі дані та Інтернет речей можна використовувати разом для покращення аналітики. Перший приклад – використання датчиків для відстеження продуктивності обладнання у виробничому процесі. Відстежуючи дані, зібрані датчиками, компанія може виявити проблеми з обладнанням і внести зміни для покращення продуктивності. Другий приклад – використання даних, зібраних метеостанціями, для підвищення точності прогнозів погоди. Третій приклад –

використання даних, зібраних інтелектуальними лічильниками, для підвищення ефективності використання енергії в будинках і підприємствах. Четвертий приклад – використання даних, зібраних пристроями GPS, для підвищення точності інформації про транспортний потік. П’ятий приклад – використання даних, зібраних камерами відеоспостереження, для підвищення точності відеоспостереження[2].

Google Nest – це система домашньої автоматизації, яка дозволяє користувачам керувати різними підключеними пристроями у своєму домі за допомогою однієї програми. Система включає концентратор, який підключається до домашнього маршрутизатора користувача та керує різними пристроями, а також мобільний додаток, який дозволяє користувачам керувати системою зі свого смартфона або планшета. Систему можна використовувати для керування такими пристроями, як термостати, освітлення, камери безпеки та дверні замки. Додаток також дозволяє користувачам встановлювати правила та розклади для своїх пристроїв, а також отримувати сповіщення, коли відбуваються певні події. Система Nest сумісна з різними пристроями, включаючи термостати Nest, камери Nest, дверні замки Nest та димові сигналізатори Nest.[3]

AWS IoT Analytics – це хмарна аналітична служба, яка дозволяє легко запускати й реалізовувати аналітику на величезних обсягах даних IoT. Служба дає змогу збирати, обробляти, зберігати й аналізувати дані Інтернету речей у масштабі. Сервіс надає аналітичну систему на основі SQL, яку можна використовувати для виконання випадкових запитів та візуалізації своїх даних. Сервіс пропонує масштабовану безсерверну архітектуру, яка полегшує запуск аналітичних конвеєрів і керування ними[4].

Philips Hue – це лінійка світлодіодних ламп, що змінюють колір, та супутніх аксесуарів, створених Philips. Система Hue включає міст, який з’єднується з маршрутизатором Wi-Fi вашого будинку і може керувати до 50 лампочками. Міст необхідний для використання системи Hue і є єдиним способом підключення світильників до Інтернету.

Програма Philips Hue – це офіційна програма від Philips, яка дозволяє керувати світлом Hue. Додаток дозволяє робити все, що може зробити міст Hue, включаючи зміну кольорів світла, створення розкладів та встановлення будильників. Додаток

також дозволяє керувати іншими продуктами Philips Hue, такими як світлові панелі Living Colours і нічник Bloom.

Додаток має три основні розділи: панель приладів, перегляд кімнат і вікно освітлення. Панель інструментів – це головний екран програми, який надає швидкий доступ до функцій, які найчастіше використовуються. Подання «Кімнати» дозволяє керувати всіма освітленнями в певній кімнаті, а режим «Світло» дає змогу керувати кожним світлом окремо[5].

Belkin WeMo – це програма для домашньої автоматизації, яка дозволяє користувачам керувати домашньою електронікою зі свого смартфона або планшета. Додаток дозволяє користувачам створювати правила для автоматизації свого будинку, наприклад, вимикати світло, коли вони виходять з дому, або вмикати кавоварку, коли вони прокидаються. WeMo також дозволяє користувачам дистанційно керувати своєю домашньою електронікою з будь-якої точки світу[6].

Samsung SmartThings – це мобільна програма, яка дозволяє користувачам керувати, контролювати й автоматизувати свій будинок за допомогою таких пристроїв, як освітлення, дверні замки, термостати тощо. Додаток також дозволяє користувачам створювати правила та процедури для автоматизації свого будинку, наприклад, вимикати світло, коли всі виходять з дому, або встановлювати термостат на певну температуру, коли хтось лягає спати. Крім того, додаток надає користувачам інформаційну панель, на якій вони можуть переглядати стан всіх своїх підключених пристроїв і переглядати історію всіх подій, що відбулися в їхньому домі[7].

IFTTT – це безкоштовний веб-сервіс, який дозволяє користувачам створювати ланцюжки простих умовних операторів, які називаються «рецептами», які запускаються на основі змін в інших веб-сервісах, таких як Gmail, Facebook, Instagram та Pinterest. IFTTT – це абревіатура від «Якщо це, то це».

Рецепт складається з двох частин: тригера (If This) і дії (Then That). Коли відбувається подія тригера, IFTTT запускає дію. Наприклад, рецепт може надсилати повідомлення електронною поштою, коли користувач публікує фотографію в Instagram.

Рецептами IFTTT можна поділитися з іншими користувачами[8].

Stringify – це програма IoT, яка дозволяє користувачам підключатися та керувати своїми улюбленими пристроями та послугами з підтримкою Інтернету. За допомогою Stringify користувачі можуть створювати правила або «Потоки», які автоматично запускатимуть певні дії на основі конкретних умов. Наприклад, користувач може створити сценарій, який вмикає світло у вітальні, коли вони приходять додому, або надсилає текстове повідомлення, якщо вхідні двері залишаються відкритими.

Stringify не обмежується лише домом, його можна використовувати в офісі або навіть на ходу. Наприклад, користувач може створити рецепт, який автоматично перевіряє їхній календар і оновлює список справ, коли вони приходять на роботу, або який вимикає світло та встановлює повідомлення про вихід, коли вони йдуть у відпустку. Stringify також сумісний з широким спектром пристроїв і служб, включаючи Nest, Philips Hue, Amazon Echo, Fitbit, IFTTT та багато інших[9].

Stringify було закрито у червні 2019 року.

Wink – це платформа для розумного дому, яка дозволяє користувачам керувати домашнім середовищем за допомогою мобільного додатка. Додаток надає користувачам можливість налаштовувати свої пристрої та керувати ними, створювати засоби автоматизації та підпрограми, а також контролювати споживання енергії. Wink також пропонує відкритий API, який дозволяє стороннім розробникам створювати нові інтеграції та програми для платформи.

Додаток Wink IoT призначений для роботи з різними пристроями в домі, включаючи освітлення, вимикачі, термостати, дверні замки, двері гаража, камери безпеки тощо. Користувачі можуть керувати цими пристроями окремо або групувати їх у кімнати чи сцени. Автоматику можна створювати для вмикання/вимкнення груп пристроїв на основі певних умов, таких як час доби або місце розташування. Рутини дозволяють користувачам виконувати кілька дій одним дотиком. Наприклад, користувач може створити розпорядок дня, який вимикає все світло і замикає двері, коли вони йдуть на роботу вранці.

Додаток Wink також надає користувачам уявлення про їхнє споживання енергії. Користувачі можуть бачити, скільки енергії споживає кожен пристрій, і відстежувати їх загальне споживання з часом. Ця інформація може допомогти

користувачам заощадити гроші, виявляючи пристрої, які споживають більше енергії, ніж необхідно[10].

Logitech Harmony – це потужна та універсальна програма, яка дозволяє керувати своєю домашньою розважальною системою зі свого мобільного пристрою. За допомогою Harmony ви можете легко налаштувати домашній кінотеатр, стереосистему та ігрові системи та керувати ними, а також керувати ними за допомогою голосу, дистанційного керування або мобільного додатка. Harmony також забезпечує безперебійну інтеграцію з пристроями розумного дому, такими як Nest і Philips Hue, тож ви можете керувати всім своїм домом за допомогою однієї простої програми.

Основна функція Harmony – надати універсальний пульт дистанційного керування для всіх ваших домашніх розважальних пристроїв. Він робить це за допомогою інфрачервоної технології для надсилання команд на ваші пристрої. Harmony може керувати практично будь-яким пристроєм, який використовує ІЧ-пульт, включаючи телевізори, програвачі Blu-ray, ігрові консолі та звукові панелі. На додаток до інфрачервоного керування, Harmony також підтримує Bluetooth для підключення до таких пристроїв, як навушники та колонки.

Harmony також має вбудовану базу даних із понад 225 000 пристроїв від понад 6000 брендів, тому цілком ймовірно, що вона буде сумісна з усім, що у вас є. по телефону, щоб допомогти вам правильно налаштувати все[11].

Insteon – це компанія з домашньої автоматизації, яка виробляє обладнання та програмне забезпечення для ринку домашньої автоматизації. Лінійка продуктів Insteon включає в себе вимикачі світла, розетки, термостати, датчики та інші пристрої, якими можна керувати через мобільний додаток або веб-інтерфейс.

Продукти Insteon засновані на власній технології сітчастої мережі, яка дозволяє пристроям спілкуватися один з одним, використовуючи як радіочастотні сигнали, так і сигнали лінії електропередачі. Цей підхід із подвійним сигналом забезпечує підвищену надійність і діапазон у порівнянні з іншими технологіями домашньої автоматизації, які покладаються на один тип сигналу.

Мережа Insteon може підтримувати до 232 пристроїв, і кожен пристрій може діяти як ретранслятор для сигналів інших пристроїв у мережі. Це означає, що навіть

якщо один пристрій у мережі виходить з ладу або знаходиться поза зоною дії, решта мережі все ще може функціонувати належним чином.

На додаток до своїх апаратних продуктів, Insteon також пропонує набір програмного забезпечення (SDK), який дозволяє стороннім розробникам створювати власні програми для керування пристроями Insteon.

Деякі функції програми Insteon IoT включають:

- Дистанційно керуйте пристроями Insteon з будь-якої точки світу.
- Створюйте розклади та таймери для автоматичного керування освітленням, приладами тощо.
- Отримувати push-повідомлення, коли двері або вікна відкриваються або закриваються.
- Контролюйте споживання енергії приладами та освітленням[12].

Додаток Control4 – це потужний інструмент, який дає користувачам можливість керувати своїми пристроями та системами домашньої автоматизації з одного централізованого місця. Додаток надає користувачам різноманітні функції та параметри, які можна використовувати для налаштування їхнього досвіду, зокрема можливість створювати власні сцени та програми, отримувати доступ до налаштувань та інформації пристрою, отримувати сповіщення та сповіщення та багато іншого.

Відмінною особливістю програми Control4 є її сумісність з широким спектром пристроїв. Додаток можна використовувати зі смартфонами, планшетами, ноутбуками, настільними комп’ютерами і навіть Smart TV. Це означає, що користувачі можуть керувати своїми пристроями з будь-якого місця, якщо у них є підключення до Інтернету[13].

ThingSpeak – це програма Інтернету речей з відкритим вихідним кодом, яка дозволяє збирати дані з датчиків і пристроїв і візуалізувати їх у вигляді діаграм і графіків. ThingSpeak також надає API, який дозволяє дистанційно керувати пристроями та датчиками.

ThingSpeak був створений Mathworks, тією ж компанією, яка виробляє популярне програмне забезпечення MATLAB. ThingSpeak розроблено для

використання з MATLAB, але його також можна використовувати з іншими мовами програмування, такими як Python, Java та C++.

ThingSpeak – це безкоштовна програма, але існує обмеження в 8 мільйонів одиниць даних на місяць. Якщо вам потрібно більше, ніж це, ви можете підписатися на платну підписку, яка починається від 20 доларів США на місяць.

Однією з найпопулярніших функцій ThingSpeak є його здатність надсилати дані іншим сервісам, таким як Twitter, Facebook і Slack. Це дозволяє створювати «попередження» при дотриманні певних умов (наприклад, якщо температура перевищує певний поріг). ThingSpeak також можна використовувати для дистанційного керування пристроями. Наприклад, ви можете використовувати ThingSpeak, щоб увімкнути світло або запустити вентилятор, коли температура перевищує певний поріг[14].

Ubidots – це програма IoT, яка дозволяє користувачам збирати, зберігати та візуалізувати дані з підключених пристроїв. Платформа пропонує широкий спектр функцій і можливостей, що робить її потужним інструментом для керування пристроями та системами IoT та моніторингу.

Ubidots розроблено для роботи з різними пристроями та протоколами, що спрощує підключення нових пристроїв і платформ. Платформа підтримує протоколи MQTT і HTTP, що робить її сумісною з широким спектром пристроїв. Крім того, Ubidots пропонує REST API, який можна використовувати для підключення інших додатків і служб.

Ubidots – це хмарна платформа, що означає, що до неї можна отримати доступ з будь-якого місця, де є підключення до Інтернету. Платформа також масштабована, що означає, що її можна використовувати для керування великою кількістю пристроїв без проблем із продуктивністю.

Однією з найкорисніших функцій Ubidots є його здатність візуалізувати дані. Платформа пропонує ряд діаграм і графіків, які можна використовувати для моніторингу даних в режимі реального часу. Це надзвичайно корисно для визначення тенденцій і закономірностей у даних.

Ще одна корисна функція Ubidots - це система оповіщення. Це дозволяє користувачам налаштувати сповіщення, які спрацьовують при дотриманні певних

умов. Наприклад, сповіщення може спрацьовувати, коли температура перевищує певний поріг. Сповіщення можна надсилати електронною поштою або SMS, що робить їх дуже універсальними[15].

Losant – це проста у використанні, потужна та масштабована платформа, яка дає змогу швидко створювати та розгортати додатки IoT. Losant надає все необхідне для початку роботи з Інтернетом речей, включаючи візуальний конструктор робочих процесів, візуалізацію даних у реальному часі, керування пристроями тощо. Losant також є повністю розширюваним, тож ви можете додати власні власні функції або інтеграції за потреби.

Losant дозволяє легко збирати дані з пристроїв і датчиків в режимі реального часу. Потім ви можете використовувати ці дані для створення інформаційних панелей та візуалізацій, щоб отримати уявлення про вашу систему. Losant також надає потужні інструменти для керування пристроями та користувачами, а також для побудови складних робочих процесів.

Losant – чудова платформа для швидкого створення прототипів додатків IoT. Він також дуже добре масштабується, тому ви можете легко розгорнути свою програму на тисячах або навіть мільйонах пристроїв[16].

Initial State – це програма IoT, яка дозволяє користувачам збирати, візуалізувати й аналізувати дані зі своїх пристроїв і датчиків. Він включає в себе хмарну платформу для зберігання та обробки даних, а також мобільний додаток для візуалізації даних.

Ядром платформи Initial State є його хмарний сервер, який збирає дані з пристроїв і датчиків за допомогою різних плагінів та інтеграцій. Ці дані потім зберігаються в хмарі, щоб до них можна було отримати доступ з будь-якого місця. Бекенд також включає потужний механізм обробки, який дозволяє користувачам виконувати комплексний аналіз своїх даних.

Мобільний додаток використовується для візуалізації даних, зібраних бекендом. Він містить різноманітні діаграми та графіки, які можна налаштувати для відображення різних аспектів даних. Додаток також дозволяє користувачам налаштовувати сповіщення, щоб вони отримували сповіщення, коли виконуються певні умови (наприклад, коли датчик виявляє аномалію).

Initial State пропонує безкоштовний рівень обслуговування, який включає до 5 ГБ пам’яті на місяць і 10 мільйонів щомісячних подій (які визначаються як окремі показання з пристроїв або датчиків)[17].

Blynk – це платформа Інтернету речей (IoT), яка дозволяє користувачам створювати, підключатися та керувати своїми пристроями IoT за допомогою програми для смартфонів. Додаток забезпечує перегляд даних пристрою в режимі реального часу, а також можливість дистанційного керування ним.

Платформа Blynk складається з трьох основних компонентів:

- Програма Blynk.
- Сервер Blynk, який відповідає за керування зв’язком між програмою та пристроями IoT.
- Бібліотека Blynk, яка вбудована у мікропрограмне забезпечення пристрою IoT і обробляє зв’язок із сервером Blynk.

У кожному проекті користувачі можуть додавати різноманітні «віджети», які представляють різні типи даних або функціональність. Ці віджети потім пов’язуються з контактами на самому пристрої IoT.

Користувачі також можуть додавати тригери в свої проекти, щоб певні дії виконувались автоматично на основі конкретних умов.

Нарешті, користувачі можуть налаштувати сповіщення, щоб вони отримували сповіщення, коли виконуються певні умови[18].

Particle – це потужна та проста у використанні платформа додатків IoT, яка дає змогу підключатися, керувати та керувати своїми пристроями з хмари. За допомогою Particle ви можете створювати підключені додатки для широкого кола галузей і варіантів використання, включаючи домашню автоматизацію, управління енергією, підключені транспортні засоби, промислову автоматизацію тощо.

Particle надає все, що вам потрібно для початку розробки IoT, включаючи потужну систему керування пристроями, просту у використанні IDE розробки та надійний набір бібліотек та інструментів. Particle також пропонує різноманітні варіанти підключення відповідно до ваших потреб, включаючи стільниковий зв’язок (3G/LTE), Wi-Fi, Bluetooth LE (BLE), Ethernet, LoRaWAN тощо.

За допомогою Particle’s Device Cloud ви можете легко підключити свої пристрої до Інтернету та почати створювати підключені додатки. Device Cloud – це безпечний брокер MQTT, який підтримує зашифровані з’єднання та надає вбудовані функції керування пристроєм, такі як оновлення мікропрограми по повітрю (OTA) та віддалений контроль пристрою[19].

Програма Helium – це потужний інструмент, який дозволяє користувачам віддалено підключатися та керувати своїми пристроями. Він надає різноманітні функції та опції, які роблять його важливим інструментом для тих, хто хоче відстежувати свої пристрої. Ось деякі з основних функцій і переваг програми Helium:

- **Постійне підключення**: додаток Helium надає користувачам можливість підключати свої пристрої до Інтернету, дозволяючи їм керувати ними віддалено. Це особливо корисно для відстеження пристроїв, які не завжди знаходяться в одному місці, таких як камери безпеки або датчики.
- **Простота використання**: додаток Helium розроблено так, щоб бути простим у використанні навіть для тих, хто не розбирається в техніці. Він має зручний інтерфейс, який спрощує підключення та керування пристроями.
- **Масштабування**: додаток Helium є масштабованим, тобто він може підтримувати велику кількість пристроїв. Це ідеально підходить для підприємств або будинків із кількома пристроями, які потрібно контролювати.
- **Безпека**: програма Helium використовує безпечні протоколи для забезпечення захисту даних. Це включає як дані, що зберігаються, так і дані, що передаються, що гарантує, що ваша інформація захищена від хакерів або інших зловмисників[20].

Exosite – це платформа додатків IoT, яка дозволяє підприємствам створювати, розгортати та керувати додатками IoT. Він пропонує низку функцій і можливостей, які дозволяють підприємствам збирати дані з пристроїв, відстежувати й керувати пристроями, а також керувати парком пристроїв. Він також надає API, які дозволяють підприємствам інтегрувати свої програми IoT з іншими бізнес-системами.

Платформа Exosite побудована на основі низки відкритих стандартів, включаючи Open Data Protocol (ODP), Data Access Protocol (DAP) і Security Assertion Markup Language (SAML). Це дозволяє підприємствам легко підключати свої пристрої та дані до платформи Exosite і полегшує інтеграцію Exosite з іншими бізнес-системами.

Ключові особливості Exosite:

- **Керування пристроями**: дозволяє підприємствам надавати пристрої, відстежувати стан пристрою та оновлювати мікропрограму.
- **Керування даними**: збирає дані з пристроїв і зберігає їх у захищеній базі даних. Доступ до даних можна отримати через API або візуалізувати за допомогою інформаційних панелей.
- **Механізм правил**: автоматизує дії на основі даних, зібраних з пристроїв. Наприклад, правила можна використовувати для надсилання сповіщень або сповіщень, коли виконуються певні умови.
- **Керування підключенням**: керує з'єднанням між пристроями та платформою Exosite. Це включає підтримку кількох протоколів і керування мережевими параметрами.
- **Безпека**: забезпечує автентифікацію та авторизацію для пристроїв і користувачів, а також шифрує дані під час передачі за допомогою SSL/TLS [21].

Carriots – це прикладна платформа Інтернету речей (IoT), яка дозволяє користувачам підключатися, збирати й аналізувати дані з пристроїв і датчиків. Carriots надає розробникам набір інструментів для створення додатків IoT, включаючи платформу потокових даних у реальному часі, структуру розробки додатків і набір API. Carriots також пропонує набір хмарних сервісів для керування пристроями, користувачами та додатками.

Платформа Carriots складається з трьох основних компонентів:

- Платформи потокових даних у реальному часі.
- Фреймворк розробки додатків.
- Хмарна система керування пристроями.

Платформа потокової передачі даних в режимі реального часу отримує дані з пристроїв і датчиків у режимі реального часу і робить їх доступними для додатків через API. Фреймворк розробки додатків дозволяє розробникам створювати програми IoT за допомогою API Carriot. Хмарна система керування пристроями надає набір інструментів для керування пристроями, користувачами та додатками.

API Carriot дозволяє розробникам отримувати доступ до даних, зібраних платформою, щоб створювати власні програми IoT. Розробники можуть використовувати API для створення нових пристроїв, додавання нових датчиків до наявних пристроїв, збору даних з пристроїв або датчиків або запиту історичних даних, що зберігаються на платформі[22].

Xively – це програма Інтернету речей, яка дозволяє компаніям і приватним особам підключати пристрої та продукти до Інтернету, збирати з них дані, а потім переглядати ці дані в режимі реального часу. По суті, це допомагає вам перетворити ваші фізичні продукти в цифрові, які можна контролювати та керувати дистанційно.

За допомогою Xively ви можете налаштувати пристроїв і продуктів у режимі реального часу. Ви можете використовувати Xively для моніторингу продуктивності ваших пристроїв і продуктів у режимі реального часу. Це корисно для виявлення проблем на ранній стадії, ще до того, як вони стануть серйозними проблемами. За допомогою Xively ви також можете дистанційно керувати пристроями. Наприклад, ви можете увімкнути/вимкнути пристрій або змінити його налаштування з будь-якої точки світу. Ці дані можна використовувати для покращення ваших продуктів чи послуг або просто для того, щоб отримати уявлення про те, як вони використовуються.

Ви також можете використовувати Xively для обміну даними з іншими – будь то колеги, партнери чи клієнти. Наприклад, ви можете поділитися показаннями датчика в реальному часі з клієнтом, щоб він міг бачити, як працює продукт у режимі реального часу[23]. Компанія Xively нещодавно була поглинута Google.

1.3. Розробки Blockchain в IoT

Потенціал технології Blockchain часто обговорюється в контексті Bitcoin та інших крипто валют. Однак потенційні застосування Blockchain виходять далеко за

межі світу фінансів. Однією з областей, де технологія Blockchain є особливо перспективною, є Інтернет речей.

Пристрої Інтернету речей – це фізичні об’єкти, які під’єднані до Інтернету та здатні збирати та обмінюватися даними. Вони стають все більш поширеними як у нашому особистому, так і в професійному житті. Наприклад, зараз у багатьох із нас вдома є розумні пристрої, які дозволяють дистанційно керувати температурою та освітленням. На робочому місці датчики IoT можна використовувати для відстеження рівня запасів і моніторингу продуктивності співробітників.

Дані, які генеруються пристроями IoT, неймовірно цінні. Однак він також дуже вразливий до злому та інших порушень безпеки. Ось тут на допомогу приходить Blockchain.

Blockchain – це розподілена база даних, яка за своєю конструкцією безпечна. Це означає, що хакерам неймовірно важко підробити дані, які зберігаються в Blockchain.

Blockchain – це цифровий реєстр транзакцій, який дублюється і розподіляється по всій мережі комп’ютерних систем на Blockchain. Кожен блок у ланцюжку містить певну кількість транзакцій, і щоразу, коли відбувається нова транзакція, запис цієї транзакції додається до книги кожного учасника. Децентралізована база даних, керована кількома учасниками, відома як Distributed Ledger Technology (рис. 1.1) (технологія розподіленого гаманця) (DLT)[24].

Рисунок 1.1 – Властивості Distributed Ledger Technology (DLT)

Одна з ключових відмінностей між типовою базою даних і Blockchain полягає в тому, як структуровані дані. Blockchain збирає інформацію разом у групи, відомі як блоки, які містять набори інформації. Блоки мають певну ємність для зберігання даних і, коли вони заповнені, закриваються і пов’язуються з раніше заповненим блоком, утворюючи ланцюжок даних, відомий як Blockchain. Вся нова інформація, яка випливає з того, що щойно доданий блок компілюється у новостворений блок, який потім також буде додано до ланцюжка після заповнення.

База даних зазвичай структурує свої дані в таблиці, тоді як Blockchain, як випливає з його назви (*Blockchain українською – ланцюг блоків*), структурує свої дані на частини (блоки), які з’єднані разом. Ця структура даних створює незворотну структуру даних, яка реалізована в децентралізованому вигляді та зав’язана на часових відмітках. Коли блок заповнюється, він фіксується і стає частиною цієї структури. Кожному блоку в ланцюжку надається точна відмітка часу, коли він додається до ланцюжка.

Мета Blockchain – дозволити записувати та поширювати цифрову інформацію, але не редагувати. Таким чином, Blockchain є основою для незмінних реєстрів або записів транзакцій, які не можна змінити, видалити або знищити[25].

Blockchain складається з трьох важливих концепцій: блоків, вузлів і майнерів.

Блоки. Кожен ланцюг складається з блоків. У свою чергу блоки мають наступну структуру:

- Дані, що містяться у блоку.
- Число з великою кількістю символів, яке називається nonce.
- 256 бітний хеш, який генерується з даних блоку та nonce.

Коли створюється перший блок ланцюга, nonce генерує криптографічний хеш. Дані в блоці вважаються підписаними і назавжди прив’язані до nonce та хешу.

Майнери. Майнери – це люди, які підтверджують транзакції на Blockchain, вирішуючи складні математичні задачі.

У Blockchain кожен блок має свій унікальний одноразовий номер і хеш, але також посилається на хеш попереднього блоку в ланцюжку, тому видобуток блоку нелегкий, особливо у великих ланцюгах.

Коли майнер вирішує проблему, він підтверджує блок транзакцій. Потім цей блок додається до Blockchain, який є загальнодоступним записом усіх транзакцій. Майнери винагороджуються за роботу з криптовалютою.

Внесення змін до будь-якого блоку раніше в ланцюжку вимагає повторного майнінгу не лише блоку зі зміною, а й усіх блоків, які йдуть після. Ось чому надзвичайно важко маніпулювати технологією Blockchain. Вузол - це комп'ютер, який підключений до мережі Blockchain. Коли вузол підключений до мережі, він може перевіряти та передавати транзакції. Кожен вузол мережі має копію Blockchain, яка автоматично завантажується, коли вузол приєднується до мережі.

Вузли(nodes). Вузол – це комп'ютер, який підключений до мережі Blockchain. Коли вузол підключений до мережі, він може перевіряти та передавати транзакції. Кожен вузол мережі має копію Blockchain, яка автоматично завантажується, коли вузол приєднується до мережі.

Вузли відіграють важливу роль у підтримці децентралізації мережі Blockchain. Якби не було вузлів, не було б мережі Blockchain. Коли користувач хоче додати

транзакцію в Blockchain, він повинен спочатку транслювати транзакцію в мережу вузлів. Потім вузли підтвердять транзакцію та додадуть її до своєї копії Blockchain.

Чим більше вузлів у мережі, тим більш децентралізована мережа. Це тому, що одному об’єкту було б дуже важко контролювати всі вузли в мережі. Навіть якби суб’єкт зміг контролювати більшість вузлів, було б дуже важко скоординувати атаку на мережу.

Децентралізація мережі Blockchain є однією з її ключових переваг. Це робить мережу більш безпечною та стійкою до атак[26].

Використання Blockchain для захисту даних із пристроїв Інтернету речей все ще знаходиться на ранній стадії. Проте вже є ряд стартапів, які працюють над цією проблемою.

Однією з компаній, яка використовує Blockchain для аналізу даних Інтернету речей, є Filament. Filament розробила децентралізовану сенсорну мережу, яку можна використовувати для відстеження та керування активами в режимі реального часу. Дані від датчиків зберігаються в Blockchain, а Filament надає панель інструментів, яка дозволяє користувачам переглядати та аналізувати дані.

Filament. Filament – це компанія, що використовує Blockchain та яка дозволяє іншим компаніям безпечно та економічно створювати децентралізовані програми в Інтернеті речей.

Бачення компанії полягає в тому, щоб створити нову еру демократії пристроїв, коли будь-який фізичний об’єкт може мати автономну економічну ідентичність і брати участь у децентралізованих програмах і ринках.

Продукт компанії Filament (рис 1.2) – це платформа Інтернету речей (IoT), заснована на Blockchain, яка дозволяє компаніям створювати децентралізовані програми (DApps) і підключати пристрої до Blockchain. Платформа надає набір інструментів для розробників, включаючи шлюз IoT, вбудоване програмне забезпечення та інтерфейс прикладного програмування (API). Filament також пропонує децентралізований магазин додатків, який називається Filament Market, де розробники можуть купувати та продавати програми та дані IoT[27].

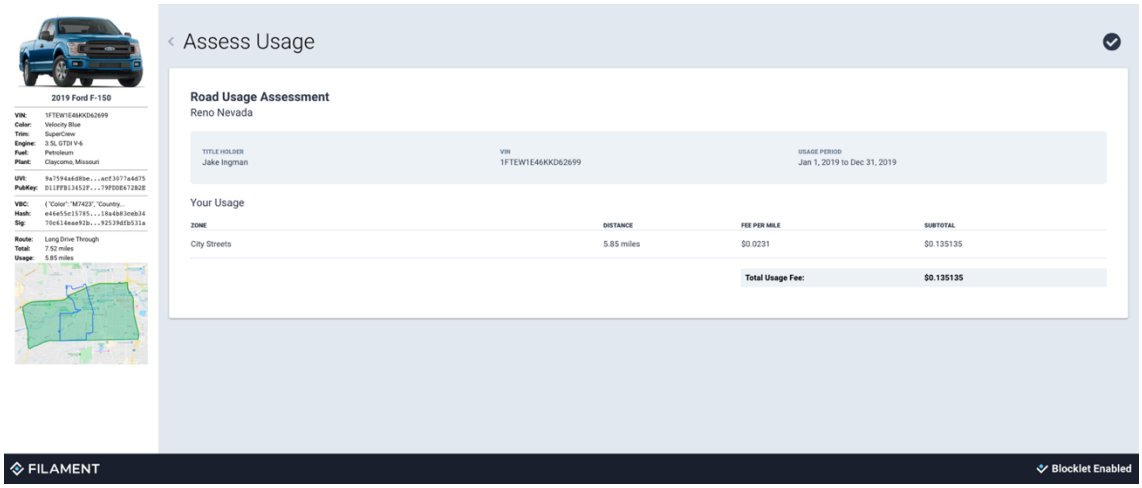


Рисунок 1.2 – Приклад продукту компанії Filament

Платформа Filament побудована на основі Blockchain і використовує InterPlanetary File System (IPFS) для зберігання даних.

InterPlanetary File System (IPFS) – це децентралізована однорангова мережа обміну файлами, яка дозволяє користувачам зберігати дані та обмінюватися ними розподіленим способом.

Така розподілена файлова система використовує модель блочного зберігання адресованого вмісту. Це означає, що кожна частина даних зберігається в блоці з унікальним ідентифікатором. Коли користувач хоче отримати доступ до файлу, він надає ідентифікатор потрібного блоку. Потім мережа IPFS отримує запитаний блок з будь-якого вузла, на якому він збережений (рис. 1.3).

Така конструкція робить IPFS стійкою до втрати або пошкодження даних, оскільки кожен блок можна перевірити за його ідентифікатором. Крім того, блоки можна реплікувати по мережі для забезпечення доступності та підвищення продуктивності.

IPFS також має вбудовану підтримку версій файлів і каталогів. Це означає, що користувачі можуть легко відстежувати різні версії файлу або папки та повертатися до старих версій, якщо це необхідно. Це особливо корисно для спільних робочих процесів або при роботі з великими обсягами даних.

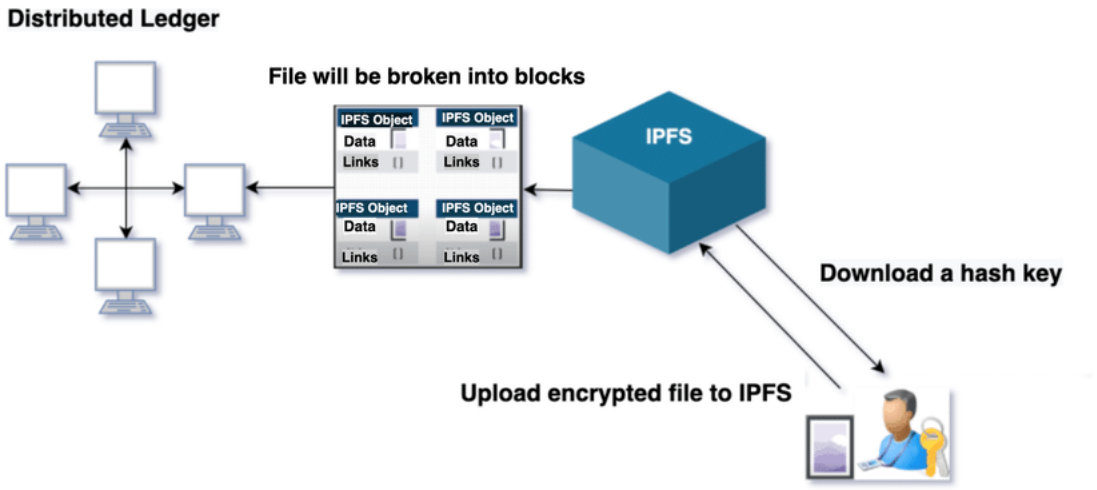


Рисунок 1.3 – Схема роботи протоколу IPFS

Нарешті, IPFS підтримує шифрування даних у стані спокою та передачі. Це гарантує, що лише авторизовані користувачі можуть отримати доступ до конфіденційної інформації, і що всі комунікації захищені від прослуховування[28].

Шлюз IoT від Filament – це апаратний пристрій, який з'єднує пристрої з Blockchain та IPFS. Шлюз дозволяє пристроям безпечно спілкуватися один з одним і з додатками на основі Blockchain. Filament Market – це децентралізований магазин додатків, який дозволяє розробникам купувати та продавати програми та дані IoT. Вбудоване програмне забезпечення Filament дозволяє розробникам створювати програми на основі Blockchain, які запускаються на пристроях. Filament API дозволяє розробникам взаємодіяти з платформою Filament і створювати DApps.

Платформа Filament розроблена, щоб забезпечити безпечний і масштабований спосіб підключення пристроїв до Blockchain. Шлюз IoT платформи, вбудоване програмне забезпечення та децентралізований магазин додатків надають розробникам інструменти, необхідні для створення DApps та підключення пристроїв до Blockchain. Filament API дозволяє розробникам взаємодіяти з платформою Filament і створювати DApps. Filament Market – це децентралізований магазин додатків, який дозволяє розробникам купувати та продавати програм та дані IoT[29].

IOTA. IOTA – це технологія розподіленого реєстру, яка забезпечує комунікацію та розрахунки за транзакціями між пристроями в Інтернеті речей (IoT). IOTA використовує структуру орієнтованого ациклічного графа (DAG), що забезпечує масштабованість і безперебійні мікроплатежі (рис. 1.4). IOTA також

використовує новий алгоритм консенсусу, який дозволяє йому бути безпечним, не вимагаючи майнерів.

Платформа IOTA розроблена для забезпечення майбутнього економіки Інтернету речей, забезпечуючи безпечне міжмашинне спілкування (M2M) і створюючи основу для нових додатків і бізнес-моделей. З IOTA немає потреби в централізованих посередниках, таких як банки або розрахункові палати, для розрахунків за операціями. Замість цього транзакції перевіряються консенсусом учасників мережі. Це дозволяє майже миттєво розрахуватися за низькою ціною.

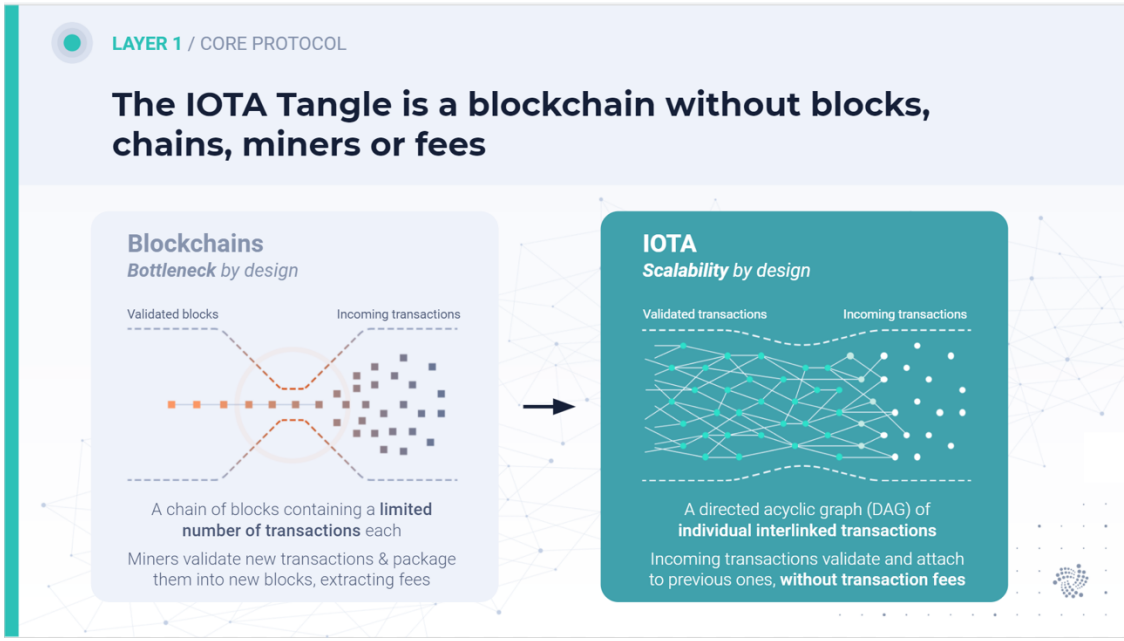


Рисунок 1.4 – Використання структури орієнтованого ациклічного графа у протоколі технології IOTA

Власний маркер IOTA, MIOTA, використовується для живлення мережі і може обмінюватися між пристроями-учасниками. Транзакції в мережі IOTA не відчуваються через її структуру DAG. Замість того, щоб майнери перевіряли блоки транзакцій, як у традиційних мережах блокчейн, кожен вузол в IOTAnet перевіряє дві попередні транзакції. Цей процес називається «вибір наконечника».

Щоб запобігти атак подвійних витрат, коли хтось витрачає свої монети більше одного разу, кожна транзакція має посилатися на дві попередні транзакції в так званому «спрямованому ациклічному графі» або DAG. Оскільки майнери не перевіряють блоки транзакцій, як традиційні блокчейни, цей процес називається «консенсус за ставками» або просто «консенсус зацікавлених сторін». Для того, щоб хтось міг атакувати мережу, йому знадобиться 51% контролю, що неймовірно

малоймовірно, враховуючи, що на даний момент існує понад 27 мільйонів унікальних адрес.

Tangle, як його називають, теоретично може масштабуватися до нескінченної кількості одночасних транзакцій за умови, що достатньо пристроїв готові діяти як валідатори. Наразі мережа може обробляти близько 500-600 TPS(транзакцій на секунду) з дуже невеликими проблемами. Команда також створила так званий алгоритм координат, який зрештою повністю усуне потребу в централізації. Коли це станеться, IOTA буде повністю децентралізована[30].

Waltonchain. Waltonchain – це блокчейн-проект, метою якого є створення екосистеми для Інтернету речей (IoT). Проект використовує унікальну комбінацію технології блокчейн і радіочастотної ідентифікації (RFID) для відстеження та керування фізичними об’єктами в реальному світі. Waltonchain має намір надати повне рішення для управління ланцюгом поставок, логістики та боротьби з підробками.

Екосистема Waltonchain (рис. 1.5) включає три компоненти: батьківський ланцюг Waltonchain, дочірні ланцюжки та токени. Батьківський ланцюг Waltonchain – це загальнодоступний блокчейн, який використовує алгоритм консенсусу Proof-of-Stake (PoS). Дочірні ланцюги – це приватні блокчейни, які підключені до батьківського ланцюга. Кожен дочірній ланцюжок має свій власний токен. Ці токени можна використовувати для оплати товарів і послуг у дочірньому ланцюжку або обміняти на інші активи в батьківському ланцюжку.

Мета Waltoncoin – створити ефективну децентралізовану екосистему, де фізичні об’єкти можуть бути легко інтегровані з технологією блокчейн. Це дозволить підприємствам ефективніше керувати своїми ланцюжками поставок і зменшити витрати, пов’язані з підробками та шахрайством. Крім того, відстежуючи фізичні об’єкти за допомогою міток RFID, підприємства зможуть отримати цінну інформацію про свою діяльність.

Основні особливості Waltoncoin включають:

- Унікальний гібридний алгоритм консенсусу, який поєднує PoS з Proof-of-Work (PoW).
- Підтримка смарт-контрактів.

- Транзакції між дочірніми ланцюжками.
- Децентралізована біржа.

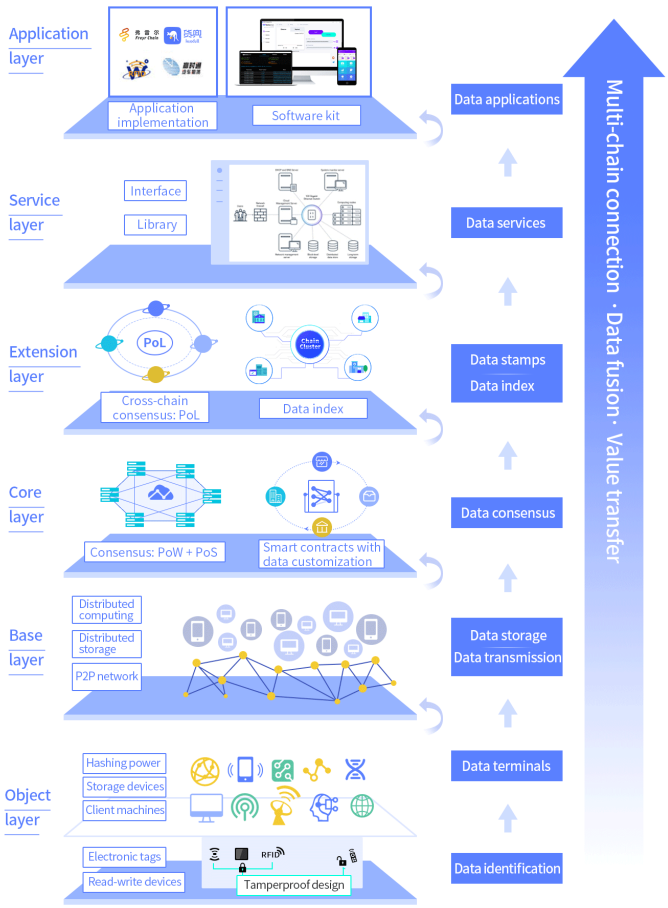


Рисунок 1.5 – Структура екосистеми Waltonchain

Основою системи є RFID – радіочастотні ідентифікаційні мітки, які розміщуються на продуктах або пристроях під час виробництва або складання. Ці мітки містять інформацію про продукт, включаючи місце, час, дату тощо. Ці дані разом з іншими деталями, такими як координати GPS, зберігаються. в блокчейнах. Ці блокчейни можуть бути як приватними, так і загальнодоступними залежно від потреби. Тепер ці теги не можна змінювати або видаляти, що забезпечує безпеку, а також прозорість у всьому процесі. Також може бути кілька рівнів безпеки, як-от пароль, qr-код тощо відповідно до потреб[31].

Відчутні переваги використання Waltoncoin:

- **Відстеження**: дуже легко відстежувати продукт на кожному етапі від виробництва до досягнення кінцевого клієнта/клієнта. Це забезпечує автентичність, а також прозорість на кожному рівні, завдяки чому створюється довіра між постачальником/виробником і клієнтом/клієнтом.

- **Безпека даних**: оскільки дані зберігаються у вигляді блоків (пристроїв IoT), зламати цю систему стає дуже важко. Крім того, як тільки дані надходять в систему, стає майже неможливим змінити їх, що ще більше посилює захист.
- **Зменшення витрат**: скорочуючи час, необхідний на кожному етапі до остаточної доставки, можна досягти відчутної економії грошей і часу, тим самим додаючи до прибутку компанії.

Ambrosus. Ambrosus – це децентралізована мережа для відстеження та забезпечення якості продукції. Мережа працює на основі токена Ambrosus (AMB) і використовує технологію Blockchain і розумні контракти, щоб забезпечити захист від несанкціонованого доступу про шлях продукту від походження до продажу. Це дозволяє підприємствам і споживачам перевіряти справжність і якість продукції, а також відстежувати їх по всьому ланцюжку поставок.

Мережа Ambrosus (рис. 1.6) складається з трьох основних компонентів: датчиків, шлюзів і платформи зберігання даних Amber. Датчики використовуються для збору даних про продукт, таких як температура або вологість. Потім ці дані зберігаються на платформі Amber, яка є децентралізованою базою даних, яка використовує технологію Blockchain. Шлюзи використовуються для підключення пристроїв (наприклад, датчиків) до платформи Amber.

Рисунок 1.6 – Архітектура Ambrosus

Мережа Ambrosus використовує технологію Blockchain і смарт-контракти для створення надійного запису шляху продукту від походження до продажу. Коли датчик збирає дані про продукт (наприклад, температура або вологість), ці дані зберігаються на платформі Amber. Потім дані перевіряються шлюзами, які використовуються для підключення пристроїв (наприклад, датчиків) до платформи Amber. Після перевірки дані не можуть бути змінені чи видалені. Це гарантує, що підприємства та споживачі можуть довіряти, що інформація про шлях продукту є точною та надійною.

На додаток до забезпечення прозорості в усьому ланцюжку поставок, Ambrosus також прагне підвищити ефективність шляхом автоматизації таких процесів, як контроль якості або управління запасами. Наприклад, якщо датчик виявляє, що партія їжі піддавалася впливу занадто сильного тепла, Ambrosus може автоматично викликати сповіщення, щоб можна було вжити заходів для запобігання псування до того, як воно станеться. Зрештою, це може призвести до зменшення марнотратства в усьому ланцюжку поставок і підвищення рівня задоволеності клієнтів для підприємств, які використовують рішення Ambrosus[32].

1.4. Аналіз існуючих датчиків IoT, вхідні дані датчиків

Існує велика різноманітність датчиків, які використовуються в Інтернеті речей (IoT), кожен зі своїми унікальними можливостями та перевагами.

Температурні датчики є одним з найбільш часто використовуваних типів датчиків у додатках IoT. Їх можна використовувати для вимірювання температури кімнати, обладнання чи навіть тіла людини. Датчики температури (рис. 1.7) доступні в різних формах, включаючи термопари, резистивні температурні детектори (RTD) і датчики на основі напівпровідників[33].

Рисунок 1.7 – Температурний датчик LMT84LP від компанії Texas Instruments

На ринку представлено багато різних типів датчиків температури, кожен з яких має свої переваги та недоліки. Може бути важко визначити, який датчик найкраще підходить для конкретного застосування.

Одним з найважливіших критеріїв при виборі датчика температури є точність датчика. Датчики температури можуть бути дуже точними, але точність може відрізнятися в залежності від типу датчика. Наприклад, термопари, як правило, більш точні, ніж RTD. Однак RTD є більш стабільними з часом, тому вони можуть бути кращим вибором для додатків, де важлива довгострокова стабільність.

Іншим важливим фактором є діапазон температур, який може вимірювати датчик. Деякі датчики призначені лише для вимірювання невеликого діапазону температур, тоді як інші можуть вимірювати широкий діапазон температур. Наприклад, інфрачервоні датчики можуть вимірювати дуже високі температури, але вони не так точні, як інші типи датчиків при нижчих температурах.

Нарешті, потрібно враховувати вартість датчика. Датчики температури можуть сильно відрізнятися за ціною, від кількох доларів до сотень доларів. Загалом, більш дорогі датчики будуть точнішими і мають більш широкий діапазон температур. Однак вам потрібно збалансувати вартість датчика з потребами вашого застосування[34].

Датчики вологості – це ще один тип датчиків, який зазвичай використовується в додатках IoT. Вони використовуються для вимірювання кількості водяної пари в повітрі і доступні в різних формах, включаючи ємнісні, резистивні та оптичні датчики.

На ринку існує безліч різних типів датчиків вологості, кожен з яких має свої переваги і недоліки.

Ємнісні датчики вологості є найпопулярнішим типом датчиків вологості на ринку. Вони відносно недорогі і дуже точні. Ємнісні датчики вологості працюють шляхом вимірювання зміни ємності, що відбувається, коли водяна пара поглинається датчиком.

Резистивні датчики (рис. 1.8) вологості є ще одним популярним типом датчиків вологості. Вони менш дорогі, ніж ємнісні датчики вологості, але вони не

такі точні. Резистивні датчики вологості працюють, вимірюючи зміну опору, яка виникає, коли водяна пара поглинається датчиком.

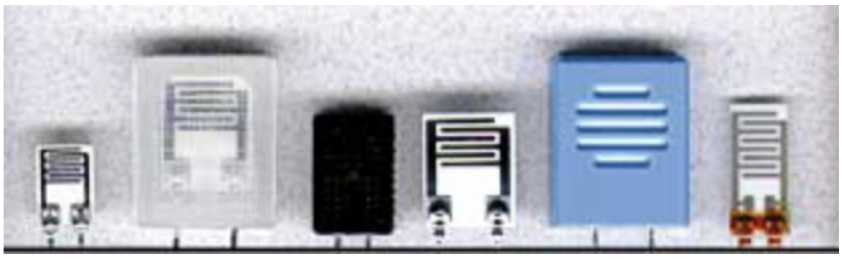


Рисунок 1.8 – Приклад датчиків для вимірювання вологості повітря

Оптичні датчики вологості є найдорожчим типом датчиків вологості. Це найточніший тип датчика вологості, але вони також найскладніші у використанні. Оптичні датчики вологості працюють шляхом вимірювання зміни показника заломлення водяної пари[35].

Датчики тиску використовуються для вимірювання тиску газу або рідини і доступні в різних формах, включаючи п’єзоелектричні, п’єзорезистивні та ємнісні датчики.

П'єзоелектричні датчики (рис. 1.9) є найпоширенішим типом датчиків тиску. Вони працюють шляхом перетворення тиску в електричний сигнал. Резистивні датчики працюють, вимірюючи зміну опору при застосуванні тиску. Ємнісні датчики працюють, вимірюючи зміну ємності при застосуванні тиску.

Рисунок 1.9 – П'єзоелектричний датчик CP211 для вимірювання тиску від компанії Meggitt

Кожен тип датчиків має свої переваги і недоліки. П'єзоелектричні датчики є найточнішим типом датчиків, але вони також і найдорожчі. Ємнісні датчики менш точні, ніж п’єзоелектричні, але вони дешевші. Резистивні датчики - найменш точний тип датчика, але вони найдешевші[36].

Датчики потоку використовуються для вимірювання швидкості потоку газу або рідини і доступні в різних формах, включаючи датчики диференціального тиску, позитивного зміщення та турбінні датчики.

Датчики диференціального тиску працюють шляхом вимірювання падіння тиску через обмеження на шляху потоку. Датчики потоку з позитивним зміщенням вимірюють об’єм рідини, яка проходить через них, а турбінні датчики потоку вимірюють швидкість рідини, коли вона проходить через датчик.

Датчики диференціального тиску добре підходять для вимірювання потоку рідин, газів і парів. Вони доступні в різних типах, включаючи діафрагми, трубки Вентурі та трубки Піто. Датчики диференціального тиску можна використовувати для вимірювання швидкості потоку від кількох мілілітрів на хвилину до кількох тисяч літрів на хвилину.

Датчики потоку з припливним зміщенням ідеально підходять для вимірювання потоку в’язких рідин, таких як масла та хімікати. Вони працюють, вимірюючи об’єм рідини, що проходить через них. Датчики потоку з припливним зміщенням доступні в різних типах, включаючи поршневі витратоміри, зубчасті витратоміри та дискові витратоміри[37].

Датчики руху використовуються для виявлення руху об’єктів і доступні в різних формах, включаючи ультразвукові датчики, мікрохвильові датчики та фотоелектричні датчики.

Найпоширенішим типом датчика руху є інфрачервоний (ІЧ) датчик(рис 1.10), який використовує інфрачервоне світло для виявлення руху. ІЧ-датчики працюють, виявляючи зміни в кількості інфрачервоного світла в зоні. Коли щось рухається в зоні, воно відбиває більше ІЧ-світла назад до датчика, що викликає тривогу.

Рисунок 1.10 – Інфрачервоний датчик руху LX-402 від компанії Optex

Ультразвукові датчики випромінюють високочастотні звукові хвилі та виявляють відображення від об’єктів у цій зоні. Коли щось рухається, воно відбиває більше звукових хвиль назад до датчика, що викликає тривогу. Ультразвукові датчики використовуються в охоронних системах і охоронній сигналізації.

Мікрохвильові датчики випромінюють мікрохвильове випромінювання та виявляють відображення від об’єктів у цьому районі. Коли щось рухається, воно відбиває більше мікрохвиль назад до датчика, що викликає тривогу.

Фотоелектричні датчики використовують світлочутливий елемент для виявлення руху. Коли щось рухається в зоні, воно відбиває більше світла назад до датчика, що викликає тривогу[38].

Датчики наближення використовуються для виявлення присутності об’єктів і доступні в різних формах, включаючи інфрачервоні, ультразвукові та магнітні датчики.

Датчики наближення можна розділити на два основних типи: активні та пасивні. Активні датчики випромінюють сигнал і вимірюють відбиття, а пасивні датчики визначають зміни в електромагнітному полі.

Найпоширенішим типом датчика наближення є інфрачервоний датчик. Ці датчики працюють шляхом виявлення інфрачервоного випромінювання, яке випромінює об’єкт.

Ультразвукові датчики є іншим типом датчиків наближення. Ці датчики працюють, випромінюючи ультразвукові звукові хвилі та вимірюючи відбиття.

Іншим типом датчиків наближення є магнітні датчики(рис 1.11). Ці датчики працюють, виявляючи зміни в магнітному полі[39].

Рисунок 1.11 – Магнітний датчик наближення MPSR03C1A10 від компанії C&K

Датчики світла використовуються для вимірювання інтенсивності світла і доступні в різних формах, включаючи фотодіоди, фототранзистори та фоторезистори.

Одним з найпоширеніших типів світлових датчиків є фотодіоди. Фотодіоди – це напівпровідникові прилади, які перетворюють світло в електричний струм. Вони широко використовуються в різноманітних застосуваннях, включаючи цифрові камери, оптоволоконний зв'язок і перетворення сонячної енергії.

Інший тип датчика освітленості – фототранзистор. Вони подібні до фотодіодів, але вони більш чутливі до світла і можуть використовуватися в різних програмах, включаючи виявлення світла, вимірювання інтенсивності світла та активацію перемикача світла.

Ще одним типом світлового датчика є фоторезистор (рис. 1.12). Фоторезистори – це напівпровідникові прилади, які змінюють свій опір у відповідь на світло. Вони використовуються в різних програмах, включаючи виявлення світла, вимірювання інтенсивності світла та активацію вимикача світла[40].

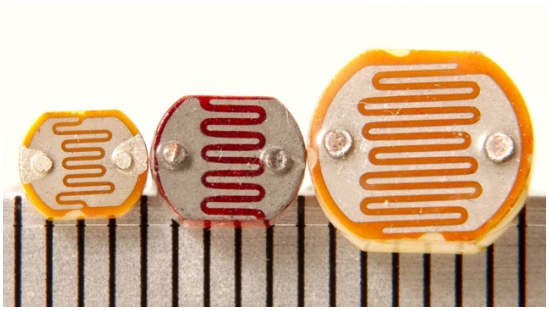


Рисунок 1.12 – Фоторезистори різних розмірів

Звукові датчики використовуються для перетворення звукових хвиль в електричні сигнали і доступні в різних формах, включаючи мікрофони та п’єзоелектричні датчики.

Існує два основних типи звукових датчиків: мікрофонні та п’єзоелектричні датчики. Мікрофони перетворюють звукові хвилі в електричні за допомогою діафрагми, яка вібрує, коли вона стикається зі звуковими хвилями. П’єзоелектричні датчики, з іншого боку, використовують п’єзоелектричний кристал для перетворення звукових хвиль в електричні сигнали.

П’єзо-звукові датчики (рис. 1.13) дуже чутливі до навколишнього середовища, і якість звуку зменшиться, якщо датчик знаходиться не в ідеальному положенні[41].

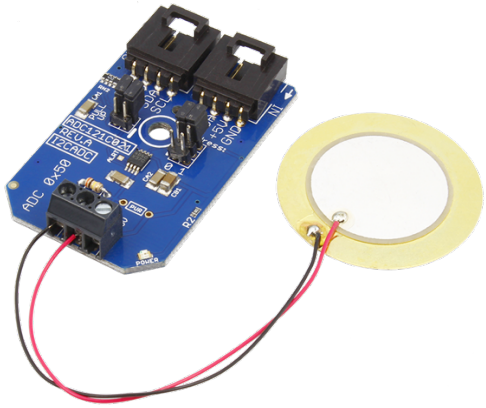


Рисунок 1.13 - П’єзо-звуковий датчик All I2C Mini від компанії nodeLynk

Хімічні датчики використовуються для виявлення присутності хімічних речовин і доступні в різних формах, включаючи датчики газу, датчики pH, які використовуються для вимірювання кислотності або лужності розчину, датчики кисню, які використовуються для вимірювання концентрації кисню в газі або рідині, і датчики вуглекислого газу, які використовуються для вимірювання концентрації вуглекислого газу в повітрі.

Найпоширенішим типом хімічних датчиків є газовий датчик, який використовується для виявлення присутності газів у навколишньому середовищі.

Хімічні датчики зазвичай виготовляються з матеріалу, чутливого до хімічної речовини, що представляє інтерес. Матеріалом може бути метал, напівпровідник або полімер. Тип використовуваного матеріалу залежить від хімічної речовини, яку потрібно виявити, і бажаної чутливості датчика. Наприклад, датчики газу з оксиду металу виготовлені з оксиду металу, чутливого до газу, який цікавить. Оксид металу зазвичай наносять на керамічну підкладку, а електричний опір оксиду металу вимірюють для визначення концентрації газу. Напівпровідникові газові датчики виготовлені з напівпровідника, наприклад кремнію, чутливого до газу, який цікавить. Для визначення концентрації газу вимірюють електропровідність напівпровідника. Полімерні газові датчики виготовлені з полімеру, чутливого до газу, який цікавить. Для визначення концентрації газу вимірюють електричний опір полімеру[42].

Існує широкий спектр інших типів датчиків, які використовуються в додатках IoT, включаючи датчики GPS, сканери штрих-кодів і RFID-мітки.

1.5. Вимоги нормативних документів для комерційних будівель

Для побудови та використання будівель у будь-яких цілях, комерційних чи житлових, необхідно притримуватися норм, які зазначені у ДБН В.2.2-9-2009[43]. Цей документ описує основні положення про експлуатацію будівель. У ньому зазначені норми безпеки для людей та споживання ресурсів будівлею.

Для того, щоб будівлю можна було використовувати потрібно пройти декілька перевірок та подати необхідні документи на розгляд до Державної інспекції архітектури та будівництва, Санітарно-епідеміологічної станції та Державної служби України з надзвичайних ситуацій.

У випадку виробничих/комерційних будівель під час перевірки власник чи уповноважена особа має надати робочій комісії на перевірку ряд документів:

- Проектну документацію.
- Інформацію про організації, що брали участь у будівництві.
- Документи на матеріали, конструкції, вироби, обладнання.

  Після отримання усіх документів та їх перевірки на місці базування органу створюється спеціальна комісія, яка має відвідати об’єкт та перевірити необхідні умови на місці. Комісія перевіряє наступні умови:

- Відповідність прийнятих рішень завершеного будівництва узгодженому проекту та вимогам охорони праці, СЕС та ін.
- Відповідність робіт нормативної документації.
- Відповідність документації нормативним вимогам.

  Нормативні вимоги вказані у нормативних документах, які затверджують вищезгадані інстанції[44].

  До основних норм з безпеки експлуатації будівель відносять п.10 ДБН В.2.2-9-2009:

- Про забезпечення безпечних підходів та під’їздів до будинків, можливість їх безпечного перебування та переміщення усередині будівлі.
- Дозволяється передбачати та встановлювати захисні пристрої для унеможливлення несанкціонованого проникнення до будівлі.
- Про висоту порогів та дозволений рівень перепадів рівня підлоги усередині приміщення.

- Про правильне встановлення різних видів дверей, їх позначення та обмеження на встановлення деяких видів скляних дверей.
- Про встановлення огорожі на покрівлі будівлі згідно ГОСТ 25772 та ДБН В.1.1-7.
- Про правильне встановлення систем водостоку.
- Про необхідність розробки заходів про перепрофілювання будівлі у разі зміни її функцій.

Також у цьому ж документі зазначено вимоги щодо енергозбереження:

- Оптимальність енерговитрат має бути встановлена згідно з вимогами ДСТУ Б А.2.2-8 і ДСТУ-Н Б А.2.2-5.
- Про необхідність збереження теплозахисних функцій огороджувальних систем згідно ДБН В.2.6-31, СНиП 2.04.05.
- Усі будівлі, що під'єднані до загальної інфраструктури мають бути обладнані автоматизованими приладами обліку спожитого ресурсу.
- Не допускається встановлення систем дефлекторів будь-якого типу.
- Про загальні норми проектування систем вентиляції та можливість обладнання таких приладами автоматизованого управління.
- Рекомендується у разі встановлення індивідуальних джерел енергопостачання використовувати відновлювані джерела (сонячну енергію, теплову енергію)[43].

Експлуатація будівлі може бути почата тільки після отримання акту про готовність об'єкту до експлуатації. Він має визначену форму і у ньому вказується наступна інформація про об'єкт:

- Місто, номер та дата складання акту.
- Найменування об'єкта, вид та категорія складності, згідно з проектною документацією.
- Адресу та код об'єкта (згідно з державним класифікатором).
- Інформація про підрядників (субпідрядників, якщо такі були), проектувальників та про проектну документацію.
- Інформація про дозвіл на будівництво, час початку та закінчення будівельно-монтажних робіт.

- Дані основних показників об'єкта (площа, продуктивність, потужність, виробнича площа, кількість робочих місць та інші показники).
- Вартість будівництва.
- Підписи замовника, генерального проектувальника, генерального підрядника, субпідрядника, страхової організації, представника профспілкової організації[45].

1.6. Висновки до розділу

У розділі було сформульовано та поставлено задачі, необхідні для виконання роботи. Проаналізовано більше 20 аналогів системи, що необхідно розробити включаючи як системи, які використовують Blockchain, так і ті, що не використовують. Проведено аналіз основних типів існуючих датчиків IoT та описано основні принципи роботи кожного з них. Досліджено та проаналізовано нормативні документи, які описують вимоги до використання та створення комерційних приміщень/будівель. Вивчено питання аналітики даних безпосередньо у IoT системах та методи аналізу таких даних.

РОЗДІЛ 2. РОЗРОБКА ПРОЕКТУ ТА МЕТОДІВ ОБРОБКИ ДАНИХ IOT РІШЕННЯ

2.1. Аналіз методів обробки даних з IoT датчиків

Існує багато методів аналізу даних для IoT, але деякі з найбільш поширених і корисних методів включають візуалізацію даних, статистичний аналіз, аналіз даних і машинне навчання.

1. Візуалізація даних.

Візуалізація даних Інтернету речей – це процес представлення даних, зібраних з пристроїв Інтернету речей, у візуальному форматі. Це можна зробити за допомогою різних методів, включаючи графіки та діаграми.

Візуалізація даних є потужним інструментом, який можна використовувати, щоб отримати уявлення про набори даних. Візуалізуючи дані, можна легко визначити закономірності та тенденції. Поширені методи візуалізації даних включають:

- Лінійні графіки.
- Гістограми.
- Стовпчасті діаграми.
- Секторні діаграми.

Візуалізація даних IoT може бути використана для покращення зручності використання даних IoT як для людей, так і для машин. Наприклад, візуалізації можуть полегшити людям розуміння складних наборів даних, а також їх можна використовувати для створення алгоритмів, які можуть автоматично виявляти закономірності в даних.

Існує ряд різних програмних інструментів, які можна використовувати для візуалізації даних IoT, включаючи інструменти з відкритим кодом і комерційні інструменти. Деякі поширені інструменти з відкритим кодом включають D3.js, Highcharts і Chart.js (рис. 2.1). Комерційні інструменти включають Tableau і Qlik.

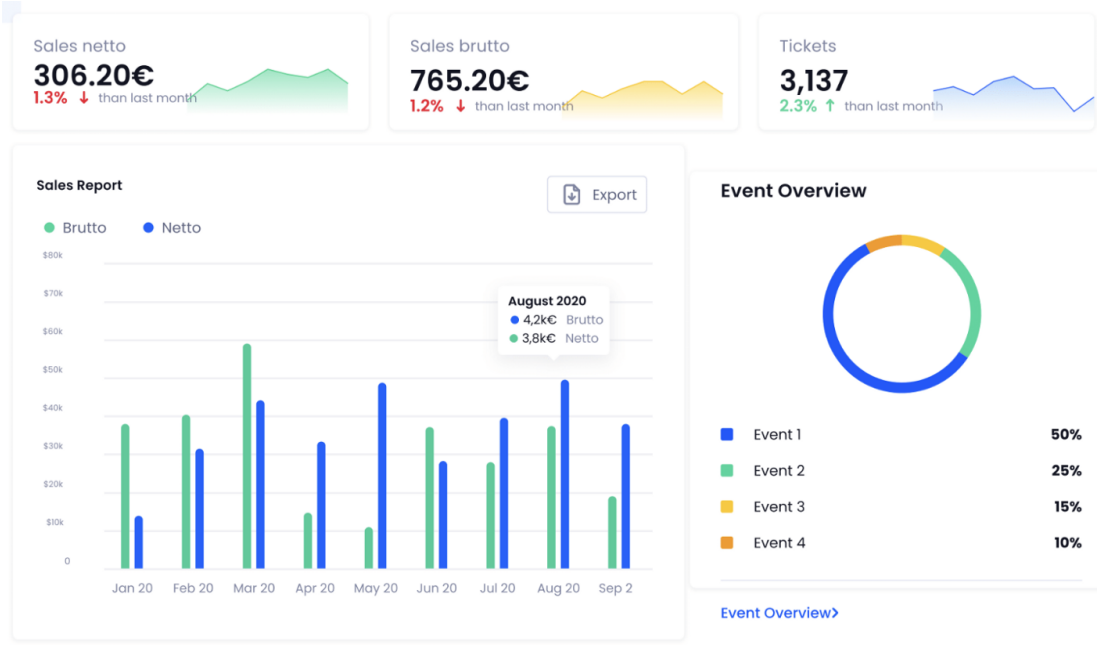


Рисунок 2.1 – Приклад побудови графіків різних типів за допомогою Chart.js[47]

Візуалізацію даних IoT можна використовувати для створення різноманітних типів візуалізацій, включаючи статичні візуалізації, інтерактивні візуалізації та візуалізації в реальному часі. Статичні візуалізації зазвичай створюються у вигляді зображень, які можна роздрукувати або поділитися в електронному вигляді. Інтерактивні візуалізації зазвичай створюються за допомогою веб-технологій, таких як HTML, CSS і JavaScript, і вони дозволяють користувачам взаємодіяти з даними різними способами[46].

2. Статистичний аналіз.

Статистичний аналіз є ще одним поширеним методом, який використовується для аналізу даних у додатках IoT. Статистичний аналіз можна використовувати для визначення тенденцій і закономірностей у наборах даних. Його також можна використовувати для прогнозування майбутніх даних.

Існує багато різних типів статистичного аналізу, і відповідний тип для використання залежить від характеру даних і питання, яке ставиться. Деякі поширені типи статистичного аналізу включають:

- Описова статистика. Цей тип статистичного аналізу узагальнює дані та використовується для опису характеристик набору даних.
- Статистика, що базується на висновках. Цей тип статистичного аналізу робить висновки або передбачення щодо сукупності на основі вибірки.

- Регресійний аналіз. Цей тип статистичного аналізу визначає зв'язки між різними змінними.

- Аналіз часових рядів. Цей тип статистичного аналізу використовується, щоб зрозуміти, як змінна змінюється з часом[48].

3. Інтелектуальний аналіз даних.

Інтелектуальний аналіз даних – це процес вилучення цінної інформації з великих наборів даних. Інтелектуальний аналіз даних можна використовувати для визначення закономірностей і тенденцій у наборах даних. Його також можна використовувати для прогнозування майбутніх даних. Поширені методи аналізу даних включають:

- Кластеризація – це метод аналізу даних, який групує точки даних, схожі один на одного. Це можна використовувати для виявлення прихованих закономірностей або групування даних.

- Класифікація – це метод аналізу даних, який використовується для прогнозування класу об’єкта. Це метод навчання під наглядом, що означає, що для нього потрібен навчальний набір даних, позначений правильним класом. Алгоритми класифікації вивчають ці навчальні дані, а потім їх можна використовувати для прогнозування класу нових даних.

- Нейронні мережі – це тип алгоритму машинного навчання, який можна використовувати для аналізу даних. Нейронні мережі можуть навчитися розпізнавати закономірності в даних і робити прогнози щодо майбутніх даних.

- Дерева рішень – це тип алгоритму машинного навчання, який можна використовувати для інтелекту даних. Дерева рішень можуть навчитися розпізнавати закономірності в даних і робити прогнози щодо майбутніх даних[49].

4. Машинне навчання.

Машинне навчання – це процес навчання комп’ютерів вчитися на даних. Машинне навчання можна використовувати для визначення закономірностей і тенденцій у наборах даних. Його також можна використовувати для прогнозування майбутніх даних.

Існує три типи алгоритмів машинного навчання:

- Навчання під керівництвом – алгоритм навчається на позначеному наборі даних, що означає, що бажаний результат уже відомий.
- Навчання без нагляду – алгоритм навчається на немаркованому наборі даних, що означає, що бажаний результат невідомий.
- Навчання з підкріпленням – алгоритм тренується за допомогою сигналу підкріплення, що означає, що він постійно налаштовується на основі його продуктивності.

Найпоширенішими алгоритмами машинного навчання є лінійна регресія, логістична регресія, дерева рішень та нейронні мережі[50].

2.2. Огляд протоколів IoT

Протоколи IoT загалом можна розділити на два типи:

- Протоколи передачі даних.
- Мережеві протоколи.

Протоколи передачі даних.

MQTT. MQTT – це легкий протокол обміну повідомленнями, який дозволяє пристроям спілкуватися один з одним через Інтернет. Він призначений для використання в обмежених середовищах, таких як пристрої з низьким енергоспоживанням або мережі з обмеженою пропускною здатністю.

MQTT використовує модель публікації/підписки, в якій клієнти (пристрої) можуть підписатися на теми та отримувати повідомлення, коли нові дані публікуються для цих тем. Це робить його ідеальним для використання в додатках IoT, де датчики можуть публікувати дані на центральний сервер, а інші пристрої можуть підписатися на ці дані.

При використанні MQTT кожен клієнт повинен підключитися до брокера MQTT. Брокер відповідає за маршрутизацію повідомлень між клієнтами та управління клієнтськими підписками. Доступна низка відкритих і комерційних брокерів MQTT.

Щоб використовувати MQTT, кожен клієнт спочатку підключається до брокера за допомогою TCP-з’єднання. Після підключення клієнт може підписатися на теми, публікувати повідомлення та отримувати повідомлення від брокера.

Повідомлення надсилаються як масиви байтів, тому вони можуть мати будь-який розмір або формат (включаючи двійкові дані).

Протокол MQTT має ряд функцій, завдяки яким він добре підходить для додатків IoT:

- Він легкий та ефективний, що робить його ідеальним для використання на обмежених пристроях з обмеженими ресурсами.
- Він використовує модель опублікування/підписки, що дозволяє легко розповсюджувати дані від датчиків кільком клієнтам без необхідності, щоб кожен датчик підтримував власне з'єднання з кожним клієнтом.
- Він підтримує рівні якості обслуговування (QoS) 0 і 1, що гарантує надійну доставку повідомлень навіть у мережах з низькою пропускною здатністю або високою затримкою.
- Він має вбудовані функції безпеки, такі як аутентифікація пароля та шифрування SSL/TLS

AMQP. AMQP – це протокол прикладного рівня, який використовує модель публікації/підписки. У цій моделі клієнти можуть публікувати повідомлення на центральному сервері, який називається брокером, і будь-яка кількість абонентів може отримувати ці повідомлення. Це дозволяє здійснювати асинхронний зв'язок між додатками або компонентами без необхідності безпосереднього підключення кожного компонента до будь-якого іншого компонента.

AMQP пропонує кілька переваг перед іншими протоколами черги повідомлень:

- Це відкритий стандарт з кількома реалізаціями, доступними від різних постачальників. Це дозволяє організаціям вибирати найкраще рішення для своїх потреб, не прив’язуючись до продукту одного постачальника.
- Він має вбудовані функції надійності, такі як підтвердження повідомлень і гарантії доставки, які забезпечують доставку повідомлень навіть у разі збоїв мережі або системи.
- Брокери AMQP можуть бути об’єднані разом, щоб забезпечити масштабованість і високу доступність.

- AMQP надає широкі можливості маршрутизації, які дозволяють маршрутизувати повідомлення на основі вмісту, тобто отримуватимуть його лише абоненти, яких цікавить певний тип повідомлень.
- AMQP підтримує кілька шаблонів обміну повідомленнями з коробки, включаючи напрямок, публікацію/підписку та запит/відповідь.

Через свою «тяжкість» цей протокол дуже рідко використовують при розробці IoT. За рахунок своєї складності AMQP негативно впливає на час роботи пристроїв та займає велику частину обчислювальних потужностей.

WebSockets. WebSockets – це протокол, який забезпечує двостороннє спілкування в реальному часі між клієнтом і сервером.

WebSockets використовує сокетове з'єднання для встановлення повнодуплексного каналу зв'язку, по якому повідомлення можуть передаватися між клієнтом і сервером. З’єднання через сокет ініціюється клієнтом, який надсилає запит на рукостискання серверу. Потім сервер відповідає власним повідомленням про рукостискання, після чого обидві сторони можуть почати надсилати дані один одному.

Підключення WebSocket є постійними, вони не закриваються після кожного повідомлення, як з'єднання HTTP. Це означає, що після встановлення з’єднання WebSocket обидві сторони можуть продовжувати надсилати повідомлення туди-сюди необмежений час без необхідності щоразу відновлювати з’єднання. Завдяки цьому WebSockets добре підходить для програм, які вимагають довготривалих з’єднань, таких як програми чату та багатокористувацькі ігри.

Його можна використовувати для керування передачі даних між кількома пристроями в мережі IoT. Він найчастіше використовується в місцях, які діють як клієнти або сервери.

Мережеві протоколи.

Wi-Fi. Wi-Fi – це технологія бездротової мережі, яка дозволяє пристроям підключатися до Інтернету без використання фізичних кабелів. Він використовує радіохвилі для передачі даних і, як правило, швидше, ніж інші бездротові технології, такі як Bluetooth.

Wi-Fi мабуть найрозповсюджена технологія бездротової мережі. Вона зазвичай використовується в пристроях IoT, оскільки пропонує ряд переваг перед іншими бездротовими технологіями. Wi-Fi має більш широкий діапазон, тому він може охоплювати більші площі, а також може передавати дані на більших швидкостях. Це робить його ідеальним для програм, де потрібно швидко передавати дані, наприклад, потокове відео або ігри.

Ще одна перевага Wi-Fi полягає в тому, що він споживає менше енергії, ніж інші бездротові технології, тому його можна використовувати в пристроях з батарейним живленням без значного скорочення терміну служби батареї. Це робить його ідеальним для використання в портативних пристроях IoT, таких як носимі пристрої або гаджети для розумного дому.

Нарешті, Wi-Fi широко підтримується, тому не існує проблем з його використанням на пристроях IoT. Це означає, що виробники можуть розробляти продукти, які працюють із існуючими мережами та інфраструктурою Wi-Fi, що спрощує розгортання та масштабування рішення IoT.

Bluetooth/BLE. Bluetooth – це стандарт технології бездротового зв'язку малої дальності для обміну даними на короткі відстані (менше 30 метрів). Bluetooth може підключати кілька пристроїв, дозволяючи їм працювати разом для виконання завдань або обміну даними.

Bluetooth Low Energy (BLE) — це версія Bluetooth, розроблена для низького споживання енергії. BLE використовується для програм, які вимагають лише періодичної передачі даних. BLE також можна використовувати для більш складних програм, які вимагають частої передачі даних, наприклад, потокового аудіо чи відео.

BLE є ідеальною технологією для додатків IoT, оскільки вона дозволяє пристроям спілкуватися один з одним, використовуючи дуже мало енергії. Пристрої BLE можуть роками працювати від батарейок типу «таблетка», що робить їх ідеальними для використання в пристроях Інтернету речей, що живляться від батарей. Крім того, пристрої BLE можна перевести в режим сну, коли вони не використовуються, що ще більше зменшує їх споживання енергії.

BLE вже використовується в широкому спектрі додатків IoT, включаючи автоматизацію розумного дому, промисловий моніторинг і контроль, відстеження активів та багато іншого.

ZigBee. ZigBee – це протокол зв'язку, розроблений для малопотужних цифрових радіоприймачів, які використовуються в різних програмах, включаючи бездротові сенсорні мережі, домашню автоматизацію та персональні мережі.

Однією з ключових переваг ZigBee є низьке енергоспоживання. Це робить його ідеальним для пристроїв, що живляться від батареї, таких як бездротові датчики, яким може знадобитися працювати протягом тривалого часу без підзарядки чи заміни. Ще однією перевагою ZigBee є його сітчаста мережа(mesh), яка дозволяє пристроям спілкуватися один з одним, навіть якщо вони не знаходяться в прямому діапазоні один від одного. Це означає, що один пристрій може діяти як шлюз між двома іншими, які інакше були б поза зоною дії.

ZigBee також широко використовується в Інтернеті речей (IoT). Zigbee добре підходить для додатків IoT, оскільки може підтримувати велику кількість пристроїв з низьким споживанням енергії, тривалим терміном служби батареї та можливостями сітчастої мережі.

LoRaWAN. LoRaWAN – це технологія глобальної мережі з низьким енергоспоживанням (LPWAN), розроблена для того, щоб дозволити пристроям IoT спілкуватися на великі відстані з низьким споживанням енергії. Технологія розроблена для пристроїв, що працюють від батареї, яким потрібно зв'язуватися нечасто та на великі відстані, наприклад, розумні лічильники, датчики навколишнього середовища та системи безпеки.

Однією з ключових переваг LoRaWAN є його здатність працювати як у сільському, так і в міському середовищі. Технологія використовує адаптивні швидкості передачі даних, які автоматично пристосовуються до мінливих умов, таких як перешкоди або сила сигналу, щоб підтримувати надійне з'єднання. Це робить його добре придатним для додатків, які повинні працювати в зонах зі змінними або складними радіочастотними умовами.

Крім того, LoRaWAN підтримує наскрізне шифрування з метою безпеки. Це допомагає гарантувати, що лише авторизовані пристрої мають доступ до даних, і допомагає захистити від підслуховування або втручання.

Наразі LoRaWAN використовується в різноманітних додатках IoT, включаючи інфраструктуру розумного міста, відстеження активів, віддалений моніторинг тощо[51].

2.3. Вибір технологій та протоколів

Для розробки системи було обрано наступні технології та протоколи:

Мова програмування PHP. PHP є широко використовуваною мовою сценаріїв загального призначення з відкритим кодом, яка особливо підходить для веб-розробки та може бути вбудована в HTML.

PHP-код можна просто змішати з HTML-кодом або використовувати в поєднанні з різними механізмами шаблонів і веб-фреймворками. PHP-код зазвичай обробляється інтерпретатором PHP, реалізованим як модуль на веб-сервері або як виконуваний файл.

Веб-сервер поєднує результати інтерпретованого та виконаного PHP-коду, який може бути будь-яким типом даних, включаючи зображення, зі згенерованою веб-сторінкою. PHP також розвинувся, включивши інтерфейс командного рядка і може використовуватися в окремих програмах.

Для створення веб-інтерфейсу буде використовуватися PHP 8.1 в якому було додано багато різноманітного функціоналу, що тільки покращив мову, залишаючи її у топі мов, що обираються для веб розробки.

Фреймворк Symfony. PHP Symfony — це фреймворк веб-додатків PHP для додатків MVC. Він надає розробникам архітектуру, компоненти та інструменти для швидшого створення складних веб-додатків.

Фреймворк Symfony має широкий спектр функцій, які роблять його придатним для створення великомасштабних програм:

- Гнучка система маршрутизації, яку можна використовувати для створення URL-адрес для різних сторінок і ресурсів у вашій програмі.

- Потужний механізм створення шаблонів, який дозволяє легко створювати та підтримувати постійний вигляд на всьому веб-сайті або в додатку.
- Підтримка інтернаціоналізації (I18N), що дозволяє легко створювати багатомовні веб-сайти та програми.
- Складна система безпеки, яку можна використовувати для захисту вашої програми від потенційних загроз, таких як XSS, CSRF або SQL Injection.

Apache. Apache – це веб-сервер, доступний для багатьох різних операційних систем. Це програмне забезпечення з відкритим кодом, тобто будь-хто може переглядати та змінювати вихідний код. Apache є одним із найпопулярніших веб-серверів у світі, і використовується багатьма великими організаціями, включаючи Facebook, IBM та уряд США.

Apache є дуже універсальним веб-сервером, який можна використовувати для різних цілей. Його можна використовувати для розміщення простих веб-сайтів або для створення великих і складних веб-додатків. Apache також можна використовувати як зворотний проксі-сервер, що означає, що він може спрямовувати трафік на інші сервери за певними критеріями.

Apache є дуже стабільним і надійним веб-сервером, який часто використовується в середовищах з високим трафіком. Він також дуже масштабований, що означає, що його можна легко розширити, щоб розмістити більше користувачів і більше трафіку.

HTTP-сервер Apache є найбільш поширеним програмним забезпеченням веб-сервера. Програмне забезпечення доступне безкоштовно на веб-сайті Apache. Програмне забезпечення випускається під ліцензією Apache, безкоштовною ліцензією на програмне забезпечення з відкритим вихідним кодом.

MQTT. MQTT було обрано як брокер повідомлень серед конкурентів через його легкість, швидкість та зручність використання.

Як вже було зазначено, MQTT – це легкий протокол обміну повідомленнями для публікації та підписки, розроблений для обмежених пристроїв і мереж з низькою пропускною здатністю та високою затримкою. MQTT часто використовується в поєднанні з HTTP, щоб забезпечити більш ефективний протокол зв'язку для обмежених пристроїв.

Останніми роками MQTT стає дедалі популярнішим як протокол зв’язку IoT через низькі витрати та просту модель обміну повідомленнями.

Протокол MQTT складається з трьох основних компонентів:

- Брокер, який відповідає за отримання та пересилання повідомлень.
- Клієнт, який підключається до брокера та публікує або підписується на теми.
- Повідомлення, яке є даними, які публікуються або на які підписані.

JavaScript. JavaScript — це мова програмування, яка дозволяє створювати динамічно оновлюваний контент, керувати мультимедіа, анімувати зображення та багато іншого.

JavaScript робить веб-сторінки інтерактивними. Він працює на комп’ютері відвідувача і не вимагає постійного перезавантаження сторінок із сервера.

JavaScript можна використовувати для перевірки форм перед відправкою, створення файлів cookie, виявлення користувацьких агентів та багато іншого.

У цьому випадку JavaScript було обрано для взаємодії та керуванням графіками та діаграмами, що будуть створені за допомогою ChartJS.

2.4. Архітектура (модель) аналізу даних комерційних будівель

На основі проаналізованих даних та отриманих знань було створено архітектуру системи зображену у Додатку А, що дозволяє зрозуміти суть усіх учасників системи та передбачити застосування обраних технологій.

На діаграмі системи можна побачити наступні компоненти:

- Користувач (User).
- Веб сервер (Apache).
- Базу даних (MySQL).
- Пул нод блокчейну (Pool of the Blockchain Nodes).
- Ноду блокчейну (Blockchain Node).
- Прошарок, або «обкладинка» для блокчейну (Wrapper).
- Хаби – система, що агрегує дані з датчиків (Hub).
- Датчики – невеличка програма, що імітує роботу датчиком за допомогою відправки довільних значень на агрегатор (Sensor).

Датчики. Датчики на діаграмі зображено у вигляді паралелепіпеду. Датчик на діаграмі мається на увазі під будь-яким датчиком: температура, рух, тиск, тощо.

Датчики надсилають інформацію до Хабу(Hub), за допомогою різних мережевих протоколів (Wi-Fi, BLE, ZigBee тощо). Усі датчики перед використанням їх у системі мають бути пов’язані з Хабом (авторизовані) таким чином, щоб знати куди необхідно надсилати інформацію. Датчики у діаграмі виступають лише як засіб для збору інформації з приміщення/будівлі.

Хаби. Хаби у діаграмі мають роль агрегатора даних що надходять від датчиків. Хаб збирає інформацію і, як пристрій вже більш високого рівня надсилає її до Пулу нод блокчейну. Хаб фактично це мікрокомп’ютер, який реалізує додатковий рівень фільтрації та безпеки перед передачею даних безпосередньо до блокчейну. Хаб є учасником MQTT обміну, що дає йому можливість надсилати нові дані від датчиків далі до пулу нод блокчейну.

Пул нод блокчейну. Пул нод блокчейну отримує дані від Хабів за допомогою MQTT та додає дані у наступний блок. Фактично дані отримує не весь пул нод, а лише одна нода, до якої прислав дані Хаб (найближча нода). У свою черга нода додає отримані дані у блок, який потім майнить та додає у блокчейн. Завдяки невиликій складності обчислень майнинг не займає багато часу і проходить доволі швидко, а дані які потрапили до блоку потім вже не можливо буде змінити. Ноди спілкуються поміж собою за допомогою TCP P2P(peer2peer) зв’язку. Таким чином ноди можуть дізнатися про новий добутий блок та включити його до свого блокчейну.

Варто додати, що оскільки P2P сповіщає тільки ноди новина про створений блок просто так не зможе дістатися далі до Веб-серверу за допомогою цього ж типу зв’язку. Для того, щоб сервер дізнався про нові дані задіяно MQTT, який у свою чергу надасть Веб серверу інформацію про те що новий блок було знайдено і необхідно забрати з нього дані для подальшої обробки.

Веб сервер. Веб сервер це HTTP сервер Apache, що буде відповідати за обробку та надання даних. Веб сервер отримує дані від Пулу нод (Ноди, що змайнила блок) за допомогою MQTT та оброблює його, зберігає інформацію у базу даних, звіряється з списком пристроїв, що було додано користувачем.

Веб сервер у даному випадку виступає більше як прошарок між користувачем та сховищем даних. Він виконує обробку інформації та надає її користувачу у вже скорегованому вигляді з усіма допоміжними даними. Веб сервер спілкується з користувачем як за допомогою і стандартних HTTP запитів, так і за допомогою WebSockets, для того, щоб мати змогу надавати дані у режимі реального часу.

Веб сервер виступає і в ролі автентифікатора для захисту від несанкціонованого доступу к даним.

Користувач. Користувач, тобто клієнт у свою чергу має пристрій з доступом до мережі інтернет і логін/пароль для доступу до панелі. Для зручності користувача використовується метод візуалізації даних, що дозволяє легко ознайомитися з даними.

2.5. Висновки до розділу

У розділі проаналізовано методи обробки та аналізу даних, які отримані з датчиків IoT, у які входять візуалізація, статистичний аналіз, інтелектуальний аналіз та машинне навчання . Проведено огляд та аналіз протоколів IoT, а саме MQTT, AMQP, WebSockets, Wi-Fi, Bluetooth/BLE, ZigBee, LoRaWAN. Спроектовано систему збереження та обробки даних від датчиків IoT за допомогою технології Blockchain (Додаток А). Було проведено вибір технологій для реалізації та впровадження спроектованої системи збереження та аналізу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ АРХІТЕКТУРИ ІОТ РІШЕННЯ

3.1. Розробка імітаційної моделі генерації даних

При реалізації імітаційної моделі генерації даних (далі – «датчик») її було спроектовано як мікросервіс. Датчик побудовано за допомогою мови програмування PHP на основі фреймворку Symfony без додавання бандлів(пакетів) для веб-інтерфейсу, що зробило його особливо «легким» та малозатратним у розрізі використання потужностей пристрою.

Початковою точкою є встановлення самого PHP та всіх необхідних розширень для нього. Після встановлення PHP необхідно встановити Composer.

Composer – це інструмент для управління залежностями в PHP. Це дозволяє вам оголошувати бібліотеки, від яких залежить ваш проект, і керуватиме (встановлювати/оновлювати) ними за вас.

Composer не є менеджером пакетів у тому ж сенсі, як Yum або Apt. Однак він моделює себе за цими інструментами і забезпечує зручний спосіб встановлення та оновлення бібліотек, які потрібні вашій програмі.

Composer відрізняється від цих інструментів у двох важливих аспектах. По-перше, він не встановлює нічого глобально. Усі бібліотеки, якими він керує, встановлені в каталозі, який називається vendor, у корені вашого проекту. Це значно полегшує керування залежностями для різних проектів, оскільки вам не доведеться турбуватися про конфлікт залежностей одного проекту з залежностями іншого.

По-друге, і, можливо, важливіше, Composer використовує інший підхід до керування версіями, ніж ці інші менеджери пакетів. Замість того, щоб використовувати номери версій, як-от 1.2.3, він використовує схему керування версіями, яка називається Semantic Versioning, або скорочено SemVer.

У SemVer версії представлені номером із трьох частин, що складається з основної версії, другорядної версії та версії виправлення. Основна версія збільшується, коли вноситься критична зміна, друга версія збільшується, коли додається нова функція, а версія виправлення збільшується, коли виправляється помилка.

Це може здатися незначною відмінністю, але вона має значний вплив на те, як бібліотеки версійні та як Composer вирішує залежності.

Загалом, Composer намагається розв'язувати залежності максимально консервативним способом. Тобто він вибере найнижчий можливий номер версії, який задовольняє залежності вашого проекту.

Composer доступний для Windows, Mac і Linux. Його можна завантажити з веб-сайту Composer.

За допомогою Composer встановлюємо Symfony (рис. 3.1).

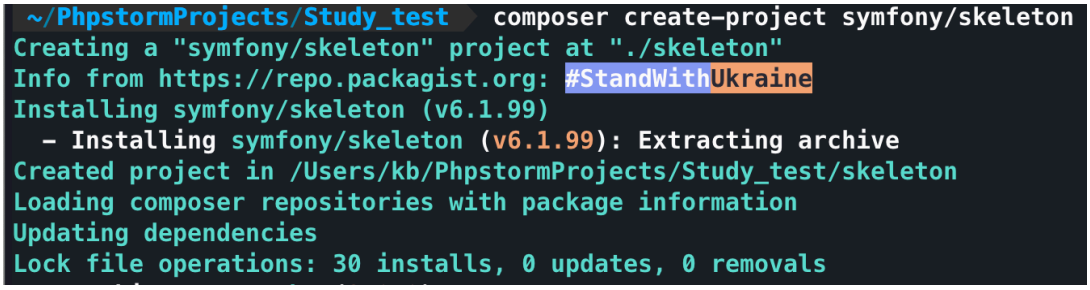


Рисунок 3.1 – Процес встановлення Symfony за допомогою термінала та Composer

Для редагування коду на мові PHP використано IDE PhpStorm від компанії JetBrains.

PhpStorm – це кросплатформна IDE для PHP, розроблена JetBrains. Він написаний на Java і забезпечує інтегроване середовище розробки (IDE) для розробників PHP. IDE має багатий набір функцій, включаючи завершення коду, рефакторинг, дебагінг та модульне тестування. PhpStorm доступний як окрема програма або як плагін для IntelliJ IDEA.

PhpStorm надає повний набір інструментів для розробників PHP. IDE включає в себе редактор коду, дебагер, профайлер, інструмент модульного тестування та систему контролю версій(Git). PhpStorm також підтримує широкий спектр плагінів, які можна використовувати для розширення функціональних можливостей IDE.

Редактор коду в PhpStorm заснований на редакторі IntelliJ IDEA і підтримує ряд інших мов, таких як HTML, CSS і JavaScript.

Дебагер у PhpStorm заснований на розширенні Xdebug. Він забезпечує графічний інтерфейс для налагодження програм PHP. Дебагер можна використовувати для пошагового перегляду коду, перевірки змінних і встановлення точок зупинки.

Профайлер у PhpStorm заснований на розширенні XHProf. Він забезпечує графічний інтерфейс для профілювання PHP-додатків. Профайлер можна використовувати для виявлення вузьких місць і оптимізації коду.

Інструмент модульного тестування в PhpStorm заснований на платформі PHPUnit. Він забезпечує графічний інтерфейс для запуску та налагодження модульних тестів. Інструмент модульного тестування можна використовувати для створення, виконання та налагодження модульних тестів.

Система контролю версій у PhpStorm базується на системі контролю версій Git. Він надає графічний інтерфейс для керування репозиторіями Git. Систему контролю версій можна використовувати для керування змінами коду та відстеження історії проекту.

PhpStorm – це потужна IDE для розробки PHP. Вона надає повний набір інструментів для розробки PHP-додатків, через що його і було обрано для роботи з PHP.

Після виконання команди маємо наступний зміст директорії у редакторі PhpStorm (рис. 3.2).

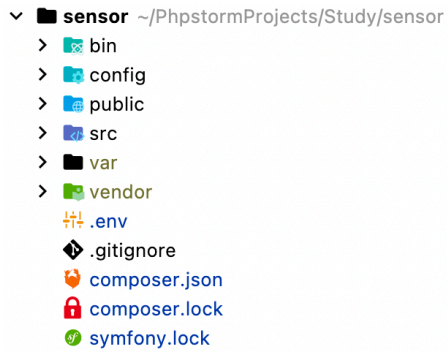


Рисунок 3.2 – Зміст директорії проекту у PhpStorm після встановлення Symfony

Наступним кроком необхідно створити клас, що буде відповідати за консольну команду, що генерує та надсилає дані до хабу.

Для імітації передачі даних по протоколам, зазначеним на додатку А, використано звичайний HTTP запит типу POST з вказанням кінцевої адреси хабу.

Тіло запита представляє собою звичайні дані у форматі «raw»(рис. 3.3).

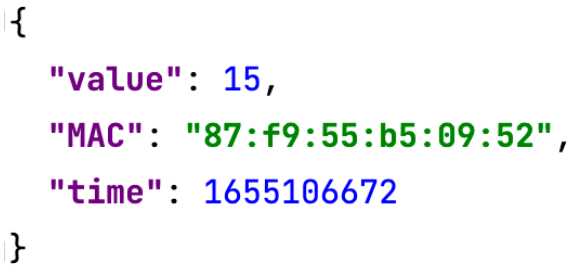


Рисунок 3.3 – Приклад сформованого запиту

Саме ж виконання програми є нескінченним задля постійного безперебійного генерування та відправки даних на хаб.

3.2. Реалізація системи агрегації та проксування згенерованих даних

Наступним елементом у схемі після датчиків є системи агрегації та проксування згенерованих даних (далі – «хаб»), яка агрегує дані та за допомогою MQTT відправляє їх далі до блокчейну.

Для реалізації хабу обрано Node.js через його швидке впровадження, низьке навантаження на систему та влаштовану взаємодію з MQTT через лише одну маленьку бібліотеку.

Node.js – це середовище виконання JavaScript, побудоване на движку JavaScript V8 Chrome. Екосистема пакету Node.js, npm, є найбільшою екосистемою бібліотек з відкритим кодом у світі.

Node.js зазвичай використовується для створення веб-додатків та API. Node.js є популярним вибором для мікросервісів. Він також використовується в Інтернеті речей, робототехніці та для створення настільних додатків.

Node.js має унікальну систему модулів. Кожен файл Node.js є модулем. Модулі, як правило, є бібліотеками, які можна включати в інші програми. Node.js має вбудовану систему модулів, а менеджер пакетів npm дозволяє легко встановлювати модулі та керувати ними.

Програми Node.js зазвичай пишуться на JavaScript. Однак Node.js також підтримує багато інших мов програмування через мовні прив’язки.

Node.js є однопоточним. Однак Node.js використовує керовану подіями, неблокуючу модель вводу-виводу, що робить його легким та ефективним. Програми Node.js можуть обробляти багато одночасних підключень з дуже невеликими витратами.

Node.js є відкритим вихідним кодом і має велику та активну спільноту. Існує багато модулів і бібліотек, доступних для Node.js.

Для створення проекту на Node.js спочатку необхідно впевнитися, що на пристрої присутні node та npm.

Після цього можна створити проект за допомогою команди «npm init». Для роботи з MQTT необхідно додати спеціальний пакет командою «npm install mqtt», він забезпечить доступ до функцій для роботи з MQTT(рис 3.4).

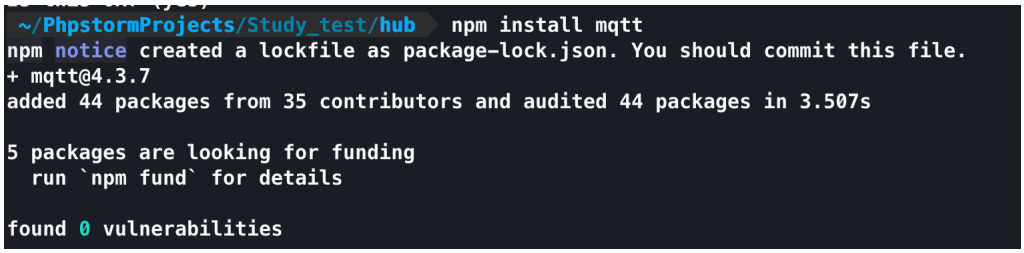


Рисунок 3.4 – Встановлення пакету для роботи з MQTT та Node.js

Задля створення системи у якій можна буде передавати дані необхідний брокер, який ці дані буде модерувати між клієнтами. Для цього використано сервіс від HiveMQ, що надає можливість створити власний хмарний брокер MQTT безкоштовно.

HiveMQ – це брокер MQTT повідомлень, оптимізований для Інтернету речей. Це ідеальний інструмент для розробників додатків IoT. Він дозволяє створити масштабоване, надійне та безпечне рішення. Брокер можна запускати локально або в хмарі, і він доступний у безкоштовній та комерційній версії.

Брокер має можливість масштабування і може обробляти мільйони повідомлень в секунду. Система HiveMQ дозволяє легко інтегрувати власні плагіни. Брокер також поставляється з веб-панеллю(рис.), яка полегшує керування та моніторинг брокера.

Брокер HiveMQ MQTT доступний у безкоштовній версії для спільноти та платний для комерційного корпоративного використання. Community edition безкоштовне для особистого використання та використання для розвитку. Саме такий план обрано для використання.

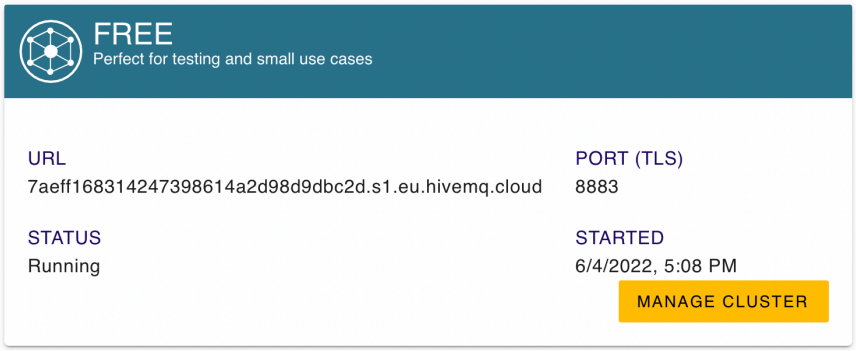


Рисунок 3.5 – Веб-панель HiveMQ

Node.js виступає у ролі серверу, який може постійно реєструвати та відповідати на звернення датчиків. Після отримання запиту від датчику він перевіряє зміст корисного навантаження та на основі цього результату блокує запит чи передає дані далі до MQTT брокера.

Для того, щоб будь-хто не мав можливості скористатися брокером реалізовано систему авторизації та автентифікації у веб застосунку.

Для того, щоб пристрій можна було використовувати його необхідно зареєструвати. Така обережність зберігається для забезпечення захисту даних та унеможливлення використання брокера зловмисником.

Створено користувача для хабу(рис. 3.6).

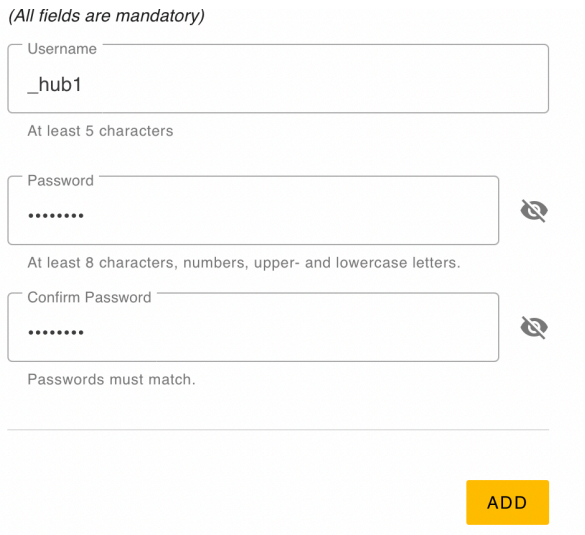


Рисунок 3.6 – Створення користувача для хабу

Оскільки MQTT використовує структуру типу Брокер-Слухач-Публіцист необхідно виділити окремий канал для спілкування хабу з обкладинкою блокчейну. Створено канал який називається «hub-node», по ньому буде передаватися лише інформація від хабу до обкладинки. Загальна модель передачі даних за допомогою MQTT представлена на рисунку 3.7.

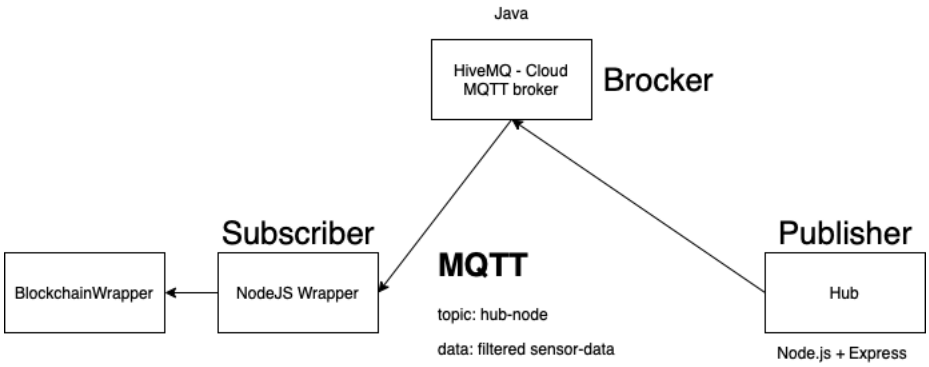


Рисунок 3.7 – Структурна модель інтеграції MQTT у систему

Задля тестування системи можна підписатися на канал «hub-node» щоб переглядати повідомлення у реальному часі (рис. 3.8). Для цього необхідно встановити mqtt-cli застосунок на свій пристрій. Після цього можна командою «mqtt sub» підписатися на канал використовуючи згенеровані логін та пароль. Так само за допомогою терміналу можна публікувати повідомлення в канал за допомогою команди mqtt pub. Загально за інструменту mqqt з командного рядку можна тестувати систему, прослуховувати канали, публікувати повідомлення напряму, без необхідності реалізувати це програмно, необхідно тільки мати сам застосунок, знати синтаксис команди та логін/пароль для автентифікації на каналі.

Рисунок 3.8 – Прослуховування каналу MQTT за допомогою терміналу

Таким чином реалізовано термінал, який агрегує та відправляє дані обкладинці за допомогою хмарного MQTT брокера HiveMQ.

3.3. Реалізація обкладинки для блокчейну

«Обкладинка» блокчейну фактично представляє собою типовий прошарок та призначена для забезпечення додаткового рівня обробки у разі його потреби. Також у реальній системі такий прошарок може займатися чимось схожим на «маршрутизацію» запитів до блокчейну. Таким чином прошарок може забезпечити раціональне розподілення даних між нодами.

Сам прошарок реалізовано за допомогою Node.js, знову ж таки, завдяки його перевагам, що вже були описані.

Для функціонування цього модуля необхідно мати встановлений Node.js та npm у системі. За допомогою команди «npm init» у новій директорії створено проект. Для зручності розробника після запуску команди надається інтерактивна система питання/відповідь(рис. 3.9) для заповнення усієї інформації про проект включаючи назву, опис, репозиторій Git, автора, ключові слова, тип ліцензії на використання, команду для початку тестування та вхідну точку програми.

```
~/PhpstormProjects/Study_/js-blockchain-wrapper  npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (js-blockchain-wrapper)
version: (1.0.0)
description: This is a blockchain wrapper based on typical Node.js application structure.
entry point: (index.js)
test command: npm test
git repository: https://github.com/test_user/test_repo
keywords: Blockchain, Node, Wrapper, MQTT
author: Kostiantyn Butenko
license: (ISC)
```

Рисунок 3.9 – Інтерактивна система питання/відповідь при запуску «npm init»

Варто зазначити, що оскільки прошарок слухає тільки канал «hub-node», у який надсилає інформацію вже зазначений хаб, будь-яка інша інформація його оминає на рівні брокера завдяки чому знімається зайве навантаження і wrapper не займається обробкою та валідацією усієї інформації. Після отримання та валідації даних, отриманих з хабу за допомогою MQTT по каналу «hub-node» прошарок надсилає вже профільтровану інформацію у форматі JSON до ноди блокчейну, яку вважає найбільш раціональним кандидатом для отримання та обробки даних, за допомогою POST запит з хедером «Content-type: application/json», що говорить про те що у корисному навантаженні містяться дані саме у JSON форматі.

3.4. Реалізація блокчейну для обробки та збереження даних

Блокчейн є одним з основних модулів даної системи і забезпечує обробку та збереження даних, їх цілісність та безпеку.

Для реалізації блокчейну було використано мову програмування PHP та Composer для можливості керування додатковими пакетами.

Сама система блокчейн є децентралізованою та складається з великої кількості учасників системи (нод), що унеможливлює підміну даних зі сторони одного з учасників мережі. Це означає що необхідно організувати «спілкування» між нодами для можливості ділитися даними. У блокчейн також має бути закладено функціонал майнингу, так як було обрано Proof-of-Work, що означає, що для того, щоб додати інформацію у блок спочатку треба вирішити деяку складну математичну задачу.

Proof-of-work (PoW) – це система, яка використовується для перевірки транзакцій і запобігання подвійних витрат на блокчейні. PoW — це математичний процес, який використовується для перевірки того, що транзакція дійсна, і для її завершення потрібно докласти певну кількість роботи або зусиль. Потрібна робота може бути будь-яка: від вирішення складної математичної задачі до перевірки цифрового підпису. Після завершення роботи транзакція записується в блокчейн і не може бути змінена або видалена.

Система PoW використовується для захисту блокчейна від атак, а також для створення нових блоків. Коли створюється новий блок, він додається в блокчейн, і робота, яка була потрібна для створення блоку, перевіряється мережею. Якщо робота недійсна, блок не додається в блокчейн.

Система PoW також використовується для стимулювання людей до участі в мережі. Коли блок створюється, особа, яка створила блок, винагороджується певною кількістю криптовалюти. Ця система стимулів заохочує людей брати участь у мережі та захищати мережу.

Система PoW не позбавлена недоліків, і її критикують за енергоємність. Процес майнінгу криптовалюти може бути дуже енергоємним.

Отож необхідно створити функціонал, що буде дозволяти майнити блоки та включати інформацію у них, підписувати.

Необхідно почати зі створення каналу, по якому ноди будуть «спілкуватися» між собою. При проектуванні було зазначено, що це P2P зв’язок. Отож вирішено використати ReactPHP.

ReactPHP – це бібліотека для PHP, яка дозволяє писати асинхронний код. Асинхронний код - це код, який може виконуватися паралельно з іншим кодом. Це може бути корисно для завдань, виконання яких займає багато часу, наприклад, мережеві запити або запити до бази даних.

ReactPHP використовує модель програмування, керовану подіями. Це означає, що ваш код організовано навколо подій. Коли відбувається подія, ваш код може реагувати на неї.

ReactPHP має ряд функцій, які дозволяють легко писати асинхронний код. Наприклад, він має вбудований HTTP-сервер, який можна використовувати для створення веб-додатків. Він також має абстракцію потоку, яка дозволяє працювати з потоками даних, такими як файли або мережеві сокети.

ReactPHP швидкий і легкий. Він також сумісний з низкою фреймворків PHP, такими як Laravel і Symfony.

Отож оскільки ReactPHP використовує модель програмування, керовану подіями для повідомлень, що надходять написано обробник(рис. 3.10). Обробник працює таким чином, що в залежності від типу повідомлення проводить з його змістом маніпуляції необхідні для функціонування блокчейну як децентралізованої системи.

```
$connection->on('data', function (string $data) use ($connection): void {
    $message = unserialize($data, [Message::class]);
    switch ($message->type()) {
        case Message::REQUEST_LATEST:
            $connection->write(serialize(new Message(
                type: Message::BLOCKCHAIN,
                serialize($this->node->blockchain()->withLastBlockOnly())
            )));

            break;
        case Message::REQUEST_ALL:
            $connection->write(serialize(new Message(
                type: Message::BLOCKCHAIN,
                serialize($this->node->blockchain())
            )));

            break;
        case Message::BLOCKCHAIN:
            $this->handleBlockchain(unserialize($message->data(), [Blockchain

            break;
    }
});
```

Рисунок 3.10 – Обробник для вхідних повідомлень P2P

Були також додані супровідні методи, що проводять обробку та необхідні для роботи з ReactPHP. За допомогою консольного застосунку можна перевірити як працює з'єднання та що за інформація передається (рис. 3.11).

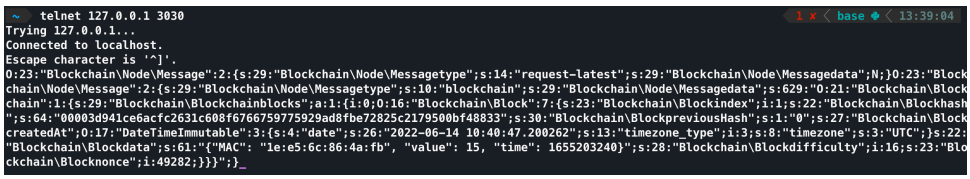


Рисунок 3.11 – Інформація, якою обмінюються ноди за допомогою TCP з'єднання

Для того, щоби взаємодіяти з блокчейном необхідно створити базовий API.

Перший ендпоінт необхідно створити для безпосередньо передачі даних для майнингу. Для цього будемо використовувати звичайний raw запит типу POST за посиланням «/mine»(рис. 3.12).

```php
case 'POST:mine':
    return new JsonResponse($this->node->mineBlock($request->getBody()->getContents()));
```

Рисунок 3.12 – Вхідна точка ендпоінту для майнингу

Сам процес майнингу було спрощено до необхідно мінімального. Встановлено стабільну складність для майнингу 4 нулі. натомість блокчейн у його звичайному представленні має динамічну складність, що дозволяє регулювати швидкість майнингу та генерувати різну нагороду за видобутий блок.

Алгоритмом для кешування обрано SHA-256. SHA-256 – це криптографічна хеш-функція. SHA-256 є членом сімейства криптографічних хеш-функцій SHA-2. SHA-256 – це 256-бітове (32-байтове) хеш-значення, яке зазвичай виражається у вигляді 64-значного шістнадцяткового числа.

Алгоритм SHA-256 використовується багатьма веб-сайтами та програмними додатками для створення цифрових підписів і перевірки цілісності даних. Наприклад, коли ви завантажуєте файл із веб-сайту, веб-сайт може надати цифровий підпис, який можна використовувати для підтвердження того, що файл не був змінений під час передачі.

SHA-256 є односторонньою функцією, це значить, що неможливо повернути функцію назад і створити вихідне повідомлення з хеш-значення. Ось чому цифрові підписи на основі SHA-256 вважаються безпечними.

SHA-256 часто використовується багатьма організаціями як заміна більш «дорослого» SHA-1.

Наступним етапом розроблено точку запиту типу GET для отримання усіх блоків «/blocks»(рис. 3.13). За допомогою неї розробник отримує усі блоки у вигляді JSON. У випадку рішення не шифрувати дані у блокчейну можна просто закрити цей ендпоінт для зовнішнього світу і дозволити запити тільки з блокчейну.

```
case 'GET:blocks':
    return new JsonResponse($this→node→blocks());
```

Рисунок 3.13 – Вхідна точка ендпоінту для отримання списку блоків

Аналогічно «/blocks» створено ендпоінт «/peers» з запитом типу GET для отримання списку усіх учасників мережі блокчейн.

Необхідно також ділитися даними і з самою панеллю аналізу.

Для цього використано бібліотеку RabbitMQ. RabbitMQ — це проміжне програмне забезпечення, яке реалізує розширений протокол черги повідомлень (AMQP). Він написаний мовою програмування Erlang і побудований на платформі Open Telecom Platform. RabbitMQ використовується в широкому спектрі додатків, включаючи банківську справу, електронну комерцію, соціальні мережі та телефонію.

RabbitMQ – брокер повідомлень з відкритим вихідним кодом, який пропонує надійний, масштабований і портативний спосіб маршрутизації та розповсюдження повідомлень. Він підтримує широкий спектр протоколів обміну повідомленнями і може бути розгорнутий у різноманітних середовищах. Він також пропонує ряд інструментів керування для моніторингу та керування вашими чергами повідомлень. Це ідеальне рішення для програм, яким потрібно обробляти велику кількість повідомлень надійним і масштабованим способом.

Для використання RabbitMQ його потрібно встановити. Оскільки він підтримуються майже на всіх платформах це не створює якоїсь проблеми, його можна легко встановити використовуючи документацію на сайті https://www.rabbitmq.com/download.html.

Було встановлено RabbitMQ для PHP за допомогою Homebrew на macOS командою «brew install rabbitmq». Сервер RabbitMQ можна запустити за допомогою команди «brew services start rabbitmq», після цього можна користуватися сервером.

У RabbitMQ є влаштована панель керування(рис. 3.14), її можна відкрити за посиланням localhost:15672/. У панелі можна побачити усю докладну інформацію про стан серверу черг та процеси.

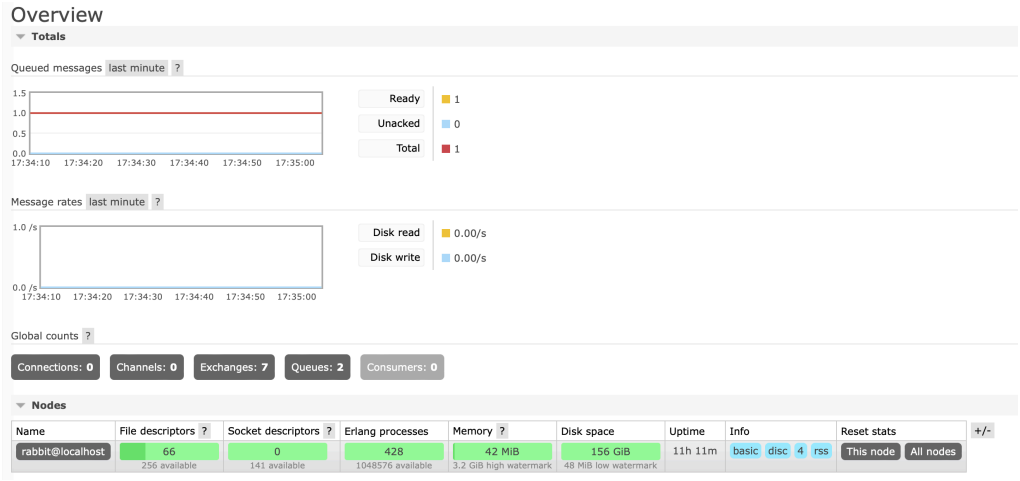


Рисунок 3.14 – Панель керування сервером RabbitMQ

RabbitMQ має дуже багато схожостей з MQTT, яку вже розглянуто. Сервер це брокер, що маніпулює усіма даними. Також є клієнт, що публікує повідомлення та слухач. Уся інформація передається по певним канал у зашифрованому вигляді, тобто у разі перехоплення пакету інформація розкрита не буде. У ролі клієнта, що публікує повідомлення буде виступати блокчейн, а у ролі слухача сервер адміністративної панелі.

Реалізація відправки повідомлень зображена на рисунку 3.15.

```
$connection = new AMQPStreamConnection( host: '127.0.0.1', port: 5672, user: self::RABBITMQ_USER,
$channel = $connection->channel();

$channel->queue_declare( queue: self::RABBITMQ_QUEUE, passive: false, durable: false, exclusive: fa

$msg = new AMQPMessage($data);
$channel->basic_publish($msg, exchange: '', routing_key: self::RABBITMQ_QUEUE);

$channel->close();
$connection->close();
```

Рисунок 3.15 – Відправка повідомлень за допомогою RabbitMQ

Для використання RabbitMQ у проекті необхідно також встановити пакет за допомогою Composer командою «composer require php-amqplib/php-amqplib», що налаштує всі пакети та дозволить одразу використовувати RabbitMQ у блокчейні.

3.5. Реалізація адміністративної панелі з візуалізацією даних та з підключенням бази даних

Для обробки даних будь-якого роду необхідно мати потужний і зручний інструмент для їх збереження і маніпуляції. Таким чином вирішено використовувати базу даних MySQL InnoDB.

InnoDB – це високопродуктивний механізм реляційної бази даних, який є компонентом популярної системи керування базами даних MySQL з відкритим вихідним кодом. InnoDB відомий своєю продуктивністю, масштабованістю та функціями відновлення після збоїв.

InnoDB реалізує механізм транзакцій бази даних, що означає, що він підтримує властивості ACID (атомність, узгодженість, ізоляція, довговічність). InnoDB також є безпечною базою даних, що означає, що вона може відновлюватися після збою системи або програми без втрати даних.

Встановити MySQL дуже просто і це можна зробити безкоштовно. Для скачування на різні платформи необхідно зайти на офіційний сайт MySQL, де можна скачати Community edition сервер та встановити його на свій пристрій. Після цього можна буде його запустити та використовувати для розробки системи.

Проведено проектування фізичної моделі бази даних(рис. 3.16).

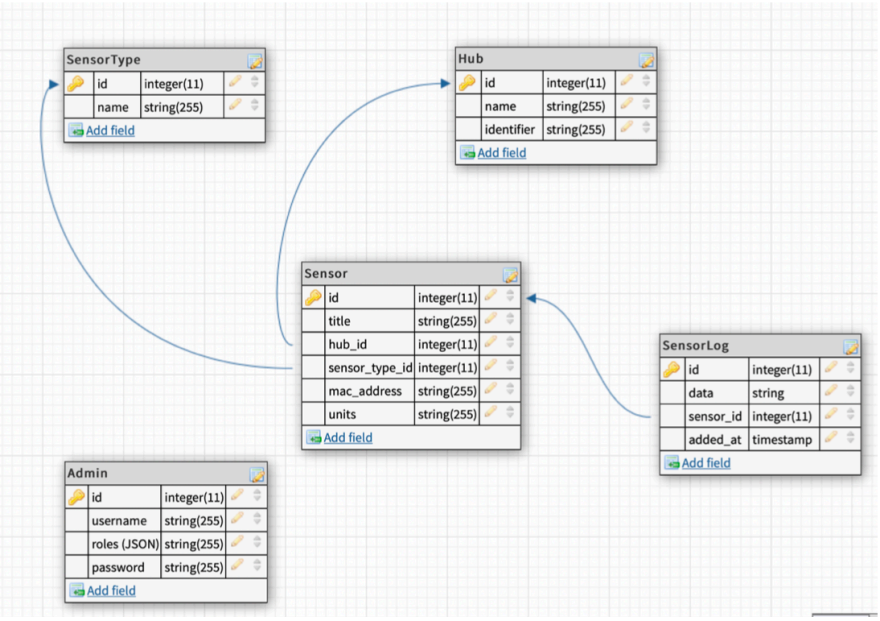


Рисунок 3.16 – Фізична модель бази даних

Створено базу даних та обрано її для наступної роботи за допомогою запиту, написаного у редакторі SQL коду(рис. 3.17).

```
1 CREATE DATABASE `diploma_db_`;
2 USE `diploma_db_`
```

Рисунок 3.17 – SQL запит на створення та використання бази даних

Після створення бази даних необхідно створити проект, до якої її можна буде підключити. Для розробки проекту обрано мову програмування PHP та фреймворк Symfony у його повній версії з модулями для відображення веб-контенту за допомогою шаблонізатора Twig.

Twig – це движок шаблонів для мови програмування PHP. Він використовується для відокремлення логіки відображення/презентації від логіки програми.

Twig використовує файл шаблону, який містить змінні, які замінюються вмістом з бази даних, а також директиви, які є інструкціями для Twig щодо того, як відтворити шаблон.

Twig компілює шаблони в необроблений код PHP. Це означає, що немає накладних витрат на завантаження або візуалізацію шаблону. Його синтаксис дуже схожий на HTML, завдяки чому дизайнерам легко розпочати роботу з Twig.

Twig безпечний. Він за своєю конструкцією не дозволяє будь-який PHP-код у шаблонах. Це робить неможливим для зловмисників введення шкідливого коду на сторінку. Twig компілює шаблони до сирого PHP-коду, що означає, що немає накладних витрат на завантаження або візуалізацію шаблонів. І врешті у Twig є повна інтеграція з Symfony, що означає, що для його встановлення не треба витрачати час.

Для встановлення повноцінної версії Symfony використано Composer. Для розширення його можливостей та набуття статусу веб-додатку необхідно ввести додаткову команду в Composer(рис. 3.18), на цьому встановлення Symfony буде завершено.

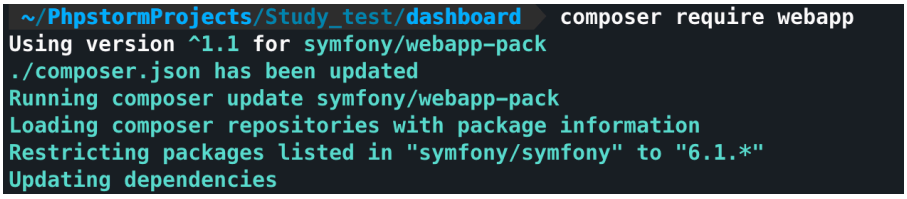


Рисунок 3.18 – Встановлення пакету веб-застосунка для Symfony за допомогою Composer

Після встановлення всіх необхідних компонентів треба налаштувати Symfony для коректної роботи з базою даних. Для цього потрібно відредагувати .env файл(рис 3.19), де вказані методи та облікові дані бази даних.

```
###> doctrine/doctrine-bundle ###
DATABASE_URL="mysql://user:password@127.0.0.1:8889/diploma_db?serverVersion=5.7&charset=utf8mb4"
###< doctrine/doctrine-bundle ###
```

Рисунок 3.19 – Налаштування зв’язку з базою даних у .env файлі

Після підключення необхідно створити таблиці у базі даних. Ще однією особливістю і перевагою Symfony є інтерактивний помічник(рис. 3.20) створення сутностей, які потім переводяться за допомогою міграцій у таблиці в базі даних, таким чином генерується і таблиця і код, який відповідає за сутність та роботу ORM з нею.

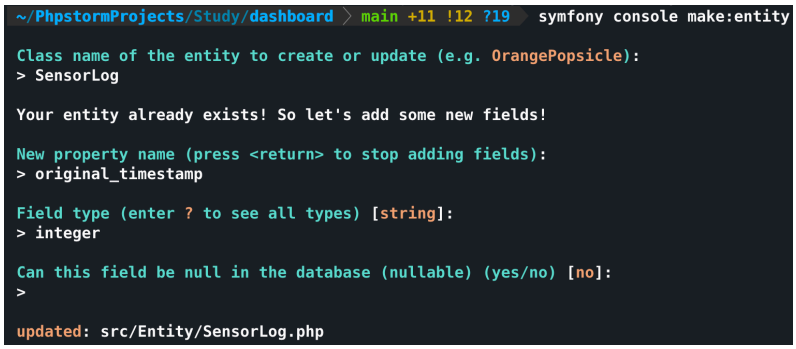


Рисунок 3.20 – Інтерактивний помічник Symfony для створення сутностей

ORM – це об’єктно-реляційний мапер, який є інструментом, який допомагає конвертувати дані між системами несумісних типів за допомогою об’єктно-орієнтованих мов програмування. У випадку PHP, ORM допоможе перетворити дані з реляційної бази даних, наприклад MySQL, в об'єкти, які можна використовувати в PHP. Це дозволить спростити маніпулювання та керування даними в PHP. Використання ORM має багато переваг, наприклад можливість легше працювати зі складними структурами даних, підвищена гнучкість під час роботи з різними типами даних та покращена продуктивність при роботі з великими наборами даних. Крім того, ORM можуть допомогти зменшити кількість коду, що повторюється.

Після створення сутностей необхідно згенерувати та запустити міграції, які додадуть всі зміни до бази даних за допомогою SQL запитів.

Міграції – це спосіб збереження змін у вашу схему бази даних в окремі автономні блоки, які можна застосувати до бази даних, щоб оновити її структуру.

Міграції зазвичай використовуються, коли потрібно внести зміни в схему бази даних, наприклад додати новий стовпець або таблицю або змінити тип даних існуючого стовпця. Міграції можуть приміняться до будь-яких типів бази даних, головне зберігати правильність написання SQL коду для кожного типу.

Встановлено модуль для роботи з RabbitMQ для PHP за допомогою Composer командою «composer require php-amqplib/php-amqplib». Для отримання інформації з блокчейну реалізований клієнт RabbitMQ, який прослуховую канал для передачі даних «node-dash». Написано обробник(рис. 3.21), який відповідає за перевірку та збереження інформації до бази даних та фільтрації контенту.

```php
private function processMessage(AMQPMessage $msg): bool {
    $msgData = $msg->getBody();

    if(!json_decode($msgData, associative: true)) return false;

    $msgData = json_decode($msgData, associative: true);

    if(!$this->isDataValid($msgData)) return false;

    $sensors = $this->sensorRepository->findAll();
    foreach ($sensors as $sensor){
        if($sensor->getMacAddress() == $msgData['MAC']){
            $sensorLog = new SensorLog();
            $sensorLog->setSensor($sensor);
            $sensorLog->setValue($msgData['value']);
            $sensorLog->setOriginalTimestamp($msgData['time']);

            $this->sensorLogRepository->add($sensorLog);

            return true;
        }
    }
    return false;
}
```

Рисунок 3.21 – Обробник, який відповідає за перевірку та збереження інформації

Таким чином обробник отримує дані, перевіряє їх валідність та після цього шукає відповідність у зареєстрованих сенсорах та зберігає інформацію.

Для взаємодії з даними, додавання датчиків і будь-якої іншої маніпуляції з базою даних необхідно створити CMS панель з повноцінним функціоналом.

Для цього можна використати ще один пакет для Symfony, який називається EasyAdmin.

EasyAdmin – це пакет Symfony, який забезпечує простий і легкий спосіб створення CMS для програм Symfony.

Він заснований на Doctrine ORM і забезпечує простий і потужний інтерфейс з великою кількістю різноманітного функціоналу.

Пакет включає в себе ряд функцій, які дозволяють легко розпочати роботу:

- Інтерфейс користувача, який є одночасно простим і потужним.
- Вбудований контролер CRUD для кожної сутності.
- Вбудована пошукова система.

Він має ряд вбудованих функцій, які можна вмикати або вимкнути за потреби, а також дозволяє створювати власний додатковий функціонал.

Для встановлення EasyAdmin необхідно використати Composer з командою «composer require easycorp/easyadmin-bundle»(рис. 3.22).

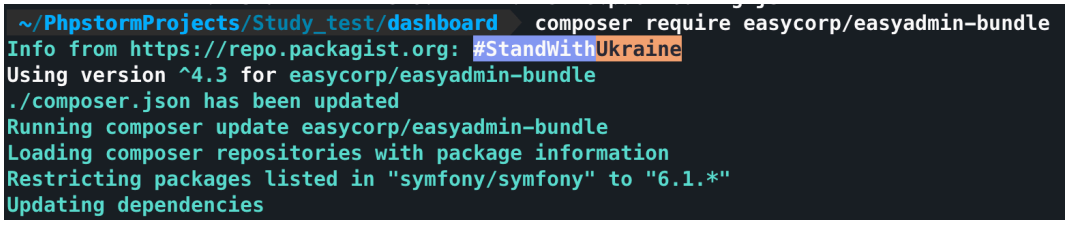


Рисунок 3.22 – Встановлення EasyAdmin за допомогою Composer

EasyAdmin дуже легко налаштовується та має дуже велику кількість налаштувань.

Створено модель та CRUD систему для сутності датчика.

CRUD – це абревіатура для Create, Read, Update та Delete. Операції CRUD – це основні операції, які можна виконувати над базою даних.

Операції CRUD є основою багатьох веб-додатків. Наприклад, коли користувач заповнює форму на веб-сайті, дані форми зазвичай зберігаються в базі даних. Коли користувач надсилає форму, виконується операція CRUD для створення нового запису в базі даних. Аналогічно, коли користувач переглядає список записів на веб-сайті, виконується операція CRUD для читання записів із бази даних.

CRUD від EasyAdmin містить усі необхідні функції. Таблиця для виведення усіх записів конкретної таблиці(рис. 3.23) має необхідні налаштування та зображення записів.

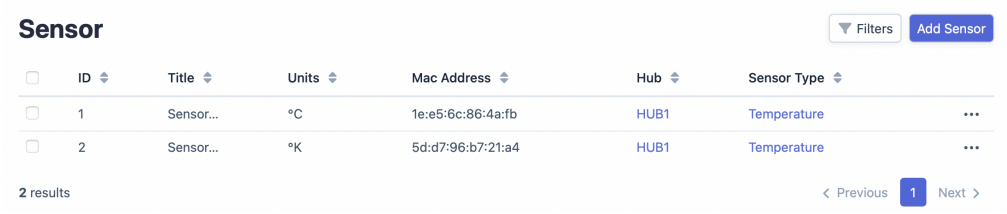


Рисунок 3.23 – Таблиця з зображенням усіх записів

На сторінці також присутня фільтрація(рис. 3.24) за звичайними полями та за зв'язаними сутностями, сортування за будь-якими полями.

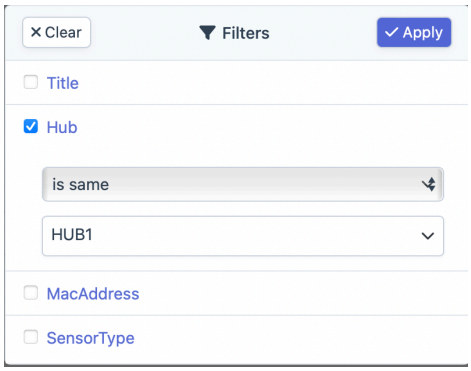


Рисунок 3.24 – Фільтрування за зв'язаною сутністю

За бажанням можна додати новий запис, редагувати чи видалити старий. На сторінці створення та редагування у користувача має можливість підставити сутність зі зв'язаної таблиці(рис. 3.25). На кожній формі обов'язково присутня валідація, щоб не допустити помилки зі сторони користувача при заповненні.

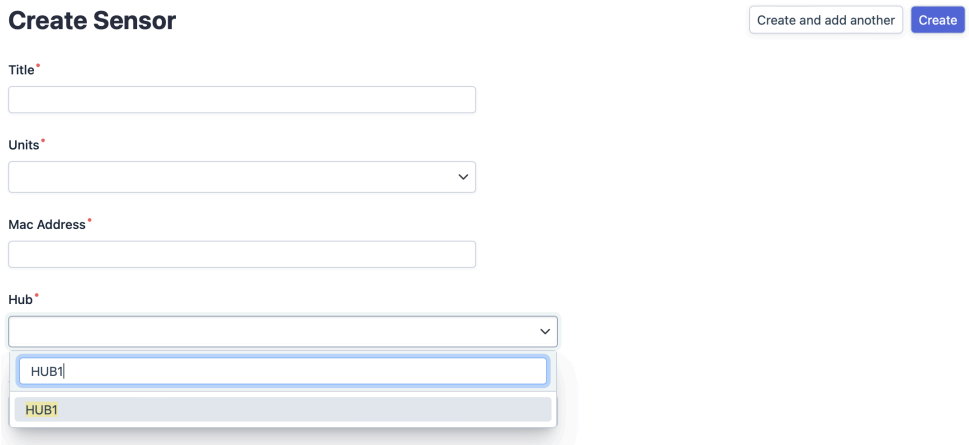


Рисунок 3.25 – Можливість підставляти сутність з пов'язаної таблиці при створенні чи редагуванні запису

Таким чином створено систему, що дозволяє користувачу легко маніпулювати даними та переглядати їх, але також необхідно адміністративну панель захистити. Для цього було створено сутність Admin та додано до бази даних MySQL. За допомогою цієї сутності можна керувати користувачами та проводити процес авторизації та автентифікації без жодних проблем. Було створено сторінку з формою

авторизації(рис 3.26) та можливістю «запам’ятовувати» користувача для його зручності.

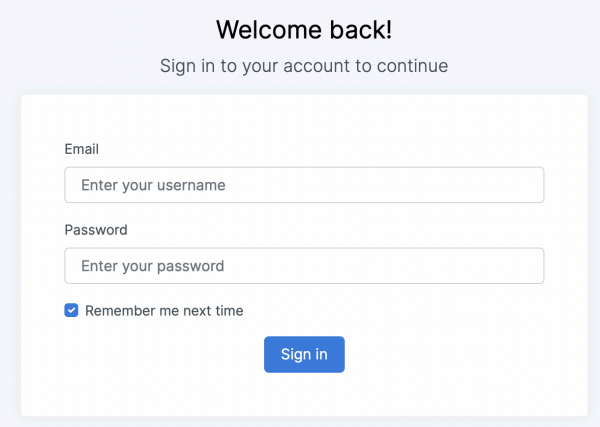


Рисунок 3.26 – Форма авторизації для користувача, що надає доступ до адміністративної панелі

Дані отримані від датчиків необхідно аналізувати. Одним з типів аналізу даних є візуалізація даних.

Візуалізація даних – це графічне представлення даних. Це передбачає створення зображень, які повідомляють глядачам зображень відносини між представленими даними. Цю комунікацію можна здійснити за допомогою використання графіки та технологій.

Існує багато різних способів візуалізації даних. Одними з найпоширеніших методів є діаграми, графіки.

Діаграми – це тип візуалізації даних, який використовує графічні зображення для відображення взаємозв’язків між наборами даних. Поширені типи діаграм включають лінійні діаграми, стовпчасті діаграми та кругові діаграми.

Графіки – це інший тип візуалізації даних, який використовує лінії або криві для відображення зв’язків між наборами даних. Поширені типи графіків включають лінійні графіки та графіки розсіювання.

Для побудови графіків та діаграм використано JavaScript бібліотеку, яка називається ChartJS.

ChartJS – це потужна, багатофункціональна бібліотека діаграм JavaScript, яка дозволяє створювати інтерактивні діаграми, які можна налаштувати.

За допомогою ChartJS можна легко створювати лінійні, стовпчасті, кругові та інші типи діаграм.

ChartJS доступний як безкоштовна бібліотека з відкритим кодом.

ChartJS підтримує різноманітні типи діаграм, такі як:

- Лінійні діаграми.
- Гістограми.
- Секторні діаграми.
- Бульбашкові діаграми.
- Точкові діаграми.
- Динамічні діаграми.

За допомогою цієї бібліотеки було створено лінійно-точковий графік та пончикову діаграму(рис. 3.27).

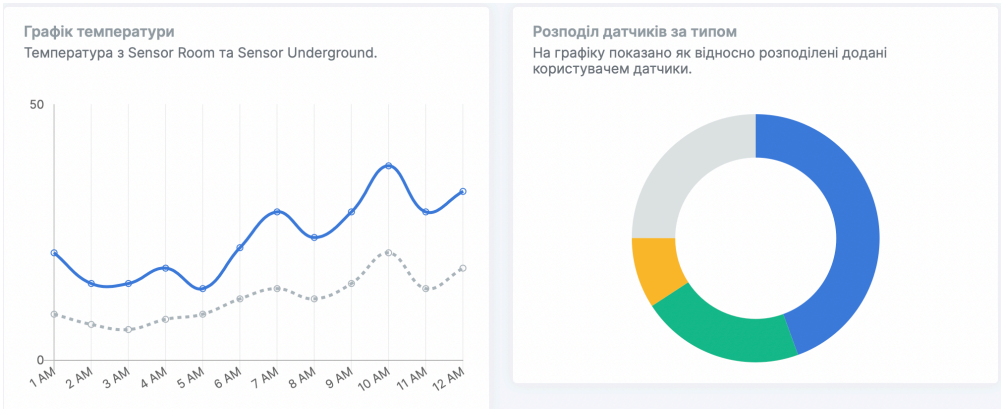


Рисунок 3.27 – Графіки згенеровані на основі отриманих даних за допомогою ChartJS

При наведенні на будь-яку точку на графіку або сектор діаграми з’являється підказка з легендою конкретної ділянки діаграми. Таким чином реалізовано візуалізацію отриманих даних. Графіки майже усіх типів можна відобразити для конкретного датчика, або для групи датчиків.

3.6. Висновки до розділу

У розділу в загальному міститься інформація щодо процесу розробки системи зберігання та обробки даних, отриманих з датчиків IoT за допомогою технології Blockchain на основі проекту створеного у розділі 2. Таким чином було розроблено:

- імітаційну моделі генерації даних («датчик»);
- систему агрегації та проксування згенерованих даних(«хаб»);
- прошарок для блокчейну;

- блокчейн для обробки та збереження даних;
- адміністративну панель з візуалізацією даних та з підключенням бази даних.

ВИСНОВКИ

Проаналізовано більше 20 аналогів системи, включаючи як системи, які використовують Blockchain, так і ті, що не використовують. Проведено аналіз основних типів існуючих датчиків IoT та описано основні принципи роботи кожного з них. Досліджено та проаналізовано нормативні документи, які описують вимоги до використання та створення комерційних приміщень/будівель. Вивчено питання аналітики даних безпосередньо у IoT системах та методи аналізу таких даних.

1. Проаналізовано методи обробки та аналізу даних, які отримані з датчиків IoT, у які входять візуалізація, статистичний аналіз, інтелектуальний аналіз та машинне навчання. На основі проведеного огляду та аналізу протоколів IoT обрано MQTT та AMQP(RabbitMQ) протоколи для передачі даних у системі.

2. Спроектовано архітектуру системи збереження та обробки даних від датчиків IoT за допомогою технології Blockchain. Було проведено вибір технологій для реалізації та впровадження спроектованої системи збереження та аналізу.

3. Розроблено алгоритм роботи системи збереження та обробки даних від датчиків IoT за допомогою технології Blockchain.

4. Розроблено фізичну модель бази даних системи збереження та обробки даних.

5. Розроблено застосунок для зберігання та обробки даних, отриманих з датчиків IoT, а саме:

- імітаційну модель генерації даних(«датчик»);
- систему агрегації та проксування згенерованих даних(«хаб»);
- прошарок для блокчейну;
- блокчейн для обробки та збереження даних;
- адміністративну панель з візуалізацією даних та з підключенням бази даних.

У результаті виконання дослідження, спроектовано та створено IoT систему збереження та аналізу даних, що генерують датчики IoT, з використанням технології Blockchain з ціллю їх наступного аналізу за допомогою візуалізації результатів на основі проаналізованих даних аналогів.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. IoT Analytics: What Is It And How to Implement It in Your Organization [Електроний ресурс] – Режим доступу: https://mobidev.biz/blog/what-is-iot-data-analytics-how-to-implement-it-in-your-organization
2. What Is IoT Data? Here Are 11 Examples. [Електроний ресурс] – Режим доступу: https://builtin.com/big-data/iot-big-data-analytics-examples
3. Why Google's Nest sees beyond connected homes and the Internet of Things [Електроний ресурс] – Режим доступу: https://internetofbusiness.com/why-googles-nest-sees-beyond-connected-homes-and-the-internet-of-things/
4. AWS IoT Analytics [Електроний ресурс] – Режим доступу: https://aws.amazon.com/iot-analytics/
5. Офіційний сайт компанії Philips. Сторінка присвячена Інтернету речей [Електроний ресурс] – Режим доступу: https://www.usa.lighting.philips.com/internet-of-things
6. Офіційний сайт компанії Belkin [Електроний ресурс] – Режим доступу: https://www.belkin.com/us/smart-home/c/wemo/
7. Офіційний сайт компанії Fibaro. Сторінка про використання програми SmartThings від компанії Samsung [Електроний ресурс] – Режим доступу: https://www.fibaro.com/en/smartthings/
8. What is IFTTT [Електроний ресурс] – Режим доступу: https://ifttt.com/explore/new_to_ifttt
9. Stringify – Automation service for the Internet of Things (IoT) [Електроний ресурс] – Режим доступу: https://iotbyhvm.ooo/stringify/
10. Офіційний сайт компанії Wink [Електроний ресурс] – Режим доступу: https://www.wink.com
11. https://www.logitech.com/en-us/products/harmony/harmony-hub.915-000238.html
12. Офіційний сайт компанії Insteon [Електроний ресурс] – Режим доступу: https://www.insteon.com
13. Офіційний сайт компанії Control4 [Електроний ресурс] – Режим доступу: https://www.control4.com

14. Офіційний сайт компанії Thingspeak [Електроний ресурс] – Режим доступу: https://thingspeak.com
15. Офіційний сайт компанії Ubidots [Електроний ресурс] – Режим доступу: https://ubidots.com
16. Офіційний сайт компанії Losant [Електроний ресурс] – Режим доступу: https://www.losant.com
17. Офіційний сайт компанії InitialState [Електроний ресурс] – Режим доступу: https://www.initialstate.com
18. Офіційний сайт компанії Blynk [Електроний ресурс] – Режим доступу: https://blynk.io
19. Офіційний сайт компанії Particle [Електроний ресурс] – Режим доступу: https://www.particle.io
20. Офіційний сайт компанії Helium. Сторінка присвячена екосистемі, що розробляє компанія [Електроний ресурс] – Режим доступу: https://www.helium.com/ecosystem
21. Офіційний сайт компанії Exosite [Електроний ресурс] – Режим доступу: https://www.exosite.com
22. Презентація системи аналізу даних IoT від компанії Carriots [Електроний ресурс] – Режим доступу: https://www.altairsmartworks.com/newFrontend/img-carriots/press_room/CARRIOTS_technical_presentation.pdf
23. What is Xively and How it Helps in Building IoT Applications? [Електроний ресурс] – Режим доступу: https://www.it4nextgen.com/xively-building-iot-applications/
24. What is blockchain? [Електроний ресурс] – Режим доступу: https://www.euromoney.com/learning/blockchain-explained/what-is-blockchain
25. Guide to Blockchain [Електроний ресурс] – Режим доступу: https://www.investopedia.com/terms/b/blockchain.asp
26. What is blockchain? [Електроний ресурс] – Режим доступу: https://builtin.com/blockchain
27. Офіційний сайт компанії Filament [Електроний ресурс] – Режим доступу: https://filament.com

28. Офіційна документація технології IPFS [Електроний ресурс] – Режим доступу: https://docs.ipfs.io
29. Сторінка презентації продукту компанії Filament [Електроний ресурс] – Режим доступу: https://filament.com/company/product-information.html
30. База знань IOTA Filament [Електроний ресурс] – Режим доступу: https://wiki.iota.org/learn/about-iota/an-introduction-to-iota
31. Whitepaper Waltonchain [Електроний ресурс] – Режим доступу: https://www.waltonchain.org/pdf/5ee1c6b5b10cf.pdf
32. Про компанію та екосистему Ambrosus [Електроний ресурс] – Режим доступу: https://ambrosus.io/about
33. Top 15 Sensor Types Being Used Most By IoT Application Development Companies [Електроний ресурс] – Режим доступу: https://www.finoit.com/blog/top-15-sensor-types-used-iot/
34. How to Choose a Temperature Sensor for IoT [Електроний ресурс] – Режим доступу: https://medium.com/@temboo/how-to-choose-a-temperature-sensor-for-iot-350a79b7ac74
35. Choosing a Humidity Sensor: A Review of Three Technologies IoT [Електроний ресурс] – Режим доступу: https://www.fierceelectronics.com/components/choosing-a-humidity-sensor-a-review-three-technologies
36. Common Types of Pressure Sensors [Електроний ресурс] – Режим доступу: https://www.thomasnet.com/articles/instruments-controls/pressure-sensors/
37. All About Flow Sensors [Електроний ресурс] – Режим доступу: https://www.thomasnet.com/articles/instruments-controls/all-about-flow-sensors/
38. What are Motion Sensors And How Do They Work [Електроний ресурс] – Режим доступу: https://www.elprocus.com/working-of-different-types-of-motion-sensors/
39. IoT Devices – Proximity Sensors [Електроний ресурс] – Режим доступу: http://www.infiniteinformationtechnology.com/iot-devices-proximity-sensors
40. What is a light sensor? Types, Uses, Arduino Guide [Електроний ресурс] – Режим доступу: https://www.seeedstudio.com/blog/2020/01/08/what-is-a-light-sensor-types-uses-arduino-guide/

41. What is a sound sensor? – Uses, Arduino Guide, Projects [Електроний ресурс] – Режим доступу: https://www.seeedstudio.com/blog/2020/01/03/what-is-a-sound-sensor-uses-arduino-guide-projects/
42. What is a chemical sensor? [Електроний ресурс] – Режим доступу: https://www.fierceelectronics.com/electronics/what-a-chemical-sensor
43. ГРОМАДСЬКІ БУДИНКИ ТА СПОРУДИ Основні положення [Електроний ресурс] – Режим доступу: https://www.minregion.gov.ua/wp-content/uploads/2017/12/58.1.-DBN-V.2.2-9-2009.-Budinki-i-sporudi.-Gromadski-bu.pdf
44. Порядок введення в експлуатацію будівель та споруд [Електроний ресурс] – Режим доступу: https://tuexpert.com.ua/poriadok-vvoda-v-ekspluataciyu
45. Документальне оформлення введення в експлуатацію промислових споруд [Електроний ресурс] – Режим доступу: https://tuexpert.com.ua/oformleniye-prom-objektov
46. IoT Data Visualization For Better Understanding Of The Big Data [Електроний ресурс] – Режим доступу: https://blog.contus.com/iot-data-visualization-for-iot-platforms-applications/
47. Chart.js + Next.js = Beautiful, Data-Driven Dashboards. How to create them fast and efficiently. [Електроний ресурс] – Режим доступу: https://towardsdev.com/chart-js-next-js-beautiful-data-driven-dashboards-how-to-create-them-fast-and-efficiently-a59e313a3153
48. 7 Types of Statistical Analysis: Definition and Explanation [Електроний ресурс] – Режим доступу: https://www.analyticssteps.com/blogs/7-types-statistical-analysis-definition-explanation
49. Data Mining Methods [Електроний ресурс] – Режим доступу: https://www.educba.com/data-mining-methods/
50. What Is Machine Learning: Definition, Types, Applications And Examples [Електроний ресурс] – Режим доступу: https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/
51. A Complete Guide to IoT Protocols & Standards In 2021 [Електроний ресурс] – Режим доступу: https://www.nabto.com/guide-iot-protocols-standards/

52. Internet of Things statistics for 2022 - Taking Things Apart [Електроний ресурс] – Режим доступу: https://dataprot.net/statistics/iot-statistics/

ДОДАТКИ

ДОДАТОК А

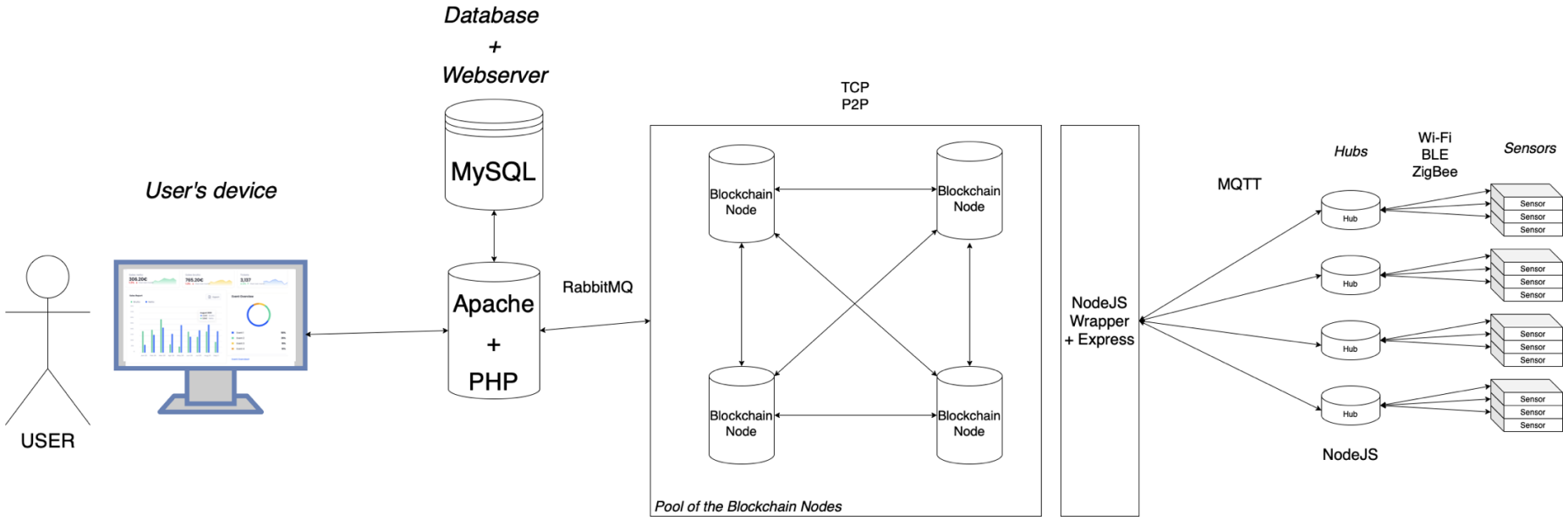
Архітектура спроектованої системи

Листів 1

Розробник Бутенко К.О.

Керівник Ліснеський Р.В.

Київ - 2022


Database
+
Webserver
MySQL
User's device
USER
Apache
+
PHP
RabbitMQ
TCP
P2P
Blockchain Node
Blockchain Node
Blockchain Node
Blockchain Node
Pool of the Blockchain Nodes
NodeJS
Wrapper
+ Express
MQTT
Hubs
Wi-Fi
BLE
ZigBee
Sensors
Hub
Hub
Hub
Hub
Sensor
NodeJS

ДОДАТОК Б

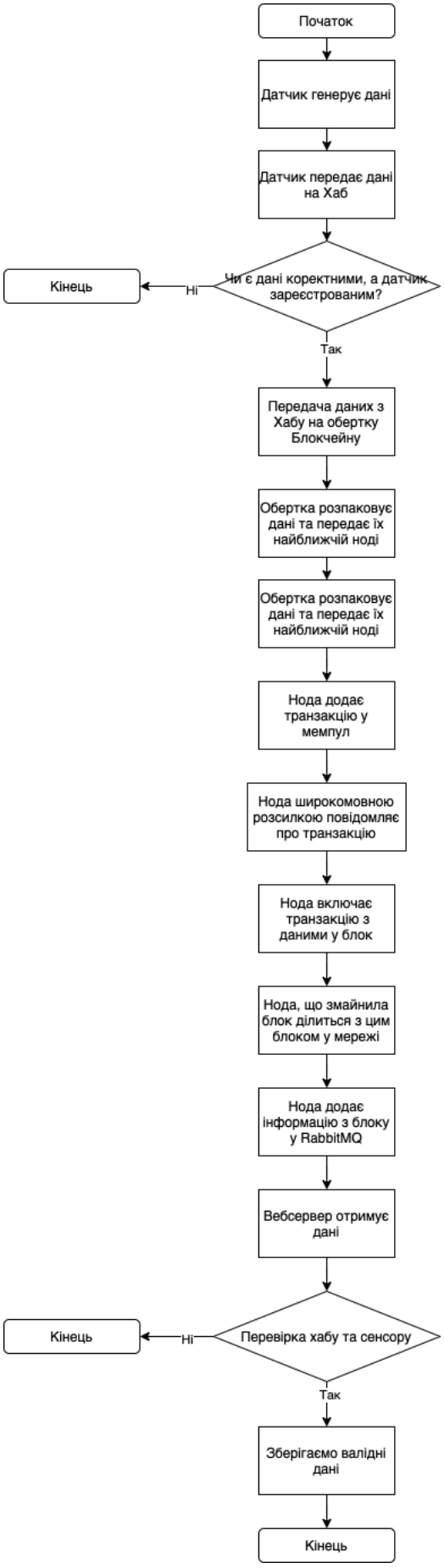
Алгоритм роботи спроектованої системи

Листів 1

Розробник Бутенко К.О.

Керівник Ліснеський Р.В.

Київ - 2022


Початок
Датчик генерує дані
Датчик передає дані на Хаб
Чи є дані коректними, а датчик зареєстрованим?
Ні
Кінець
Так
Передача даних з Хабу на обертку Блокчейну
Обертка розпаковує дані та передає їх найближчій ноді
Обертка розпаковує дані та передає їх найближчій ноді
Нода додає транзакцію у мемпул
Нода широкомовною розсилкою повідомляє про транзакцію
Нода включає транзакцію з даними у блок
Нода, що змайнила блок ділиться з цим блоком у мережі
Нода додає інформацію з блоку у RabbitMQ
Вебсервер отримує дані
Перевірка хабу та сенсору
Ні
Кінець
Так
Зберігаємо валідні дані
Кінець

ДОДАТОК В

Презентація кваліфікаційної роботи

Листів 6

Розробник Бутенко К.О.

Керівник Ліснеський Р.В.

Київ - 2022

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

керівник к.т.н., доцент
Лісневський Ростислав Валерійович

студент групи ІР-41
Бутенко Костянтин Олександрович

ТЕМА ДИПЛОМНОЇ РОБОТИ

«Розробка проекту системи аналізу даних датчиків IoT з використанням технології Blockchain»

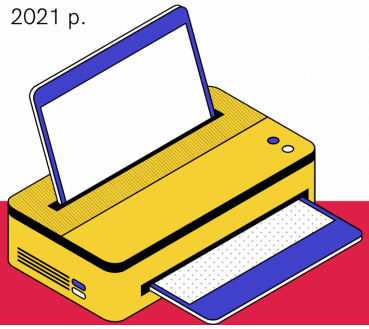
ЦІЛІ РОБОТИ

- Аналіз можливості застосування технології Blockchain для збереження даних згенерованих IoT.
- Проектування системи збереження та аналізу інформації за допомогою технології Blockchain.
- Розробка системи Blockchain, яка може зберігати та оброблювати вхідну інформацію з датчиків.
- Спроектувати та реалізувати модель передачі даних між усіма компонентами системи.
- Реалізувати застосунок для отримання інформації з датчиків та візуалізуації даних.

КОРОТКІ ВІДОМОСТІ

Публікації

- Міжнародна науково-практична конференція «Прикладні системи та технології в інформаційному суспільстві» - 30 вересня 2021 року
- VIII Міжнародна науково-технічна Internet-конференція «Сучасні методи, інформаційне, програмне та технічне забезпечення систем керування організаційно-технічними та технологічними комплексами» (НУХТ) - 25-26 листопада 2021 р.



Мета дипломної роботи

Аналіз існуючих систем збереження та обробки даних IoT, проектування та створення IoT системи збереження та аналізу даних, що генерують датчики IoT, за допомогою технології Blockchain з ціллю їх наступного аналізу за допомогою візуалізації результатів на основі проаналізованих даних.

Об'єкт дослідження

Технологія Blockchain, яка використовується для збереження даних з можливістю їх наступного аналізу.

Предмет дослідження

Система збереження та аналізу інформації, отриманої від датчиків IoT за допомогою технології Blockchain.

ПРИКЛАДИ РОЗГЛЯНУТИХ ДАТЧИКІВ

СТЕК ТЕХНОЛОГІЙ ТА ПРОТОКОЛІВ

PHP

SYMFONY

MQTT

RABBITMQ

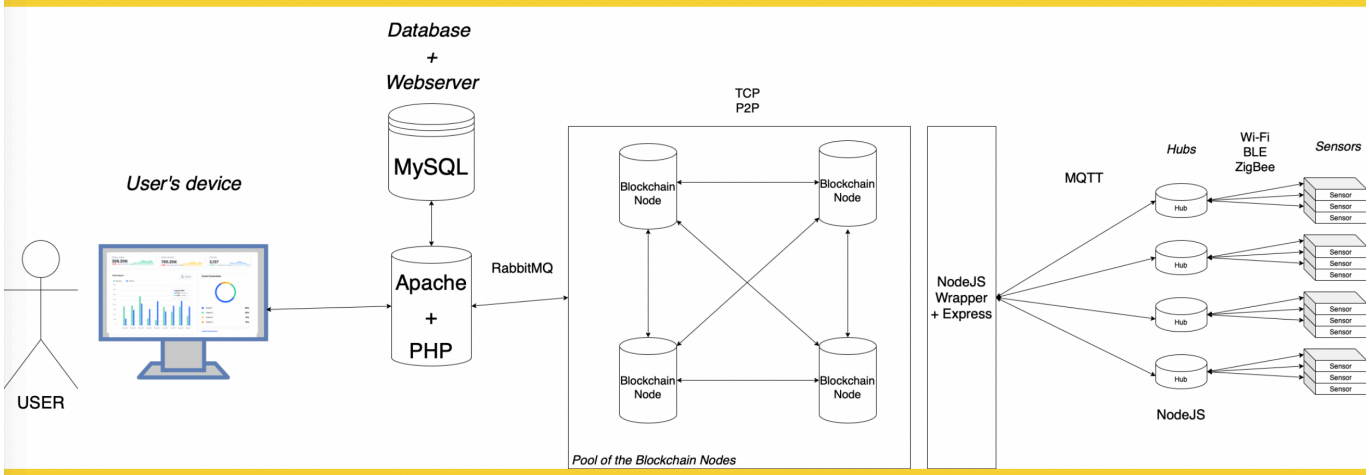
JAVASCRIPT

NODEJS

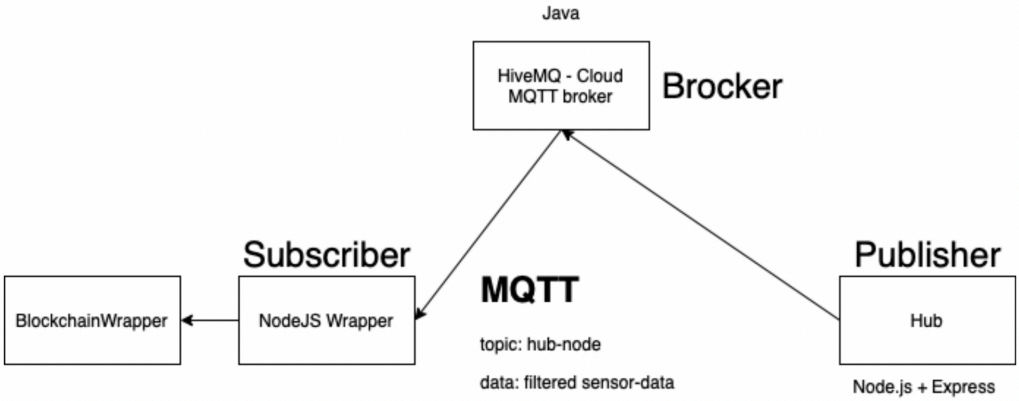
APACHE

CHARTJS

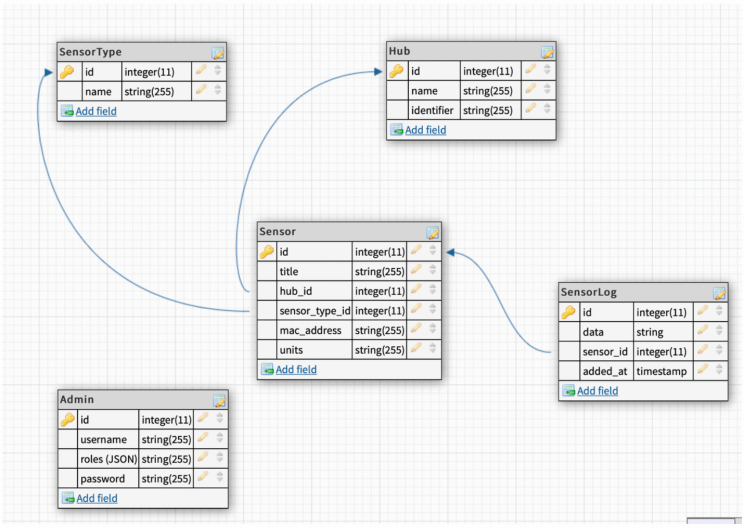
АРХІТЕКТУРА СИСТЕМИ ОТРИМАННЯ ТА ОБРОБКИ ДАНИХ ВІД ДАТЧИКІВ З ВИКОРИСТАННЯМ BLOCKCHAIN



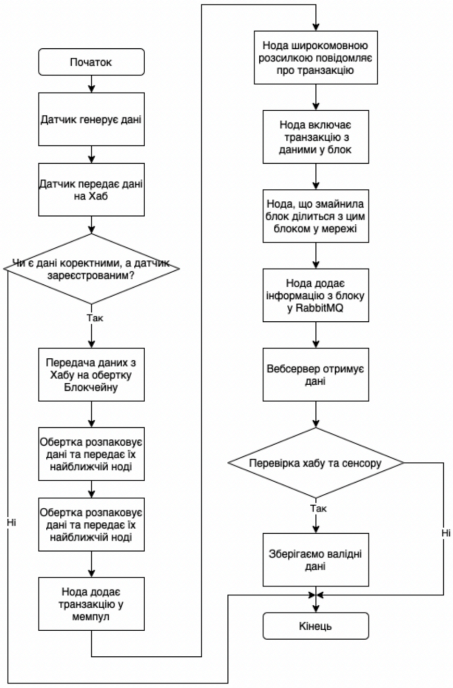
МОДЕЛЬ КОМПОНЕНТА ПЕРЕДАЧІ ДАНИХ ЗА ДОПОМОГОЮ MQTT ТА HIVEMQ



ФІЗИЧНА МОДЕЛЬ БАЗИ ДАНИХ СИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ



АЛГОРИТМ РОБОТИ СИСТЕМИ



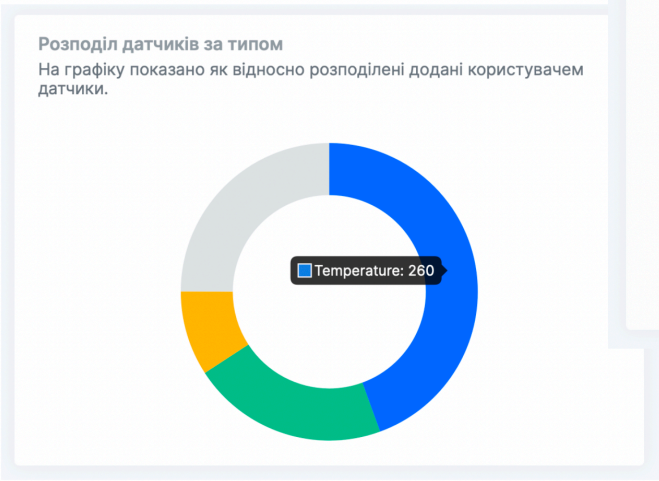
ПРИКЛАД РОБОТИ СИСТЕМИ

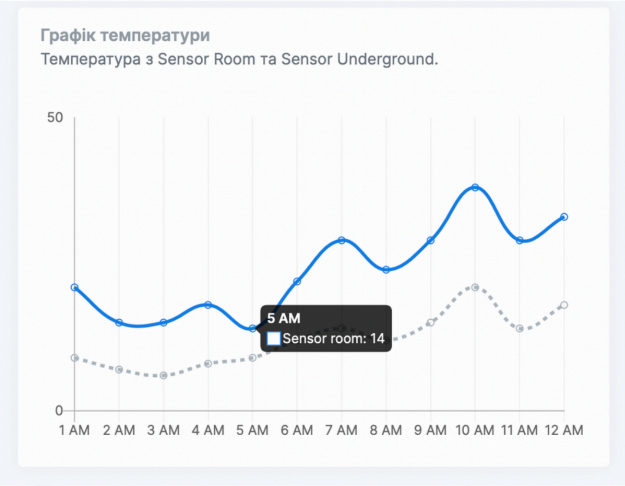
```
============
Launching the Sensor
============
[07-06-2022 05:24:09] Sent from 7d:ad:2e:05:66:3d  :---:  13
[07-06-2022 05:24:16] Sent from 2f:58:20:aa:20:77  :---:  29
[07-06-2022 05:24:23] Sent from 89:70:78:3f:72:dd  :---:  25
[07-06-2022 05:24:30] Sent from 01:76:7a:a9:66:48  :---:  26
[07-06-2022 05:24:35] Sent from 85:7c:9f:7b:d7:6c  :---:  29
[07-06-2022 05:24:44] Sent from 87:f9:55:b5:09:52  :---:  16
```


{"index":1,"hash":"0000b12b2c22a0d3eb97a405bf78385319196e95e45b3f2a67e7cbfbf3ca5e4f","previousHash":"0
","createdAt":"2022-06-07 05:24:09","data":"
{\"value\":\"13\",\"MAC\":\"7d:ad:2e:05:66:3d\",\"time\":\"1654579449\"}","difficulty":16,"nonce":8462
4},
{"index":2,"hash":"00003f1f3b35d960c1ec53900a686b2c2e5f5a07cf7ea2721e982c39022dd2b8","previousHash":"0
000b12b2c22a0d3eb97a405bf78385319196e95e45b3f2a67e7cbfbf3ca5e4f","createdAt":"2022-06-07
05:24:16","data":"
{\"value\":\"29\",\"MAC\":\"2f:58:20:aa:20:77\",\"time\":\"1654579456\"}","difficulty":16,"nonce":1000
34},
{"index":3,"hash":"0000e62bf850ed080e2de103c1773d3f26a5f1e052721d9215c8891411ebd1b0","previousHash":"0
0003f1f3b35d960c1ec53900a686b2c2e5f5a07cf7ea2721e982c39022dd2b8","createdAt":"2022-06-07
05:24:23","data":"
{\"value\":\"25\",\"MAC\":\"89:70:78:3f:72:dd\",\"time\":\"1654579463\"}","difficulty":16,"nonce":1173
58},
{"index":4,"hash":"0000891c2e6ebbe5361a127f4142db3de827a27e262da390e136d0956d8fa60f","previousHash":"0
000e62bf850ed080e2de103c1773d3f26a5f1e052721d9215c8891411ebd1b0","createdAt":"2022-06-07
05:24:30","data":"
{\"value\":\"26\",\"MAC\":\"01:76:7a:a9:66:48\",\"time\":\"1654579470\"}","difficulty":16,"nonce":3627
16},


```
~ mqtt sub -h 7aeff168314247398614a2d98d9dbc2d.s1.eu.hivemq.cloud -p 8883 -s -u _hub1 -pw -t 'hub-node'
Enter value for --password (The password for authentication):
{"value":"13","MAC":"7d:ad:2e:05:66:3d","time":"1654579449"}
{"value":"29","MAC":"2f:58:20:aa:20:77","time":"1654579456"}
{"value":"25","MAC":"89:70:78:3f:72:dd","time":"1654579463"}
{"value":"26","MAC":"01:76:7a:a9:66:48","time":"1654579470"}
{"value":"29","MAC":"85:7c:9f:7b:d7:6c","time":"1654579475"}
{"value":"16","MAC":"87:f9:55:b5:09:52","time":"1654579484"}
{"value":"28","MAC":"5d:d7:96:b7:21:a4","time":"1654579493"}
{"value":"28","MAC":"d6:b7:08:13:c8:6c","time":"1654579499"}
{"value":"17","MAC":"ae:54:2e:f8:91:cb","time":"1654579506"}
{"value":"29","MAC":"c7:de:7a:9b:16:d9","time":"1654579513"}
{"value":"17","MAC":"01:76:7a:a9:66:48","time":"1654579518"}
{"value":"10","MAC":"7d:ad:2e:05:66:3d","time":"1654579527"}
```

ВІЗУАЛІЗАЦІЯ ДАНИХ





ВИСНОВКИ

1. Проаналізовано методи обробки та аналізу даних. На основі проведеного огляду та аналізу протоколів IoT обрано MQTT та AMQP(RabbitMQ) протоколи для передачі даних у системі.
2. Спроектовано архітектуру системи збереження та обробки даних від датчиків IoT за допомогою технології Blockchain. Було проведено вибір технологій для реалізації.
3. Розроблено алгоритм роботи системи збереження та обробки даних від датчиків IoTза допомогою технології Blockchain.
4. Розроблено фізичну модель бази даних системи збереження та обробки даних.
5. Розроблено застосунок для зберігання та обробки даних, отриманих з датчиків IoT, а саме
 · імітаційну модель генерації даних(«датчик»);
 · систему агрегації та проксування згенерованих даних(«хаб»);
 · прошарок для блокчейну;
 · блокчейн для обробки та збереження даних;
 · адміністративну панель з візуалізацією даних та з підключенням бази даних.

У результаті виконання дослідження, спроектовано та створено IoT систему збереження та аналізу даних, що генерують датчики IoT, з використанням технології Blockchain з ціллю їх наступного аналізу за допомогою візуалізації результатів на основі проаналізованих даних аналогів.