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

Радіотехнічний факультет

Кафедра прикладної радіоелектроніки

«На правах рукопису»
УДК 194.235

До захисту допущено:
В.о. зав. кафедри
_____Андрій МОВЧАНЮК
«___»_____20__ р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інтелектуальні технології радіоелектронної техніки»

зі спеціальності 172 Електронні комунікації та радіотехніка

на тему: «Система моніторингу та керування IoT пристроями на виробництві»

Виконав (-ла):
студент (-ка) 2 курсу, групи РЕ-31мп
___Негода Нікіта Володимирович_____ _____
(Прізвище ім'я по батькові)
Керівник к.т.н., доцент Мосійчук Віталій Сергійович _____
(Посада, науковий ступінь ,вчене звання)

Рецензент к.т.н., старший викладач Михайленко
Максим Вікторович _____
(Посада, науковий ступінь ,вчене звання)

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

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Радіотехнічний факультет
Кафедра прикладної радіоелектроніки

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 Електронні комунікації та радіотехніка

Освітньо-професійна програма «Інтелектуальні технології радіоелектронної техніки»

ЗАТВЕРДЖУЮ
В.о. зав. кафедри
_____ Андрій МОВЧАНЮК
«__» _____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студента
Негода Нікіта Володимирович

(Прізвище ім'я по батькові)

1.Тема дисертації « Система моніторингу та керування IoT пристроями на виробництві» науковий керівник дисертації к.т.н. Мосійчук Віталій Сергійович затверджені наказом по університету від «08» листопада 2024 р. № 5024-с

2. Термін подання студентом дисертації 18 грудня 2024 року

3. Об'єкт дослідження процеси моніторингу та керування IoT пристроями

4. Вихідні дані архітектура IoT; протокли передачі даних; засоби моніторингу та керування IoT пристроями на виробництві; Docker;

5. Перелік завдань, які потрібно розробити аналіз архітектури IoT та основних бездротових протоколів; розгляд вимог до системи; розгляд мінікомп'ютерів; розгляд та вибір протоколу для передачі даних в системі; розгляд хмарних платформ; розробка прототипу на основі мінікомп'ютера; розробка алгоритму та рекомендацій для впровадження подібних систем;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу Презентаційні матеріали

7. Орієнтовний перелік публікацій

9. Дата видачі завдання 05 вересня 2024 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Отримання теми магістерської дисертації	05.09.2024	
2	Розробка плану магістерської дисертації	05.09.2024 - 16.09.2024	
3	Аналітичний огляд інформації та літератури для роботи	16.09.2024 - 26.09.2024	
4	Проаналізувати архітектуру та протоколи IoT	26.09.2024 - 10.10.2024	
5	Визначення вимог до системи	10.10.2024 - 16.10.2024	
6	Вибір структурної схеми для системи	16.10.2024 - 23.10.2024	
7	Визначення апаратного забезпечення та протоколу для передачі даних в системі	23.10.2024 - 06.11.2024	
8	Розробка прототипу	06.11.2024 - 20.11.2024	
9	Розробка рекомендацій для впровадження системи	20.112024 - 04.12.2024	
10	Оформлення роботи, підготовка презентації та захист	04.12.2024 - 18.12.2024	

Студент Нікіта НЕГОДА

Науковий керівник Віталій МОСІЙЧУК

РЕФЕРАТ

Пояснювальна записка до магістерської дисертації містить 79 сторінок, 27 рисунків, 4 таблиць, 38 джерел літератури.

Актуальність теми: У світі вже давно набув популярності тренд на автоматизацію, а також віддалене управління і моніторинг об'єктів виробництва. Важливо враховувати, що потреби автоматизації виникають не лише у великих підприємств із значними бюджетами, але й у малих компаній, для яких використання готових рішень може бути надто дорогим.

Саме з цієї причини обрано дану тему, щоб шляхом аналізу технологій та протоколів визначити мінімально необхідного апаратного і програмного забезпечення для забезпечення роботи системи управління та моніторингу IoT-пристроїв на виробництві, для чого необхідно розробити прототип та зазначити рекомендації для більш легкого впровадження подібних IoT-систем на виробництві.

Мета дипломної роботи: розробка прототипу та рекомендацій щодо впровадження системи моніторингу та керування IoT пристроями на виробництві.

Об’єкт дослідження: процеси моніторингу та керування IoT пристроями.

Предмет дослідження: протоколи, апаратне та програмне забезпечення IoT систем для моніторингу та керування IoT пристроями.

Методи дослідження: системний аналіз, методи порівняння, моделювання.

Практичне значення одержаних результатів: на базі прототипу можна створити більш розширену версію системи та застосувати її для інших задач. Надані рекомендації стосовно впровадження системи моніторингу та керування IoT пристроями на виробництві. Розроблено алгоритм роботи

IoT системи, з описом мінімально необхідного апаратного та програмного забезпечення.

Ключові слова: IoT, моніторинг, керування, виробництво, IoT, Raspberry, протокол.

ABSTRACT

The explanatory note to the master's thesis contains 79 pages, 27 figures, 4 tables, and 38 sources of literature.

Relevance of the topic: The trend towards automation, as well as remote management and monitoring of production facilities, has long been popular in the world. It is important to consider that such needs arise not only in large enterprises with significant budgets, but also in small companies, for which the use of ready-made solutions may be too expensive.

It is for this reason that this topic was chosen in order to determine, through the analysis of technologies and protocols, the minimum necessary hardware and software to ensure the operation of the management and monitoring system for IoT devices in production. To develop a prototype and provide recommendations for easier implementation of such IoT systems in production.

The purpose of the thesis: development of a prototype of a monitoring and management system for IoT devices in production and recommendations for the implementation of a monitoring and management system for IoT devices in production.

Object of research: monitoring and management system for IoT devices.

Subject of research: protocols, hardware and software of IoT systems, which are used to monitor and control IoT devices.

Research methods: system analysis, comparison method, modeling.

Practical significance of the results: based on the prototype, a more advanced version of the system can be created and used for other tasks. Recommendations are given for the implementation of a system for monitoring and controlling IoT devices in production. A flowchart of the algorithm for selecting an IoT system has been developed, with a description of the minimum required hardware and software.

Keywords: IoT, monitoring, control, production, IoT, Raspberry, protocol.

ЗМІСТ

Перелік умовних скорочень

IoT - Internet of Things

MQTT - Message Queue Telemetry Transport

HTTP - Hypertext Transfer Protocol

CoAP - Constrained Application Protocol

AWS - Amazon Web Services

AMQP - Advanced Message Queuing Protocol

NATS - Neural Autonomic Transport System

BLE - Bluetooth Low Energy

ВСТУП

В наш час активно впроваджують різноманітні системи управління та моніторингу виробництва, з використанням технологій Інтернету речей (IoT). IoT - це свого роду технологічний прорив, що змінює сприйняття навколишнього світу та способи взаємодії з ним.

Технології Інтернету речей суттєво впливають на повсякденне життя, інтегруючи все більше різних пристроїв і систем у глобальну мережу, від звичайних побутових приладів, таких як термометри та сенсори вологості, до складних систем моніторингу та управління. Можна вважати, що IoT охоплює всі аспекти життя, відкриваючи нові можливості й водночас ставлячи нові виклики.

Конструкція цих систем виявлення в основному базується на зборі та обміні інформацією між пристроями, сенсорами, електронними компонентами з можливістю подальшого аналізу даних та візуалізації. Цінним навіть є простий евристичний аналіз та інтерпретацію зібраних даних користувачами. Ці системи мають ряд переваг, включаючи низьку вартість, значні обсяги зібраних даних, енергоефективність, зручність, легкість переміщення, легкість розгортання, здатність передавати дані іншим пристроям, збір даних у режимі реального часу, здатність автоматично запускати нові дії тощо.

Тому не дивно, що розвиток IoT є одним з ключових факторів для підвищення продуктивності та ефективності на підприємствах. Застосування подібних систем дозволяє покращити контроль за виробничими процесами, що в свою чергу дозволяє підвищити конкурентоспроможність, зменшити витрати на обслуговування обладнання, покращення безпеки праці.

Актуальність роботи полягає в розвитку підходів та формулюванні рекомендацій для впровадження систем автоматизації виробництва

невеликих підприємств з мінімальним бюджетом, для яких використання готових комерційних рішень є недоступним.

Саме тому обрано дану тематику аналізу технологій та протоколів, визначення мінімального необхідного апаратного і програмного забезпечення для забезпечення роботи системи управління та моніторингу IoT-пристроїв на виробництві. Розробити прототип та зазначити рекомендації для більш легкого впровадження подібних IoT-систем на виробництві. Оскільки правильний вибір апаратного та програмного забезпечення дозволяє знизити витрати на розгортання системи, дозволяє забезпечити гнучкість та підібрати той варіант системи, які найбільш за все підійде для виробництва враховуючи його специфіку.

Метою дипломної роботи є розробка прототипу системи та надання рекомендацій щодо впровадження системи моніторингу та керування IoT пристроями на виробництві.

Часткові завдання:

1. Проаналізувати архітектуру та протоколи IoT
2. Визначити вимоги до системи
3. Вибір структурної схеми для системи
4. Визначення апаратного забезпечення та протоколу для передачі даних в системі
5. Розробка прототипу
6. Розробка рекомендацій для впровадження системи

РОЗДІЛ 1
АНАЛІЗ ТЕХНОЛОГІЇ ІОТ

1.1 Опис технології IoT

За останні роки IoT лише закріпив свою позицію як одна з ключових технологій. Завдяки вбудованим пристроям ми маємо змогу з'єднувати різноманітні об'єкти – від автомобілів та побутової техніки до різних датчиків – з Інтернетом, що дає змогу безперервно здійснювати обмін інформацією між людьми та машинами. Хмарні обчислення в свою чергу дозволяють мінімально залучати людей для роботи з обміном та обробкою даних. У такому взаємопов'язаному середовищі цифрові системи здатні відстежувати, контролювати та налаштовувати всі взаємодії між підключеними об'єктами.

Якщо розглянути точніше, IoT — це концепція, що полягає в об’єднанні різноманітних фізичних пристроїв через інтернет, забезпечуючи їхню взаємодію та обмін даними. Основна ідея полягає у створенні «розумних» об’єктів, які взаємопов’язані та утворюють інтегроване середовище для збору й аналізу інформації. Це сприяє покращенню якості життя, оптимізації процесів та підвищенню ефективності. [1]

Технологія IoT базується на кількох ключових принципах:

- підключення до мережі: кожен пристрій IoT має можливість підключення до локальної мережі або Інтернету, що забезпечує доступ до пристрою з будь-якої точки;
- збір даних: датчики та сенсори, що інтегровані у пристрої, постійно фіксують та передають інформацію про стан об'єкта чи параметри навколишнього середовища;
- аналіз даних: зібрані дані обробляються для аналізу поточного стану системи та виявлення можливих відхилень чи трендів;

- автоматизація: IoT пристрої можуть самостійно виконувати певні операції, що дозволяє автоматизувати значну частину робочих процесів.

IoT має широку варіацію застосувань у різних галузях. Нижче наведено опис деяких з основних прикладів застосування IoT систем:

- Розумний дім: в таких системах зазвичай використовують розумні розетки, лампи, різні датчики, або пристрої які можуть підключатись до Wi-Fi. Керування відбувається за допомогою телефону і дозволяє підвищити комфорт та автоматизувати деякі повсякденні справи;
- Промисловість: як правило використовується для збору та аналізу даних з виробничих ліній та обладнання, що дозволяє раніше виявити потенційні проблеми, поліпшити якість продукції та запобігає аваріям;
- Енергетика: IoT системи часто використовують для впровадження інтелектуального управління енергетичними системами, такими як вітрові та сонячні електростанції, або для контролю за показниками в системі;
- Охорона праці та безпека: використовується для моніторингу належного дотримання правил безпеки. Наприклад датчики вимірюють якість повітря, рівень шуму, температуру та інші параметри;
- Медицина: така система відкриває можливості для впровадження віддаленого моніторингу стану здоров’я та використання інтелектуальних медичних пристроїв. Завдяки датчикам можна збирати інформацію про ключові показники організму пацієнтів, що дає змогу лікарям оперативно реагувати на будь-які зміни;
- Транспорт і логістика: системи IoT можуть забезпечити більш оптимальні системи пасажирських і вантажних перевезень, на основі отриманих та оброблених даних. Дистанційний моніторинг ситуації

на дорогах, розумні системи паркування можуть допомогти зменшити затори та забезпечити комфорт пасажирів; [2]

Рис. 1.1 – Основні області застосування IoT

Оскільки в даній роботі ми розглядаємо застосування IoT систем для виробництва, то варто окремо зазначити переваги, та потенційні недоліки. До переваг таких систем можна віднести наступні пункти:

1. Продуктивності: IoT надає можливість оптимізувати виробничі процеси, що веде до підвищення продуктивності для виробництва продукції. Це досягається завдяки безперервному моніторингу обладнання та процесів, а також збору даних у реальному часі. Аналіз

зібраних даних дає змогу виявити вузькі місця у виробничих ланцюжках, що зі свого боку сприяє ефективному розподілу та оптимізації ресурсів. [4]

2. Зниження витрат: Системи IoT дозволять виявляти потенційні несправності, перш ніж вони призведуть до якихось проблем. Це дозволяє проводити більш дешеве і просте обслуговування.
3. Безпека: якщо виникне надзвичайних ситуацій, наприклад пожежа або витік шкідливих речовин, то датчики IoT можуть автоматично активувати системи оповіщення, відкрити воду для тушіння.

Проте варто зазначити і потенційні виклики з запровадженням такої системи:

1. Сумісність: Часто IoT пристрої використовують тільки один протокол зв’язку, на жаль це не завжди дозволяє використовувати пристрої від різних виробників в одній системі. Проте це легко можна виправити, якщо заздалегідь правильно розробити план для запровадження такої системи, де буде враховуватись сумісність між пристроями.
2. Безпека: деякі з пристроїв вразливі до різних типів атак, таких як DDOS-атаки, експлойт програмного забезпечення, атака посередника і т.д., або ж у деяких бюджетних виробників пристроїв немає необхідного досвіду та стимулу для постійного оновлення прошивки. Тому при виборі протоколу та пристрою важливо приділити увагу питанню безпеки та шифрування даних. [5]

Загалом можна дійти висновку, що ця технологія завдяки своєму потенціалу змінювати та покращувати різні аспекти нашого життя є вже майже незамінною. І хоча технологія має свої складнощі та недоліки, її використання стає все більш поширеним у різних сферах, що в свою чергу сприяє розвитку самої технології.

1.2 Архітектура IoT

Архітектура IoT - це складна та багатогранна концепція, яка формує основу систем IoT, в ній описується, як пристрої, програми та технології IoT взаємодіють один з одним, щоб забезпечити бажану функціональність. Архітектура IoT не є універсальною моделлю, вона змінюється залежно від конкретних вимог відповідної системи. Це забезпечує системний підхід до інтеграції різних елементів IoT, що забезпечує безперебійне спілкування та обмін даними між пристроями. Розуміння архітектури IoT має важливе значення, оскільки це впливає на функціональність, продуктивність, масштабованість і безпеку системи. Зазвичай вона складається з чотирьох рівнів: рівня пристроїв, мережевого рівня, рівня обробки даних (керування) та рівня додатків (рисунок 1.2).

Рис. 1.2 – Основні рівні архітектури системи IoT

Кожен рівень має певну роль і містить певні компоненти, які сприяють загальній функціональності системи IoT. Архітектура забезпечує основу для проектування та впровадження систем IoT, керуючи вибором та інтеграцією компонентів на кожному рівні. Детальний опис цих шарів наведено нижче.

Рівень пристроїв

Рівень пристроїв, є найнижчим рівнем в архітектурі IoT. Цей рівень відповідає за генерацію та збір даних за допомогою датчиків, а також за вплив на зміни в навколишньому середовищі або в системах за допомогою виконавчих механізмів. Доступно багато різних типів датчиків, датчики відстані, тиску, газу, радіаційні датчики, температури та вологості, датчики потоку, контактні датчики і т.д. Загалом їх можна класифікувати як цифрові та аналогові. Цифрові датчики повідомляють дані в дискретних значеннях, тоді як аналогові датчики повідомляють дані в безперервних значеннях. Тип використовуваного датчика визначатиме, чи використовуються цифрові чи аналогові вхідні/вихідні контакти на платі мікроконтролера для підключення датчика. Важливо вибрати відповідний тип датчика для даних, які збираються, виходячи з необхідної точності та роздільної здатності, а також діапазону вимірюваних значень. Ще одним типом пристроїв які використовують на цьому рівні є виконавчі механізми. [7]

Такі механізми перетворюють електричний сигнал у певну фізичну дію. Їх можна класифікувати за джерелом енергії руху, наприклад, електрична, гідравлічна, пневматична, теплова чи магнітна, або ж за типом виробленого руху, як наприклад, обертальний, коливальний, лінійний. Виконавчі механізми використовуються для перетворення командних інструкцій у точні дії, викликаючи зміну стану системи, навколишнього середовища або явища.

Мережевий рівень

Мережевий рівень є другим рівнем в архітектурі IoT. Даний рівень відповідний за передачу даних від рівня пристроїв до рівня керування. Він включає різні типи мереж, наприклад локальні мережі, глобальні мережі та Інтернет. Мережевий рівень забезпечує надійну та безпечну передачу даних, що є критично важливим для функціонування системи IoT. Мережевий рівень використовує різні протоколи зв'язку для передачі даних, такі як Bluetooth, Zigbee, Z-Wave, Wi-Fi та інші. Вибір протоколу зв'язку визначається специфічними потребами IoT-системи, такими як дальність передачі, можливість масштабування, рівень енергоспоживання та швидкість обміну даними. У деяких випадках мережевий рівень може передбачати використання шлюзів і маршрутизаторів, які виконують функцію посередників між пристроями та центральним вузлом.В залежності від моделі вони можуть мати додаткові функції безпеки, наприклад автентифікація та шифрування для захисту від несанкціонованого доступу. У мережевому рівні можна комбінувати різні технології та протоколи для керування потоком повідомлень і оптимізації пропускної здатності, енергоспоживання та використання ресурсів. [8]

Рівень обробки даних

Рівень обробки даних це третій рівень в архітектурі IoT який має програмні та апаратні компоненти, які використовуються для збору та аналізу даних з IoT пристроїв, отриманих через мережевий рівень. Рівень обробки даних складається з різноманітних технологій та інструментів, як сервери, бази даних і хмарні платформи, які зберігають і обробляють дані для подальшого використання. Цей рівень також керує пристроями в системі IoT, забезпечуючи їх належне функціонування та оновлюючи їх програмне забезпечення за потреби. Рівень обробки даних забезпечує потужність обробки, необхідну для функціонування системи, а також зазвичай є центральним вузлом системи. У якості компонента який буде відповідати за цей рівень можна обрати мікрокомп'ютер, або готове

рішення на його основі. Якщо ж не хочется тримати сервер в себе, то можна використати хмарну платформу.

Рівень додатків

Рівень додатків є останнім рівнем в архітектурі IoT. На цьому рівні оброблені дані використовуються для надання корисних послуг кінцевим користувачам. Додатки можуть варіюватися від простих інструментів візуалізації даних до складних систем прийняття рішень. Рівень додатків виконує роль інтерфейсу між IoT-системою та користувачами, забезпечуючи їм доступ до корисної інформації, отриманої на основі даних, зібраних IoT-пристроями. Він охоплює широкий спектр програмного забезпечення, включаючи мобільні додатки, веб-портали та інші користувацькі інтерфейси, призначені для взаємодії з IoT-інфраструктурою. Обмін даними на цьому рівні може здійснюватися за допомогою таких протоколів, як HTTP, WebSocket, MQTT, CoAP і AMQP. Ці протоколи можна використовувати для реалізації різних форматів зв’язку, таких як архітектури клієнт/сервер і механізми підписки. Це може бути корисно під час роботи з великими обсягами даних або ввімкнення зв’язку між пристроями в реальному часі. [7]

Як результат, архітектура IoT є багаторівневою структурою, яка зазвичай включає рівень пристроїв, мережевий рівень, рівень керування та рівень додатків. Кожен рівень має певну роль і містить певні компоненти, які сприяють загальній функціональності системи IoT. Архітектура забезпечує основу для проектування та впровадження систем IoT, керуючи вибором та інтеграцією компонентів на кожному рівні.

1.3 Бездротові протоколи зв’язку для мережі IoT

Бездротові мережі не є новим винаходом у сфері технологій, але час від часу зазнавали прогресу та інновацій, щоб подолати зростаючі виклики, пов’язані зі зростанням кількості пристроїв/систем зв’язку. Протоколи

служать спільною мовою для взаємодії мережевих об’єктів, таких як сервери, шлюзи, маршрутизатори, програми та підключені пристрої IoT. Протокол визначає набір правил, яких повинні дотримуватися обидві сторони для ефективного спілкування. Ці правила визначають характер їх взаємодії, значення та атрибути, що передаються, методи отримання та обробки даних, застосовувані заходи безпеки тощо. Вибір протоколу визначає складність системи та допомагає визначити пріоритетність характеристик, таких як швидкість, чіткість, енергозбереження та безпека.

Для різних проектів і випадків потрібні різні пристрої та протоколи. Наприклад, система IoT з медичними датчиками на машинах швидкої допомоги, які передають дані пацієнтів до лікарень, має надавати пріоритет швидкості та безпеки, вимагаючи потужніших і безпечних протоколів. Система ж IoT для керування пристроями на виробництві має віддавати перевагу енергозбереженню, масштабованості та безпеці. Ось основні бездротові протоколи, які використовуються для розгортання мережі IoT системи:

BLE

BLE, як і інші стандарти зв’язку з низьким енергоспоживанням і пропускною здатністю, розроблений для передачі даних у невеликих пакетах, зводячи до мінімуму час роботи пристрою на батареї в активному режимі. Його головна відмінність від класичного Bluetooth полягає в тому, що пристрої Bluetooth Low Energy встановлюють з’єднання лише за необхідності передачі чи отримання даних.[6]

Для технології, орієнтованої на низьке енергоспоживання, Bluetooth Low Energy має досить вражаючі швидкості передачі даних, до 2 Мбіт/с для версії 5. Завдяки підтримці вузлів сну (пристроїв, які проводять більшу частину свого часу в неактивному стані, періодично «прокидаючись» на деякий час). короткий час лише для швидкого виконання свого завдання, а потім так само швидко повертається до сну), BLE пропонує гідний, хоча й

не найкращий у галузі, час автономної роботи, що дозволяє певним датчикам і перемикачам працювати понад рік маленькі монетні батарейки.

Головним недоліком є те, що Bluetooth використовує діапазон 2,4 ГГц, що й багато інших протоколів, включаючи Wi-Fi та ZigBee. Хоча Bluetooth оснащений деякими інструментами для боротьби з перешкодами (наприклад, технологією адаптивної зміни частоти, яка забезпечує можливість динамічного перемикання між 40 доступними каналами під час передачі даних, уникаючи найгучніших і найбільш завантажених), використання частоти 2,4 ГГц смуга є незаперечним недоліком.

Wi-Fi

Є одним із найпоширеніших бездротових протоколів для передачі даних. Технічно Wi-Fi має зіркову топологію, що означає, що всі його вузли підключені безпосередньо до центрального елемента - бездротового маршрутизатора.

Основною перевагою цього протоколу є здатність забезпечувати обмін великими обсягами даних на невеликих відстанях, і Wi-Fi успішно виконує цю задачу. Такі характеристики, як радіус дії чи швидкість передачі даних, варіюються залежно від версії стандарту 802.11. У більшості випадків звичайного бездротового маршрутизатора вистачає, щоб забезпечити покриття мережі на території площею 100-150 метрів. Старіші версії стандарту 802.11 мають обмеження швидкості в 54 Мбіт/с, тоді як новіші здатні передавати дані зі швидкістю в сотні мегабіт на секунду та навіть більше. [9]

Зазначені вище переваги Wi-Fi добре відомі. Завдяки їм ця технологія стала невід'ємною частиною нашого повсякденного життя. Проте, коли йдеться про високу швидкість передачі даних і здатність працювати з великими обсягами інформації, це не є пріоритетними параметрами для IoT систем. Підтримка Wi-Fi в таких системах пов'язана зі значними витратами енергії. Як стандарт високошвидкісного зв’язку, Wi-Fi є надто енергоємним

для IoT. Хоча енергоспоживання не викликає проблем у пристроїв, що підключені до постійного джерела живлення, воно стає важливим для пристроїв, які повинні мати змогу працювати автономно.

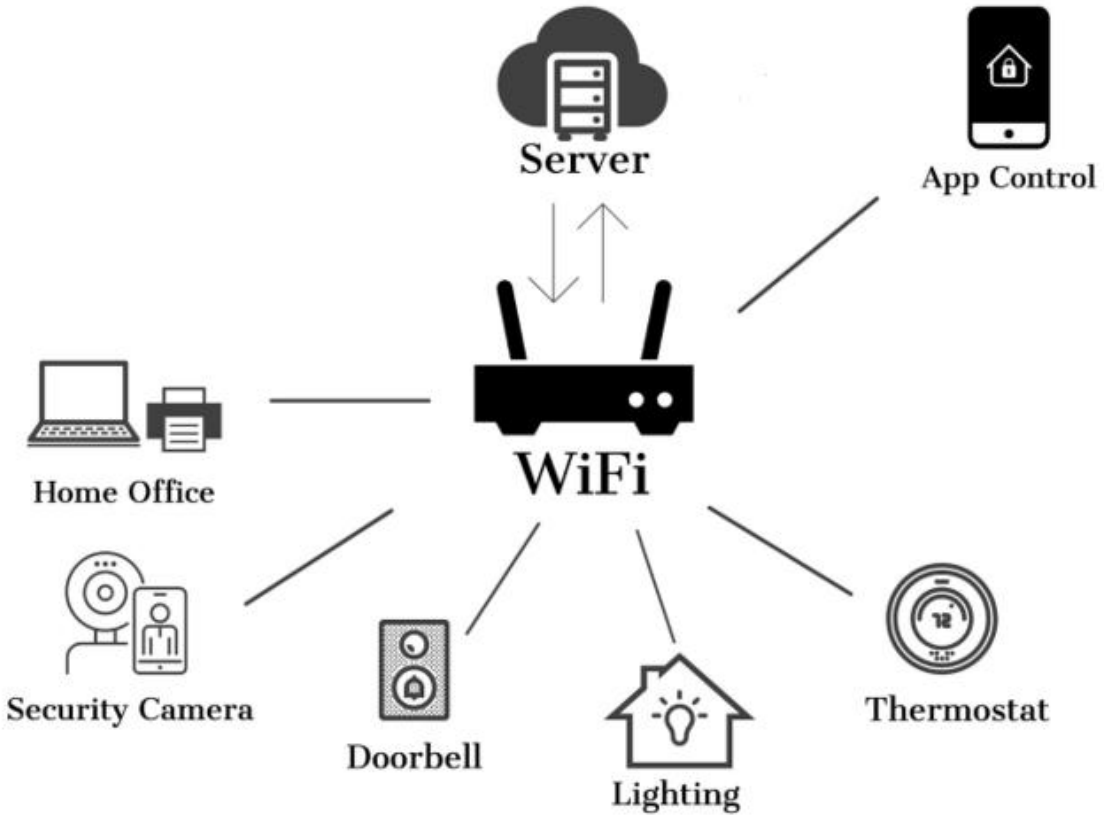

Рис. 1.3 - Топологія зірка на прикладі Wi-Fi мережі

Ще одне серйозне обмеження Wi-Fi пов'язане з його мережевою топологією. Центральний маршрутизатор, через який проходить весь трафік, має обмеження щодо кількості підключених пристроїв. Наприклад, навіть із якісним маршрутизатором можна підключити до 40 пристроїв, але для забезпечення стабільної швидкості рекомендується обмежуватися сімома. Вирішення цієї проблеми можливе шляхом придбання більш потужного маршрутизатора, однак такі пристрої коштують дорого, і їхній радіус дії все одно може виявитися недостатнім для покриття великої площі.

Z-Wave

Z-Wave використовує сітчасту топологію мережі. Протокол розроблений таким чином, що вузли мережі, виконуючи роль повторювачів, можуть передавати повідомлення через себе далі, доки воно не досягне кінцевого адресата. Такий підхід не тільки дозволяє істотно розширити зону дії бездротової мережі, але і підвищує її надійність. Кожна логічна мережа Z-Wave може підтримувати до 232 пристроїв. Якщо необхідно підключити більше пристроїв, можна об'єднати мережі. Кожна мережа Z-Wave має головний контролер, який відповідає за додавання в мережу нових пристроїв і видалення старих, створення карт маршрутизації, забезпечення безпечних з'єднань, надання можливості створювати сценарії автоматизації та інші функції для організації та моніторингу мережі. Основна особливість в тому, що Z-Wave працює в неліцензійній частоті діапазону 800-900 МГц, виділеної для пристроїв ближнього радіусу дії. Відмінною особливістю цих частот є здатність краще долати різноманітні перешкоди, включаючи стелі та стіни. [12]

Для масштабування допускається використовувати до 4 проміжних вузлів для транзитної передачі даних. Враховуючи, що дальність у зоні прямої видимості сучасних модулів Z-Wave становить близько 100 метрів, а нове покоління пристроїв збільшує це значення може бути більше за 800 метрів на відкритій місцевості, кінцева дальність передачі сигналу в одній мережі Z-Wave досить висока. достатньо для виконання більшості завдань.

Ще однією перевагою Z-Wave є безпека. Технологія Z-Wave використовує новий стандарт безпеки, відомий як Security 2 (S2). Він став обов’язковим для сертифікації всіх смарт-пристроїв Z-Wave після 2 квітня 2017 року та встановлює нові процедури підключення нових пристроїв до мережі за допомогою PIN-кодів або QR-кодів.

Також на сьогоднішній день існує велика кількість пристроїв від різних компаній. Крім того, немає обмежень щодо сумісності різних пристроїв. Тестування, проведене альянсом, гарантує, що будь-які

сертифіковані пристрої, незалежно від їх виробника, добре уживатимуться один з одним. Таким чином, гарантована сумісність є найсильнішою конкурентною перевагою Z-Wave. І це при тому, що протокол також забезпечує зворотну сумісність з усіма попередніми версіями, що є ще однією вагомою перевагою.

Основний недолік Z-Wave випливає з його переваги — використання частотного діапазону. Обрання низькочастотного, стабільного та менш завантаженого діапазону для роботи на коротких відстанях виявилося продуманим і ефективним рішенням, яке допомогло уникнути проблем із перешкодами на перевантажених частотах. Однак у різних країнах для пристроїв ближнього радіусу дії виділяються різні частоти. Наприклад, у Європі (зокрема в Україні), Китаї та деяких країнах Азії використовується частота 868,42 МГц. У США та Мексиці ці частоти зайняті GSM, тому Z-Wave працює на частоті 908,42 МГц. Через це пристрої, виготовлені для США, несумісні з європейськими стандартами.

Zigbee

ZigBee використовує маршрутизацію за пунктом призначення, що дозволяє організовувати сітчасту мережу з трьома типами пристроїв: координатором, маршрутизатором і кінцевими пристроями. Маршрутизатори, які завжди активні, потребують постійного живлення. Вони забезпечують підключення до 32 кінцевих пристроїв, тому їх розташування має бути оптимальним, а кількість – достатньою для підтримки стабільної роботи мережі. Такий підхід забезпечує самовідновлення мережі та ефективне перенаправлення даних у разі виходу з ладу будь-якого вузла, що робить ZigBee надійним рішенням для створення сітчастих мереж, подібно до Z-Wave. [13]

Однією з переваг ZigBee є його хороша масштабованість. Протокол здатен підтримувати до 65 000 вузлів, що теоретично забезпечує значне покриття, навіть попри обмежений діапазон окремих модулів (10–20 м у

приміщенні). Однак, такі цифри на практиці можуть виявитися перебільшеними. Наприклад, у мережах із кількістю вузлів, що сягає кількох тисяч, часто виникають проблеми з безперебійною роботою навіть у лабораторних умовах. Затримки у передачі даних спостерігаються вже при розгортанні кількох сотень пристроїв. Це пояснюється тим, що ZigBee використовує перевантажений частотний діапазон 2,4 ГГц, а максимальна швидкість передачі даних складає лише 250 Кбіт/с.

Енергоспоживання є однією з сильних сторін Zigbee. Хоча за цим показником він поступається Z-Wave, окремі пристрої Zigbee можуть працювати до 2 років без заміни батареї, що робить цей протокол привабливим для багатьох застосувань.

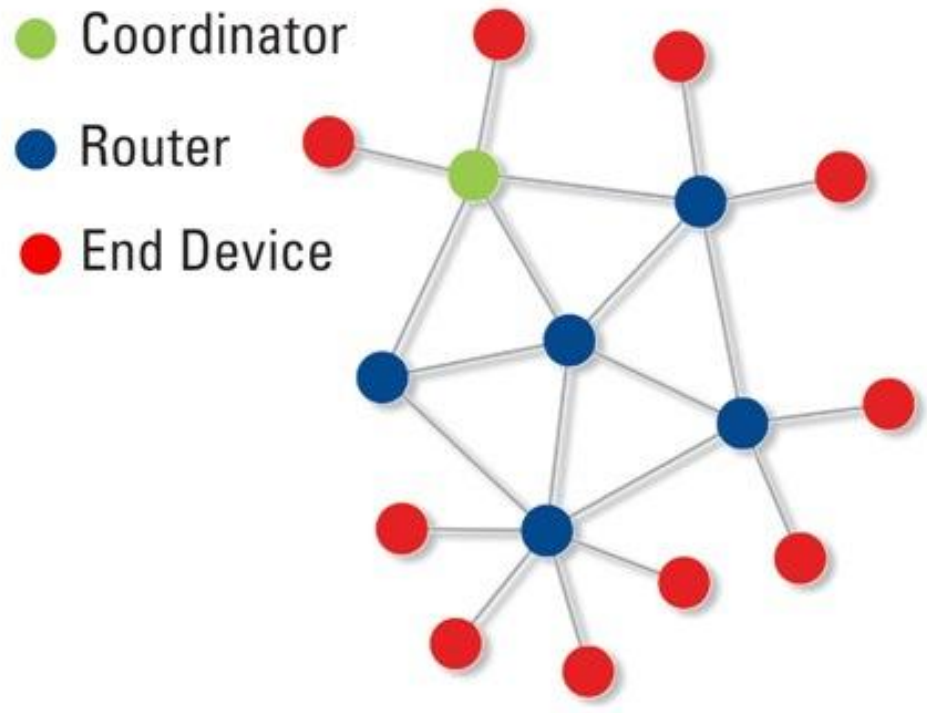

Рис. 1.4 - Топологія сітка на прикладі Zigbee мережі

Переходячи до недоліків протоколу, варто зазначити один із ключових — використання частотного діапазону 2,4 ГГц. Zigbee демонструє низьку ефективність у середовищах із сильними перешкодами від інших пристроїв у зоні покриття. Як одноканальне рішення, він не

завжди може ефективно протидіяти завадам у перевантаженому діапазоні 2,4 ГГц, який активно використовується такими технологіями, як Wi-Fi і Bluetooth. З часом ситуація лише буде погіршуватись, оскільки перевантаженість цього частотного спектра зростає щороку.

Що стосується безпеки, Zigbee пропонує широкий набір розширених заходів для захисту даних, якими обмінюються розумні пристрої. Використання 128-бітного алгоритму AES для шифрування та аутентифікації, а також трьох типів ключів для управління безпекою забезпечує високий рівень захисту. Однак періодично з’являються повідомлення про виявлення вразливостей у пристроях, що підтримують Zigbee. Тому для забезпечення належного рівня захисту важливо максимально використовувати всі доступні засоби безпеки.

LoRaWan

Як відомо, LoRaWAN є широкосмуговим протоколом, призначеним для IoT-систем. Його архітектура побудована за топологією "зірка-зірка", де шлюзи виконують роль посередників, передаючи повідомлення між кінцевими пристроями та центральним сервером. Бездротовий зв'язок базується на технології Long Range (велика дальність), яка використовується на фізичному рівні LoRa, забезпечуючи одноточкове з'єднання між кінцевим пристроєм і одним або кількома шлюзами. Протокол підтримує двосторонній зв'язок, а також групову адресацію, що сприяє ефективному використанню спектра, наприклад, для оновлення прошивки "по повітрю" чи інших масових розсилок.

Оскільки мережа дальнього радіусу дії (десь до 15 кілометрів), то шлюз повинен мати можливість для отримання повідомлень від великої кількості вузлів, що є вагомою перевагою цього протоколу. Така висока пропускна здатність розподіляється за допомогою адаптивної швидкості передачі даних і багатоканального передавача таким чином, щоб можна було отримувати одночасні повідомлення. Звісно варто також зазначити

використання МГц частот, що дозволяє бути більш стійкою до перешкод, та зменшує вплив перешкод від інших пристроїв. [14]

Серед недоліків варто виділити лише швидкість, яка буде до 50 Кбіт/с, що є найнижчим показником серед усіх розглянутих протоколів. На жаль така швидкість не дозволить використовувати LoRaWan у тих випадках, коли швидкість передачі буде важливим параметром. Так наприклад не вийде застосувати цей протокол для програм, де потребується часта передача великого обсягу даних.

З огляду на представлену інформацію, у таблиці нижче представлено порівняння основних параметрів для кожного протоколу:

Таблиця 1.1 – Порівняння основних характеристик мережевих протоколів

	Частота	Відстань	Споживання енергії	Швидкість	Топологія	Безпека
BLE	2.4 ГГц	до 10 м	Низьке	до 2 Мбіт/с	Зірка	AES-128
Wi-Fi	2.4 ГГц / 5 ГГц	До 50 метрів в приміщенні, до 150 метрів на вулиці	Високе	54 - 600 Мбіт/с	Зірка	WPA/WPA2
Z-Wave	800 - 900 МГц	До 100 метрів у приміщенні,	Низьке	100 Кбіт/с	Сітка	AES-128 + S2
Zigbee	2.4 ГГц	До 70 метрів у приміщенні, до 200 метрів на вулиці	Низьке	250 Кбіт/с	Сітка	AES-128
LoRaWan	433 МГц - 915 МГц	До 15 км	Низьке	до 50 Кбіт/с	Зірка	AES-128 / AES-256

Окремо варто розглянути протоколи, які частіше за все використовують на рівні додатків. На цьому рівні протоколи виконують найбільш інтенсивні процеси, відповідальні за обмін даними а повідомленнями між кінцевими пристроями та програмними додатками. До основних протоколів відносяться:

- MQTT – це простий легкий протокол обміну повідомленнями, який використовується для встановлення зв’язку між кількома пристроями. Це протокол на основі TCP, що базується на моделі публікації-підписки. Цей протокол зв'язку підходить для передачі даних між пристроями з обмеженими ресурсами, які мають низьку пропускну здатність і низькі вимоги до потужності;
- HTTP – це широко використовуваний протокол для передачі веб-сторінок і даних через Інтернет. Ця технологія проста в застосуванні та використовує існуючу веб-інфраструктуру. HTTP зазвичай розгортається в системах, які потребують інтеграції з веб-сервісами;
- CoAP – це протокол рівня додатків, який був представлений в 2014 році. CoAP розроблений для обмеженого середовища. Це веб-протокол, який нагадує HTTP. Він також базується на моделі запит-відповідь. Дані з одного ресурсу на інший передаються у формі пакетів повідомлень CoAP;
- AMQP - це відкритий стандартний протокол, який використовується для обміну повідомленнями та керування чергами. Він забезпечує надійну та впорядковану доставку повідомлень, підтримує різноманітні шаблони зв’язку та пропонує високий ступінь взаємодії. AMQP широко впроваджується у фінансовому секторі та корпоративному спілкуванні; [15]

Отже технологія IoT може використовувати велику кількість різних протоколів. В даному розділі були описані найбільш популярні з них, які

можна потенційно використовувати для розробки систем моніторингу та керування IoT пристроями на виробництві.

1.4 Огляд існуючих хмарних систем

В наш час для ефективної роботи систем IoT часто використовуються спеціалізовані платформи, що дозволяють керувати пристроями, зберігати та аналізувати дані, а також забезпечують інтеграцію з іншими системами. У цьому розділі буде розглянуто основні типи рішень, які можуть використовуватись для моніторингу та керування IoT системами на виробництві. Більшість рішень використовують хмарні платформи, тому саме ці варіанти варто найбільш детально розглянути. Вони забезпечують масштабованість, гнучкість та суттєво спрощують інтеграцію IoT систем. Хмарні сервіси дозволяють зберігати величезні обсяги даних, обробляти їх у реальному часі, а також забезпечувати доступ до інформації з будь-якої точки світу. Однією з найбільш відомих хмарних платформ є AWS IoT Core.

AWS IoT Core - це хмарний сервіс від Amazon, який забезпечує функціональність для легкого та безпечного підключення пристроїв IoT до хмари AWS. AWS IoT забезпечує безпечний двонаправлений зв’язок між підключеними до Інтернету IoT пристроями та хмарою AWS. [16]

Другим подібним сервісом є Microsoft Azure IoT, розроблений корпорацією Майкрософт. Він є повністю керованим сервісом, який дозволяє користувачам керувати пристроями IoT і контролювати їх. Крім того, Azure IoT забезпечує надійний і безпечний двонаправлений зв’язок між пристроями IoT і його хмарними службами. Це дозволяє розробникам отримувати повідомлення та надсилати повідомлення на пристрої IoT, діючи як центральний центр повідомлень для спілкування. Це також може допомогти використовувати дані, отримані з пристроїв Інтернету речей, перетворюючи дані Інтернету речей у практичну інформацію.

Третім популярним рішенням є Kaa IoT. Гнучка та масштабована IoT платформа для створення рішень IoT і керування підключеними пристроями. Архітектура системи базується на мікросервісному підході та використовує його в повній мірі. Кожен мікросервіс Kaa є незалежним будівельним блоком. Така архітектура дозволяє комбінувати ці блоки для створення узгоджених рішень. У масштабі всієї платформи мікросервіси Kaa виглядає як купа чорних ящиків, які виконують свою роботу. Це означає, що архітектура окремого мікросервісу не має значення для архітектури всієї платформи. [18]

Висновки до розділу 1

Аналіз архітектури IoT дозволив визначити основні рівні, що забезпечують ефективну взаємодію пристроїв у мережі. Аналіз бездротових протоколів зв’язку є однією з найважливіших технологій для побудови мереж IoT, тому розуміння їх переваг та недоліків дозводить краще підібрати відповідний протокол. Розгляд існуючих рішень показав можливість інтеграції IoT із хмарними платформами, що є потенційним альтернативним варіантом побудови системи з перенесенням керування з рівня обробки даних у хмару.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДЛЯ СИСТЕМИ МОНІТОРИНГУ

У рамках цього розділу буде описано вимоги, апаратну частину та структурну схему для системи моніторингу та керування IoT пристроїв на виробництві. Також буде більш детально розглянуто хмарні платформи, які можуть використовуватись як окреме рішення, так і в парі з персональною платформою.

2.1 Вимоги до IoT системи

З огляду на інформацію з першого розділу постає питання в тому, як саме обрати підхід для впровадження системи моніторингу та керування IoT пристроями на виробництві. Система точно матиме чотири рівні, але їх компоненти будуть відрізнятись в залежності від протоколу, та потреб виробництва. Для того, щоб більш детально розуміти як створити таку систему, необхідно перш за все визначити вимоги до IoT систем на виробництві.

1. Енергоспоживання

Обмеження енергоспоживання перш за все варто запровадити на рівні пристроїв. Це пов’язане з тим, що зазвичай вони працюють від джерел живлення з обмеженим запасом енергії, зазвичай батареї. Чим рідше виникає потреба у заміні або зарядці батареї, тим меншими будуть витрати на обслуговування. На мережевому рівні обираються оптимальні маршрути передачі даних від сенсорного вузла до координатора, враховуючи кількість проміжних вузлів, енергетичні витрати та доступний запас енергії. Крім мережевих протоколів, на енергоспоживання впливають конструктивні особливості вузлів, програмне забезпечення, механізми захисту та характер робочих додатків.

2. Масштабованість

Система моніторингу та керування IoT пристроями на виробництві має бути готовою до розширення, при цьому при мінімальних змінах у загальній структурі системи. Для запровадження масштабованості можна використовують протоколи, які мають топологію сітки, та мають можливість використовувати велику кількість пристроїв.

3. Відмовостійкість

Важливо попередньо розглянути, як система буде працювати за непередбачуваних обставин. Відмовостійкість стосується пристроїв, та програмного забезпечення, яке використовується в системі. На рівні обробки даних з певних причин може виникнути проблема з виконанням програми. Для мінімізації таких проблем використовується поділ програмного забезпечення на окремі частини. Частіше за все це робиться через Docker, в такому випадку кожна програма запускається в окремому контейнері. Якщо ж один із контейнерів зупиниться, то інші продовжать працювати. Це звісно певним чином вплине на систему, але вона буде частково працювати та буде легше виявити і виправити помилку.

4. Кібербезпека

Під час розробки системи необхідно враховувати можливі загрози, які можуть виникнути, і після завершення створення архітектури передбачити відповідні засоби для її захисту. Важливо розробити стратегію безпеки ще на початкових етапах, оскільки розуміння методів, якими зловмисники можуть скомпрометувати систему, дозволяє заздалегідь усунути потенційні ризики.

5. Зворотна сумісність

З часом протоколи оновлюються, або створюються нові версії, що може призвести до проблем сумісності з пристроями. Це призводить до того, що або не треба оновлюватись, або треба міняти більшість пристроїв. Перший варіант не є підходящим, бо нові версії як правило мають оновлені протоколи захисту, що в наш час є дуже важливим. Другий варіант призведе

до значних витрат, чого за можливості варто уникати. Тому зворотна сумісність є доволі важливою вимогою, яка дозволить полегшити підтримку IoT системи у майбутньому.

Як бачимо розробка системи моніторингу та керування IoT пристроями потребує комплексного підходу, що дозволяє попередити потенційні ризики. Бажано дотримуватись всіх вимог, щоб створити ефективну, надійну та безпечну систему.

2.2 Вибір мінікомп'ютера

В результаті аналізу розділу 1.3 можна побачити певну закономірність, яка полягає в тому, що кожен протокол використовує контролер. Деякі також використовують маршрутизатори для розширення мережі. Оскільки таких пристроїв безліч, то їх вибір має бути зроблений виходячи з потреб протоколу та кінцевих пристроїв.

Варто більш детально розглянути варіант, коли система моніторингу та керування IoT пристроями розробляється самостійно з використанням готового програмного забезпечення. В таких випадках необхідно обрати пристрій, який буде виконувати роль сервера на рівні обробки даних. Зазвичай обирають мінікомп'ютери, основними критеріями для вибору можна виділити продуктивність, об’єм оперативної пам’яті, сумісність із відкритим програмним забезпеченням, вбудована пам’ять та підтримка microSD, енергоефективність та ціну. В наш час на ринку існують різні пристрої які класифікуються як мінікомп'ютер. Найбільш поширеними серед них є Orange Pi 5, Raspberry Pi 5, Banana Pi M5, NanoPi NEO4, Odroid-C4. Розглянемо трохи детальніше кожний з цих пристроїв, щоб визначити який найбільше відповідає вказаним вимогам.

Orange Pi 5 - це потужний одноплатний комп'ютер, оснащений 4 ГБ оперативної пам'яті, що пропонує широкі можливості для розробки та впровадження різноманітних проектів. Завдяки продуктивному процессору Rockchip RK3588 та підтримці високоякісної графіки, цей мінікомп'ютер ідеально підходить для застосування в галузі автоматизації та розробки програмного забезпечення. Основн характеристики: 8-ядерний 64-бітний процесор; 4 ГБ оперативної пам'яті (є версії з 8 та 16 ГБ); підтримка портів та інтерфейсів USB 3.0, HDMI 2.1, Ethernet; сумісність із різними операційними системами для гнучкої розробки; слот для MicroSD картки; джерело живлення - Type-C, 5В, 4А.

Raspberry Pi 5 — це нове покоління мінікомп’ютерів серії Raspberry, яке забезпечує вдвічі вищу продуктивність, а також розширений набір і кількість портів вводу-виводу. Пристрій оснащений новим ARM-процесором Cortex-A76 із частотою 2.4 ГГц і 4 ГБ оперативної пам’яті LPDDR4 (доступні також моделі з 8 та 16 ГБ). Оновлений південний міст підвищує пропускну здатність USB3, а новий графічний процесор VideoCore VII працює на частоті 800 МГц (проти 500 МГц у VideoCore VI у Raspberry Pi 4). Серед приємних нововведень — кнопка ввімкнення, якої не було в попередніх версіях. [19]

Banana Pi M5 - це одноплатний комп’ютер нового покоління, який використовує чотириядерний процесор Amlogic S905X3 Cortex-A55 (2,0xxGHz). Графічний процесор Mali-G31 MP2 із 4 процесорами (650 МГц). підтримка 4 ГБ LPDDR4 і 16 ГБ флеш-пам’яті eMMC. він має 4 порти USB 3.0, порт 1GbE LAN, ІЧ-приймач, аудіороз’єм, 1 вихід HDMI і блок живлення USB type-c.

Odroid-C4 – це одноплатний комп’ютер нового покоління, який є більш енергоефективним і швидшим, ніж Odroid-C2, який був представлений понад чотири роки тому як перший у світі доступний 64-розрядний комп’ютер ARM. Основний процесор Odroid-C4 складається з

чотирьохядерного кластера Cortex-A55 з графічним процесором Mali-G31 нового покоління. Ядра A55 працюють на частоті 2,0 ГГц без терморегулювання за допомогою стандартного радіатора, що забезпечує надійний і тихий комп’ютер.

NanoPi NEO4 - це одноплатний комп'ютер від FriendlyELEC, який працює на базі потужного Rockchip RK3399 з 2-ма ядрами Cortex-A72 і 4-ма ядрами Cortex-A53. Плата має 1GB RAM, Gigabit Ethernet, вбудовані Wi-Fi та BT 4.0. За обробку графіки відповідає графічний процесор Mali-T864 з підтримкою OpenGL, OpenCL, DX11, та 4K 10-біт H265/H264 декодером. Плата оснащена роз'ємом для підключення камери до 13MP, слотом для підключення eMMC-модулів, слотом для microSD карт, портами USB 3.0, USB 2.0 та USB Type-C як OTG та порт живлення. На сьогоднішній день NanoPi NEO4 – найменша плата на основі RK3399 на ринку.

Таблиця 2.1 – Порівняльна характеристика мінікомп'ютерів

Модель	Процессор	Ядра	Частота	RAM	Пам'ять	Ціна
Orange Pi 5	RK3588	8	2.4 ГГц	4 ГБ	microSD	130-145$
Raspberry Pi 5	ARM Cortex-A76	4	2.4 ГГц	4 ГБ	microSD	80-90$
Banana Pi M5	Amlogic S905X3	4	1.5 ГГц	4 ГБ	16 ГБ eMMC, microSD	80-100$
Odroid-C4	Amlogic S905X3	4	2.0 ГГц	4 ГБ	4 ГБ eMMC, microSD	90-100$
NanoPi NEO4	Rockchip RK3399	6	2.0 ГГц	1 ГБ	eMMC, microSD	60-70$

У результаті порівняння характеристик різних мінікомп'ютерів найбільш перспективними виявилися Orange Pi 5 і Raspberry Pi 5. Перевага Orange Pi 5 полягає у використанні 8-ядерного процесору. З практичної точки зору Raspberry Pi 5 більш ніж достатньо для стандартних проектів,

помірних обчислювальних завдань, та пропонує хорошу продуктивність за більш доступною ціною. Однак Orange Pi 5, краще пристосований для важчих робочих навантажень.

2.3 Вибір протоколів рівня додатків

Окремо варто розглянути протоколи рівня додатків, оскільки вони відіграють не меншу роль, ніж протоколи для пристроїв. Основна їх задача це забезпечити швидку і ефективну взаємодію з іншим серверами, додатками, хмарами і т.д. У розділі 1.3 були описані основні протоколи цього рівня, проте детально будуть розглянуті тільки MQTT та HTTP.

Причина в тому, що це найбільш популярні протоколи, вони підтримуються багатьма програмами та платформами.Тому їх можна використовувати в тому числі для передачі даних в хмару. На рисунку 2.1 зображена статистика на якій чітко видно, що MQTT має більшу перевагу над іншими протоколами.

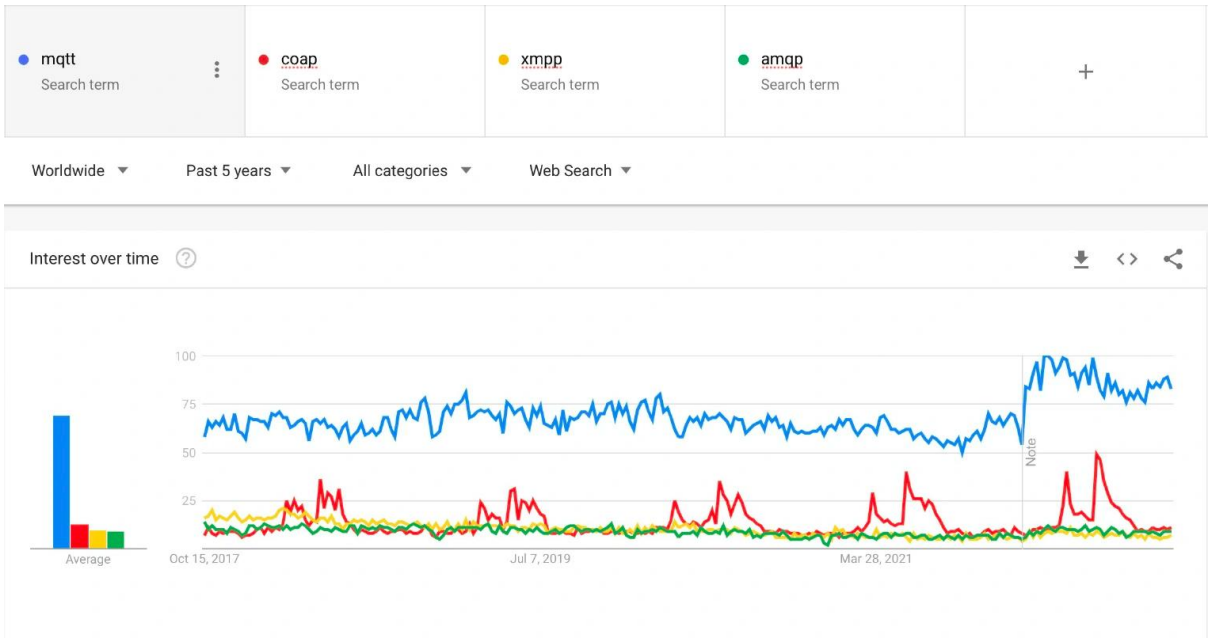

Рис. 2.1 - Статистика використання протоколів [24]

Проте ця статистика не враховує протокол HTTP, скоріш за все це пов’язано з тим, що даний протокол більше використовується для веб, а інші протоколи були розроблені саме для IoT. Проте популярність HTTP протоколу та його можливості роблять з нього сильного конкурента для MQTT. Тому в даному розділі буде проведено порівняльний аналіз та оцінка продуктивності протоколів передачі даних HTTP та MQTT на основі пропускної спроможності та розміру повідомлення. Однак спочатку розглянемо кожен протокол більш детально.

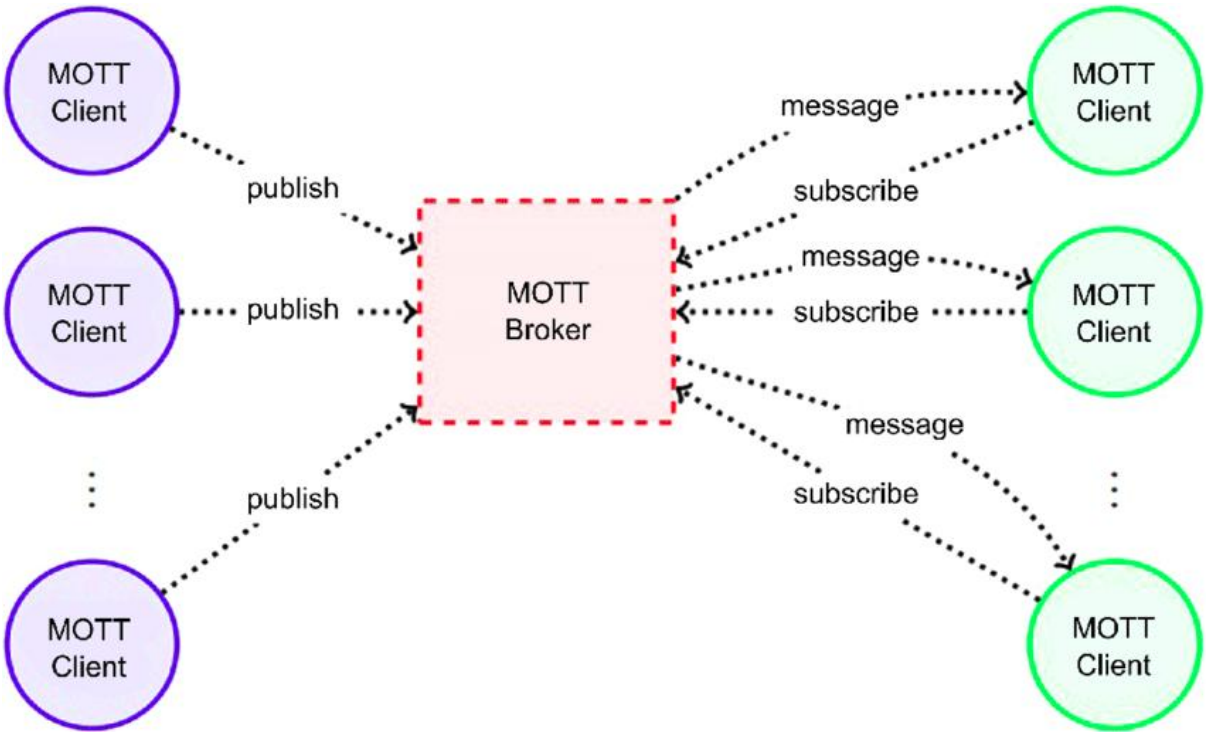

Рис. 2.2 - Модель протокола MQTT

Отже, MQTT є протоколом обміну повідомленнями, призначений до роботи з пристроях з обмеженими обчислювальними ресурсами. Протокол використовує гнучкий механізм маршрутизації та асинхронний зв'язок поверх стека TCP/IP, заснований на парадигмі публікації-підписки. MQTT складається з трьох основних компонентів: передплатника, видавця та брокера. Передплатник може повторно запитувати та отримувати повідомлення від видавця. Видавець зв'язується з брокером, щоб надсилати

повідомлення з певної теми своїм передплатникам. Нарешті, брокер отримує повідомлення від видавців та пересилає їх передплатникам.

Отже перевагами протоколу є:

- Легкий протокол, який швидко створюється та забезпечує ефективну передачу даних
- Швидка передача даних
- Низьке енергоспоживання

До недоліків можна віднести:

- Базові механізми безпеки, тому краще використовувати разом з SSL/TLS

Як було вже вказано у розділі 1.3, HTTP є основним способом спілкування веб-браузерів і серверів для обміну інформацією в Інтернеті. HTTP визначає формат та призначення повідомлень, якими обмінюються Web-компоненти, такі як клієнти та сервери. та спосіб інтерпретації полів кожного рядка повідомлення.

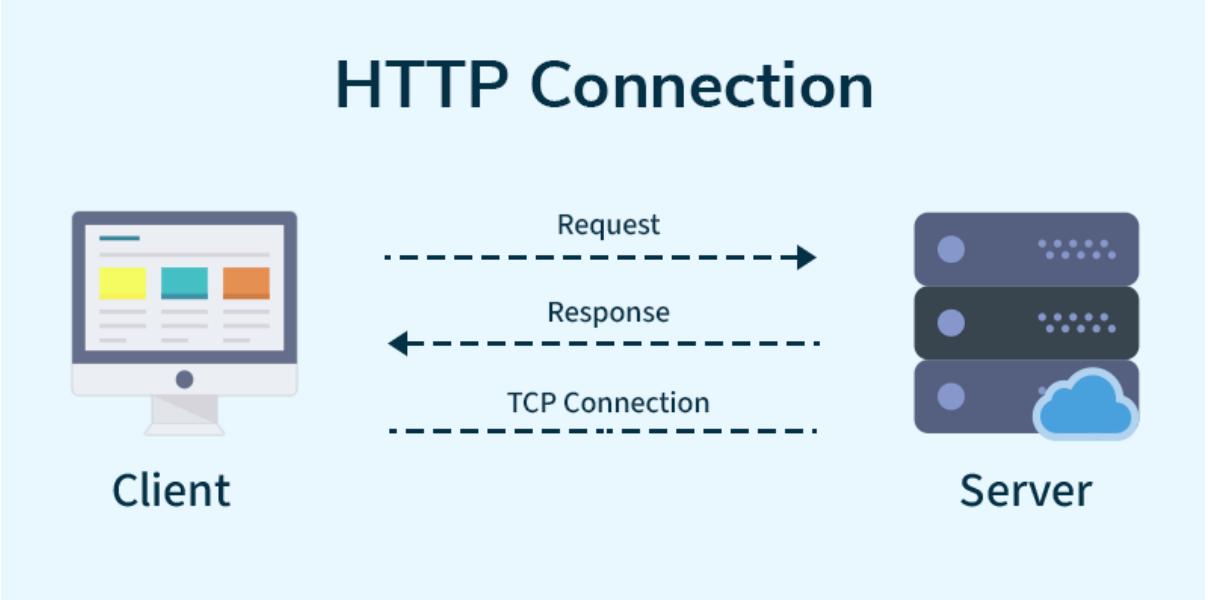

Рис. 2.3 - Модель протокола HTTP

Переваги:

- Більш низьке використання пам’яті та ЦП через меншу кількість одночасних підключень

- За рахунок встановлення зв’язку затримка зменшується, тому немає потреби в установленні зв’язку для наступних запитів

  Недоліки:

- Необхідна висока потужність для встановлення зв’язку та передачі даних

Тепер можна перейти до практичного порівняння цих протоколів. Для дослідження HTTP протоколу було створено дві програми на NodeJS. Перша є клієнт-серверним додатком з можливістю відправки JSON-об'єкта від пристрою на сервер. Його реалізація полягає в написанні API, яке прийматиме кінцеві запити, і клієнтської програми, яка буде запущена на Raspberry Pi 5, і з певним інтервалом надсилати повідомлення на сервер.

Для MQTT було використано брокер Mosquitto. Для відправки та прийому повідомлення були написані Python скрипти. Це дозволило більш швидше провести необхідні тести, видавець відправляє дані до брокера Mosquitto, а брокер вже відправляє дані клієнту.

У ході аналізу було проведено два експерименти. У першому випадку повідомлення надсилається 1 раз з інтервалом у 10 секунд протягом декількох хвилин. На цих даних розраховувалися середні показники для HTTP і MQTT протоколів. У другому випадку вид повідомлення, що надсилається, був змінений. Масив значень, який раніше містив лише чотири записи, був перетворений на масив значень, який містив у собі десять масивів з чотирма значеннями. Таким чином, досліджувався вплив збільшення вихідної інформації при відправленні даних. Пропускна спроможність розраховувалася для десяти повідомлень за формулою: $Q = I/t$, де I – обсяг інформації, а T – час передачі.

Результати аналізу відображені у таблиці 2.3

Таблиця 2.2 - Результати порівняльного аналізу

Протокол	Розмір одного повідомлення, байт	Розмір десяти повідомлень, байт	Середня швидкість передачі даних	Пропускна здатність, Байт/мс
HTTP	398	4056	141	29
MQTT	331	3868	26	148

За результатами цього аналізу можна визначити, що протокол MQTT показав найкращі результати: меншу швидкість передачі даних при подібному обсязі даних з HTTP, а також має більшу пропускну здатність, що визначає його як найбільш релевантний вибір для IoT системи.

2.4 Структурна схема системи

З огляду розділу 1.2 в нас є представлення того, як має виглядати архітектура IoT-системи. В загальному вигляді це пристрої для моніторингу, або керування, контроллер, за необхідності маршрутизатор та головний сервер. Враховуючи цю інформацію, структурна схема буде мати наступний вигляд.

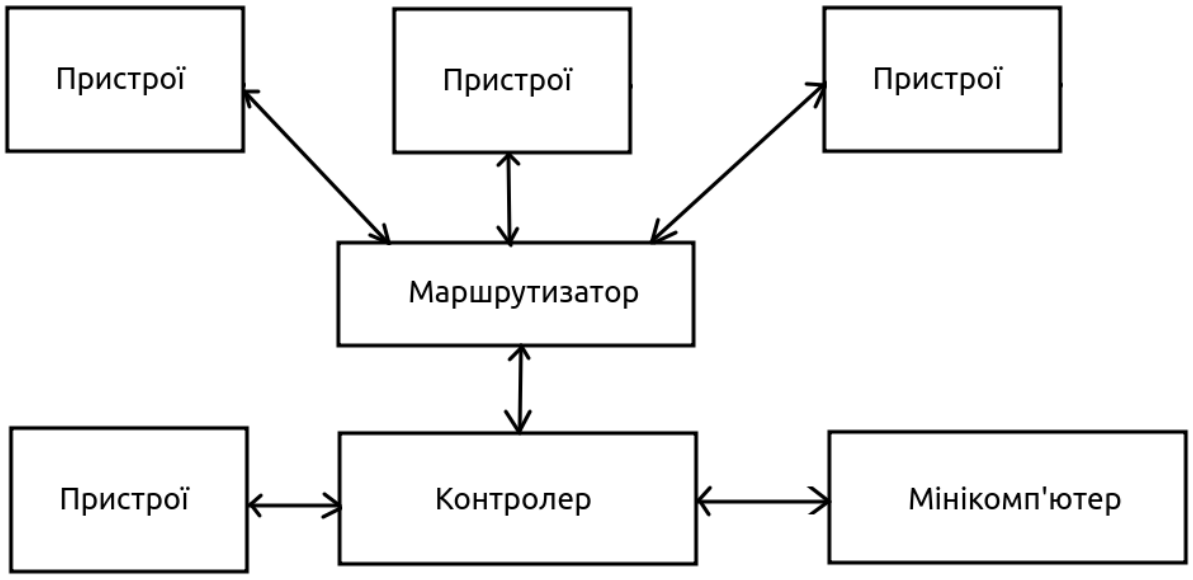

Рис. 2.4 - Структурна схема системи

Стрілки вказують в обидві сторони, бо зв’язок між елементами двосторонній. Пристрої можуть передавати дані до мінікомп’ютера, далі вони можуть бути відправлена на хмарну платформу, або відображатись в окремих додатках.

2.5 Вибір хмарної платформ IoT систем

У розділі 1.4 були розглянуті деякі хмарні платформи для впровадження систем моніторингу та керування IoT пристроями. В даному розділі проведено порівняння цих систем, щоб визначити переваги та недоліки кожної, а також визначити в яких випадках кожну з них краще застосовувати.

1. Kaa IoT

Як вже було зазначено раніше Kaa IoT є хмарною платформою з мікросервісною архітектурою (Рис. 2.5). Такий рівень абстракції можливий за рахунок певних методів:

По-перше, усі протоколи зв’язку між службами використовують HTTP і NATS для транспортування повідомлень. Ці технології чітко визначені та мають кілька реалізацій для всіх основних мов програмування. Зараз більшість мікросервісів Kaa написані на Java, Go та TypeScript.

По-друге, усі мікросервіси Kaa запаковані як образи Docker. Це допомагає операційній команді розгортати рішення Kaa без необхідності налаштування залежностей.

По-третє, на додаток до Docker використовується системи оркестровки контейнерів Kubernetes. Численні проекти екосистем, такі як Helm, NGINX, NATS, Grafana та багато інших, покращують користувацький досвід Kaa та допомагають працювати з рішеннями на основі Kaa будь-якого масштабу та будь-якої складності. [18]

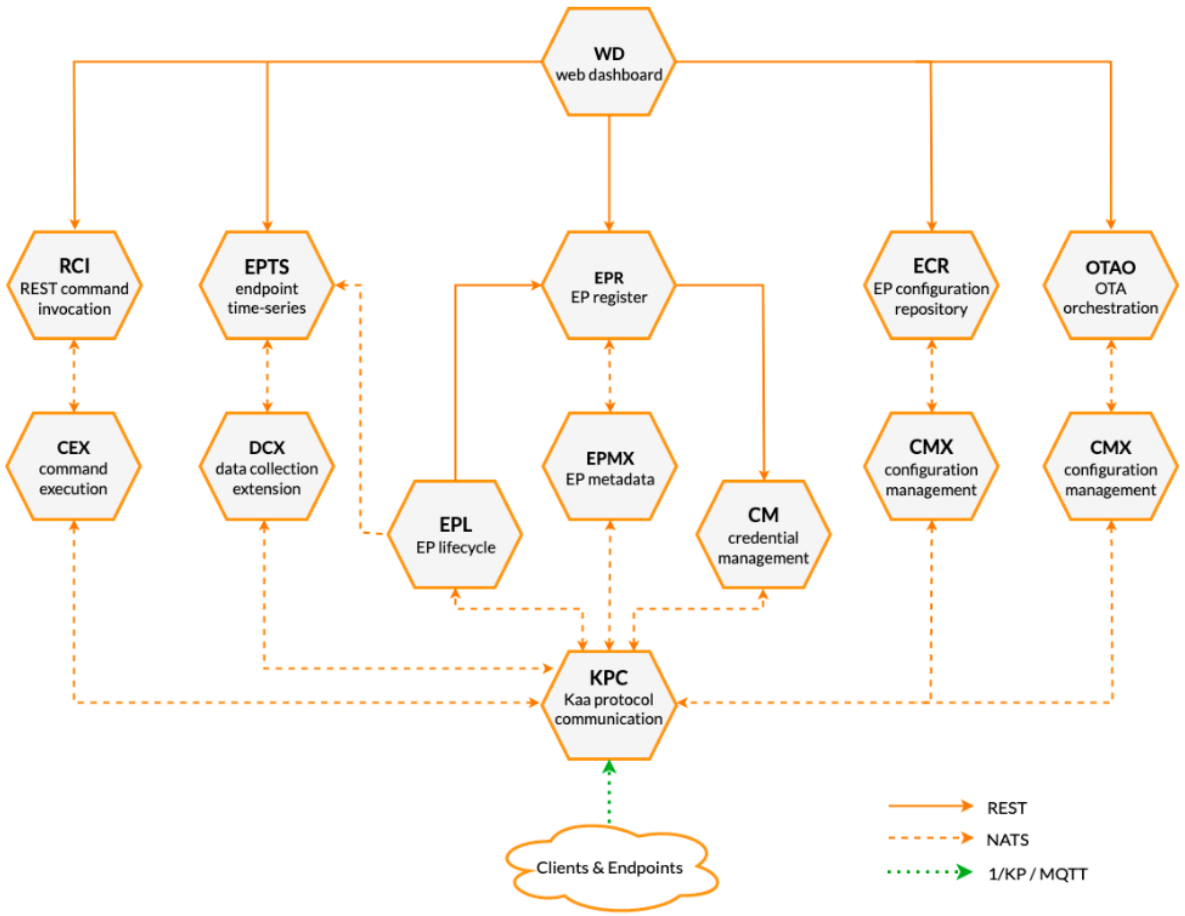

Рис. 2.5 - Архітектура Kaa IoT

Якщо ж подивитись на платформу не тільки з технічної точки зору, то Kaa IoT має гнучку модель ціноутворення, що відповідає різноманітним потребам і бюджетам. Вимоги до налаштування мінімальні, що дозволяє швидко розгортати та інтегрувати в існуючу інфраструктуру. Компанія пропонує комплексні можливості керування пристроєм, надаючи користувачам можливість безперешкодно контролювати підключення пристрою, конфігурацію та моніторинг. Налаштування є ключовою перевагою платформи Kaa IoT, що дозволяє користувачам адаптувати платформу до своїх конкретних потреб і уподобань. Незалежно від того, чи це зміна робочих процесів, створення користувальницьких інформаційних панелей або інтеграція програм сторонніх розробників.

2. AWS IoT Core

Дана платформа спрощує швидке та безпечне підключення пристроїв до хмари. Це знімає необхідність створення та управління інфраструктурою, необхідною для підтримки рішення IoT і дозволяє вам зосередитися на створенні інноваційних програм. AWS IoT Core складається з кількох основних компонентів, які забезпечують підключення та зв’язок між пристроями та хмарою:

- Шлюз пристрою - це точка входу для безпечного підключення пристроїв до AWS IoT Core. Він підтримує підключення MQTT, HTTP і WebSocket з пристроїв. Він проводить автентифікацію пристроїв та авторизує операції.
- Механізм правил дозволяє створювати правила IF THEN для обробки та маршрутизації даних із пристроїв до інших служб AWS, таких як Lambda, Kinesis тощо.
- Посередник повідомлень забезпечує публікацію повідомлень і маршрутизацію між пристроями та програмами. Він підтримує протокол MQTT із повідомленнями про публікацію/підписку.
- “Тінь пристрою” зберігає інформацію про стан пристроїв, включаючи метадані, конфігурацію та попередні значення. Він служить постійним представленням кожного пристрою.
- Реєстр, використовується для реєстрації ідентифікаційних даних пристрою та пов’язані метадані, такі як ключі та політики. Це дозволяє керувати дозволами для кожного пристрою.
- SDK для пристроїв — AWS надає SDK для вбудованих C, JavaScript, Python, iOS і Android для спрощення створення програм для пристроїв. [16]

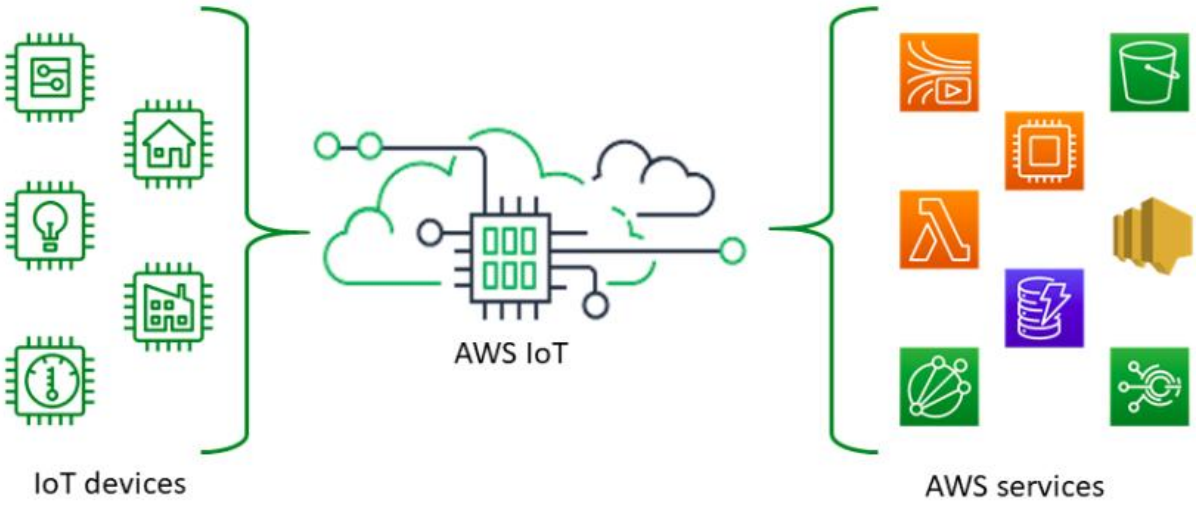

Рис. 2.6 – Архітектура AWS IoT Core

Отже як можна бачити AWS IoT Core надає керований сервіс для швидкого, безпечного та масштабного підключення пристроїв IoT до хмари. Його модель ціноутворення побудована з урахуванням різних моделей використання, але навігація в ній може бути складною, що вимагає ретельного планування, щоб уникнути не очікуваних витрат. Вимоги до налаштування для AWS IoT Core вимагають певного рівня технічної експертизи, особливо в області хмарних обчислень і мереж, що може створити проблеми для користувачів, які не знайомі з цими технологіями.

Великою перевагою є те, що варіантів налаштування в AWS IoT Core багато, що дозволяє користувачам адаптувати платформу до своїх конкретних вимог. Однак ця гнучкість має компроміс, оскільки налаштування та інтеграція різних компонентів може зайняти багато часу та бути складним.

Можливості керування пристроями в AWS IoT Core надійні, що дозволяє користувачам ефективно керувати великим парком пристроїв. Однак під час роботи з різними типами пристроїв і протоколами можуть виникнути складності, що знову ж таки вимагають глибокого розуміння та впровадження функцій керування пристроями AWS IoT Core.

Функції аналізу даних і візуалізації в AWS IoT Core дають змогу користувачам отримувати статистичні дані з даних IoT, але можуть виникнути проблеми при обробці великих обсягів даних у масштабі. Крім того, для інтеграції AWS IoT Core з іншими службами AWS для розширеної аналітики та можливостей машинного навчання можуть знадобитися додаткові знання та ресурси.

3. Microsoft Azure IoT

Microsoft Azure IoT пропонує надійну платформу для створення та керування рішеннями IoT, яка має наступні ключові характеристики:

- Можливості двонаправленого зв'язку: ця функція дозволяє пристроям IoT і хмарі спілкуватися один з одним в обох напрямках. Цей двонаправлений зв’язок забезпечує взаємодію між пристроями та хмарою в режимі реального часу, дозволяючи негайно реагувати на зміни стану пристрою.
- Телеметрія з пристрою в хмару: телеметрія від пристрою до хмари є ще однією ключовою функцією Azure IoT Hub. Ця функція дозволяє пристроям IoT надсилати телеметричні дані в хмару для обробки, аналізу та зберігання. Телеметричні дані можуть включати будь-які показники від температури, рівня вологості до показників продуктивності машини. Потім хмара може проаналізувати ці дані, щоб виявити будь-які аномалії, які можуть вказувати на потенційну проблему.
- Команди з хмари на пристрій: ця функція дозволяє Azure IoT Hub надсилати команди на пристрої IoT. Ці команди можна використовувати для керування пристроями та зміни їх поведінки.
- Подвійні пристрої та прямі методи: Device Twin — це цифрове представлення фізичного пристрою IoT у хмарі. Він зберігає поточний стан пристрою, метадані та іншу відповідну інформацію. Прямі методи дозволяють хмарі викликати дії на пристрої, як-от

перезавантаження пристрою або скидання його датчиків. Ці методи називаються «прямими», тому що вони забезпечують прямий спосіб взаємодії хмари з пристроями. [17]

Незважаючи на те, що можливості налаштування дещо обмежені порівняно з іншими платформами. Інтеграція з ширшою екосистемою Microsoft Azure є ключовою перевагою Azure IoT. Користувачі можуть використовувати широкий набір служб Azure, включаючи машинне навчання Azure і Azure Stream Analytics, щоб покращити свої рішення IoT. Однак можуть виникнути складності інтеграції, які вимагають досвіду роботи з службами та технологіями Azure.

Щодо структури ціноутворення, то вона є гнучкою та дозволяє користувачам платити лише за ті послуги, якими вони користуються. Однак орієнтуватися в цінових рівнях Azure і розуміти наслідки вартості може бути складно, особливо для користувачів, які тільки починають працювати з хмарними обчисленнями.

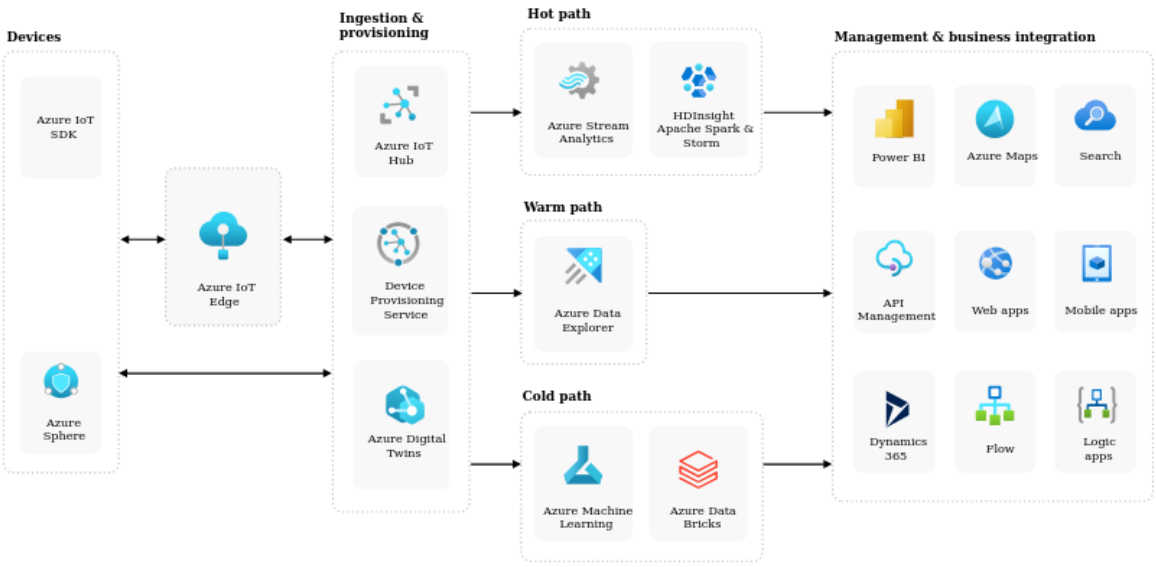

Рис. 2.7 – Архітектура Microsoft Azure IoT

Можливості керування пристроями в Microsoft Azure IoT є всеосяжними, що дозволяє користувачам легко підключати, налаштовувати та контролювати пристрої IoT, хоча іноді це може бути складним

завданням, що потребує надійних стратегій та інструментів керування пристроями.

Azure IoT забезпечує надійну аналітику даних і функції візуалізації, що дає змогу користувачам отримувати корисну інформацію з даних IoT. Параметри налаштування в Microsoft Azure IoT дещо обмежені порівняно з іншими платформами, із попередньо визначеними шаблонами та конфігураціями. Це може створити певні складнощі для користувачів із унікальними вимогами або тих, хто шукає індивідуальні рішення.

Таблиця 2.3 – Порівняльна характеристика хмарних платформ

	Kaa IoT Cloud	AWS IoT Core	Microsoft Azure IoT
Ціноутворення	На пристрій	Обчислювальні блоки + трафік + сховище + зовнішні служби	Обчислювальні одиниці + повідомлення + пам'ять
Безкоштовний ліміт	Так, 14 днів безкоштовної пробної версії, до 5 пристроїв	12-місячна безкоштовна пробна версія в розмірі 300$	Так, до 8000 повідомлень на день
Готове IoT рішення	Кілька годин	Кілька днів	Кілька днів
Необхідна рівень знань IoT	Базовий, використання консолі не потрібно	Більш експертний, консоль і програмування	Середній, консольні та програмування
Налаштування	Дозволяє розгортати власні служби + настроюваний інтерфейс користувача + інтеграція хмарних провайдерів за допомогою Kafka	Дозволяє використовувати сервіси AWS, без стандартного інтерфейсу користувача для рішення IoT	Обмежено, переважно через служби Azure; немає готових настроюваних рішень інтерфейсу користувача

Ключові переваги	настроюваний UI; доступне самостійне розгортання	Гарний рівень обслуговування; велика хмарна екосистема	Інтеграція з екосистемою Microsoft

Підсумовуючи можна зробити наступні висновки, платформа Kaa IoT постає як надійне та універсальне рішення для компаній, які розпочинають ініціативи IoT. Завдяки надійним функціям, гнучким параметрам налаштування та можливостям повної інтеграції він є найкращим вибором для організацій, які хочуть використовувати потужність IoT для отримання стратегічної переваги.

AWS IoT Core є потужною платформою для створення та керування рішеннями IoT, яка дозволяє спробувати IoT технології з безкоштовного аккаунту та з використанням потужного набору інструментів і послуг у величезній екосистемі AWS. Хоча і варто відмітити, що в параметрах налаштування та певних складнощах інтеграції вимагає певних знань.

Microsoft Azure IoT надає потужну платформу для розробки та керування рішеннями IoT, яка може похвалитися повною інтеграцією зі службами Azure та гнучкою структурою ціноутворення, але його можливості налаштування дещо обмежені порівняно з іншими платформами.

Висновки до розділу 2

Сформовані вимоги до системи моніторингу та керування IoT пристроїв на виробництві. Обрані варіанти мікрокомп'ютера та протоколів рівня додатків забезпечують оптимальний баланс між функціональністю, продуктивністю та енергоефективністю. Хмарні платформи можуть використовуватись як повноцінна частина для рівня обробки даних, або ж

як доповнення до системи, яке дозволяє використовувати спеціальні розширення та програми.

В наступному розділі на базі цієї інформації буде розроблено прототип IoT-системи моніторингу якості повітря для виробництва з друку реклами.

РОЗДІЛ 3 ПРОТОТИП ІОТ-СИСТЕМИ МОНІТОРИНГУ ЯКОСТІ ПОВІТРЯ НА ВИРОБНИЦТВІ

В даному розділі буде розроблено прототип IoT-системи для моніторингу якості повітря на виробництві з друку реклами на різних речах і поверхнях, а також надані рекомендації з впровадження подібних систем

на виробництві. З огляду на розділ 2.2 у якості мінікомп’ютера було обрано Raspberry Pi 5. Структура системи буде подібною до структури представленої в розділі 2.4. В подальших розділах буде розглянуто необхідне обладнання та програмне забезпечення.

3.1 Вибір необхідного обладнання

Для подальшого створення системи моніторингу повітря на базі IoT пристроїв треба обрати датчики та контроллер(адаптер) який дозволить їх підключити до мікрокомп'ютера. Почнемо з датчика для вимірювання основних параметрів повітря. До цих параметрів можна віднести рівень вуглекислого газу, вологість, температуру, рівень формальдегіду та летких органічних сполук. Кожен з цих параметрів по своєму може вказувати на те, що дозволяє формувати певний висновок на основі цих результатів:

- Вуглекислий газ (CO2): високий рівень CO2 свідчить про погану вентиляцію, або проблеми з системою вентиляції. Показник від 400 до 800 ppm відповідає нормі, більше 800 ppm може призвести до дискомфорту, а більше 1400 ppm свідчить про проблеми з концентрацією вуглекислого газу в повітрі [36]
- Температура і вологість: норма для даних параметрів підбирається індивідуально в залежності від конкретних потреб. У випадку з фарбою це може впливати на час її висихання та призвести до підвищеної концентрації інших речовин в повітрі
- Формальдегід (HCHO): Формальдегід це токсична речовина, яка виділяється в повітря з будівельних матеріалів, лакофарбових покриттів та продуктів деревообробки. Допустимою концентрацією є рівень до 0.1 ppm, якщо ж показник більше ніж 0.3 ppm то це свідчить про небезпечні умови для здоров’я [37]
- Леткі органічні сполуки (VOC): це органічні сполуки, які включають бензол, толуол, ксилол, аміак та інші речовини, що є компонентами

фарб, розчинників, побутової хімії та синтетичних матеріалів. Безпечним рівнем є 0 - 0.5 ppm, рівень 0.5 - 1.5 ppm вказує на помірне забруднення, а якщо більше 1.5 ppm, то концентрація органічних сполук може бути шкідливою для здоров’я.[38]

Враховуючи описані параметри у якості датчика для моніторингу якості повітря було обрано Tuya Smart Air Box (Рис. 3.1). Оскільки він вимірює показники, які були зазначені вище та як зазначено в документації має високу чутливість, що дозволяє отримувати доволі точні показники. Датчик має невеликі розміри та невелику ціну у порівнянні з конкурентами.

Рис. 3.1 - Датчик Tuya Smart Air Box

Наступним пристроєм який необхідно вибрати це адаптер для Zigbee. Даний пристрій відповідає за забезпечення зв’язку між пристроями та центральним вузлом. Можна сказати, що він є шлюзом який приймає сигнал від датчиків та передає їх до міні-комп'ютера проводячи обробку за допомогою спеціального програмного забезпечення. Основними критеріями для вибору адаптера є наступні пункти:

- Сумістність: адаптер має підтримувати стандарт Zigbee 3.0, а також мати зворотну сумісність з попередніми версіями протоколу

- Адаптер має легко інтегруватися з мінікомп'ютером Raspberry Pi. Для цього адаптерам має мати USB.
- Наявність прошивки яка буде суміна з програмним забезпеченням Zigbee2MQTT

Враховуючи ці критерії у якості адаптера було обрано Zigbee 3.0 USB Sonoff Dongle Plus (Рис. 3.2).

Рис. 3.2 - Zigbee 3.0 USB Sonoff Dongle Plus

Це універсальний USB-накопичувач Zigbee, який має прошивку на основі EZNet 6.10.3 із коробки, що дозволяє працювати з Zigbee2MQTT, або ж використовуючи схоже програмне забезпечення з відкритим кодом для керування пристроями Zigbee. Характеристики пристрою наступні:[27]

- Робоча частота: 2.4 ГГц
- Прошивка основі EZNet 6.10.3
- Максимальна кількість пристроїв: 32
- Вихідне посилення +20 дБм

- Габарити: 75x25, 5x13, 5mm

Отже наразі обрано всі мінімально необхідні компоненти для створення системи моніторингу керування IoT пристроями. Варто зазначити, що ключовими елементами є мінікомп'ютер та адаптер, оскільки саме вони відповідають за взаємодію з пристроями IoT з використанням відповідних протоколів. Як і було зазначено на початку розділу, датчики моніторингу повітря були обрані для перевірки системи та практичної демонстрації процесу налаштування такої системи. Основна апаратна частина є універсальною основою для будь-яких пристроїв з підтримкою Zigbee протоколу, та може бути адаптована для різних сценаріїв.

3.2 Налаштування Raspberry Pi

Перш за все необхідно встановити операційну систему на Raspberry Pi, та налаштувати необхідне програмне забезпечення. Для спрощення цього процессу буде використано утиліту Raspberry Pi Imager, яка дозволяє користувачам вибрати конкретну операційну систему, яку вони хотіли б встановити, а потім програма завантажує останню версію цієї операційної системи у фоновому режимі перед початком інсталяції. Також необхідно мати картку microSD ємністю бажано 16 ГБ або більше. Варто зазначити, що може знадобиться картрідер, якщо комп’ютер не має вбудованого.

Після завантаження образу для операційної системи запускаємо Raspberry Pi Imager. В розділі Choose Device обираємо пристрій для якого будемо записувати операційну систему, в полі Choose OS обираємо необхідну версію, в полі Choose SD Card обираємо нашу microSD карту. Після цього натискаємо Next і обираємо необхідні налаштування:

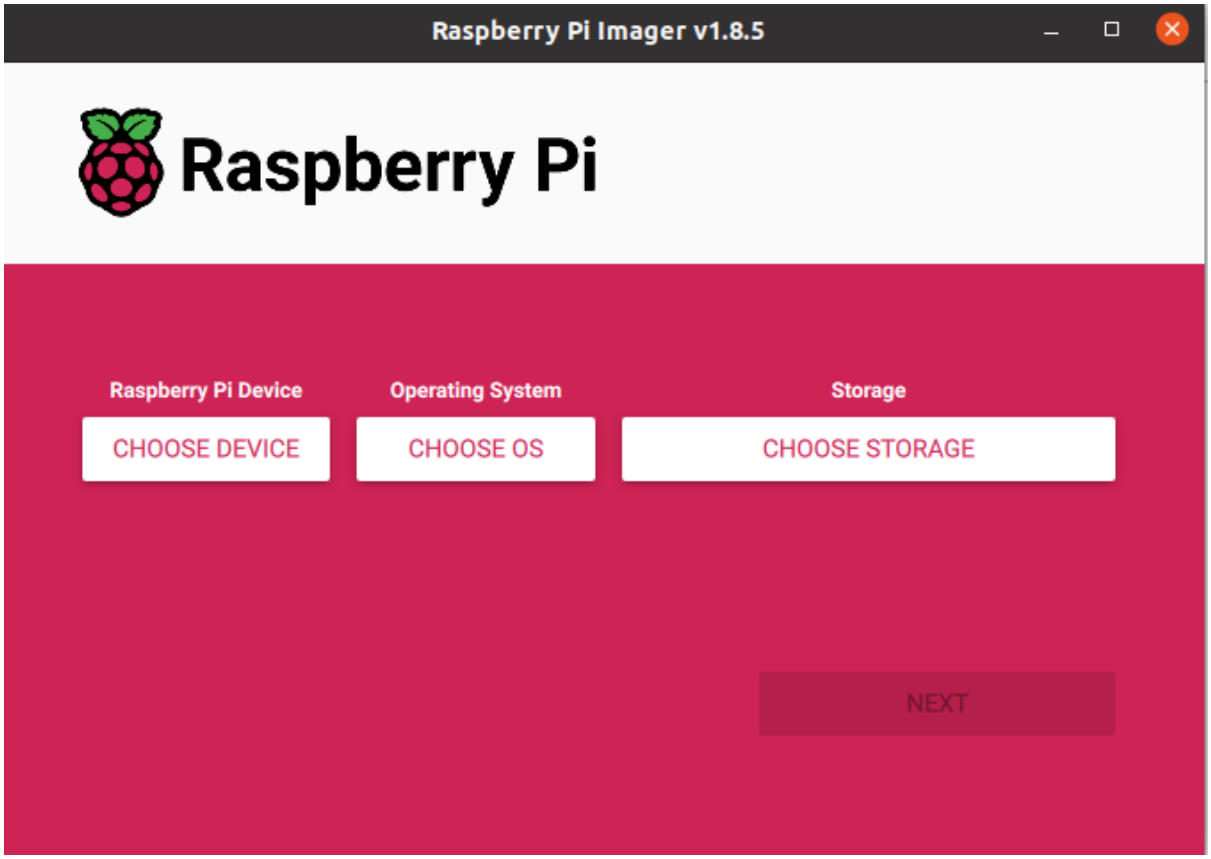

Рис. 3.3 - Вікно запису програми Raspberry Pi Imager

Коли Raspberry Pi Imager завершить запис файлів на microSD карту, він перевірить, чи зображення на карті ідентичне файлу зображення, який використовувався для запису зображення. Зазвичай це займає менше хвилини, але може зайняти більше часу. Коли процес перевірки буде завершено, відкриється вікно сповіщення про те, що запис пройшов успішно та що тепер можна безпечно вийняти microSD карту. Наступним етапом виконуємо перший запуск, потрібно трохи почекати поки всі налаштування для запуску будуть виконані. В результаті можна буде підключитись до Raspberry Pi за допомогою монітору, або через ssh.

Для підключення через ssh треба встановити Putty. Підключаємо Raspberry Pi до живлення та Ethernet провод до комп’ютера. Запускаємо програму Putty та вводимо у якості хоста raspberrypi.local, щоб підключитись до Raspberry Pi. На перший раз може виникнути помилка з хостом, після цього треба спробувати ще раз, після чого буде відкрито

командне вікно для логіну. Вводимо ім’я та пароль, які були вказані при запису OS на microSD. В результаті отримуємо доступ до терміналу Raspberry Pi (рисунок 3.4).

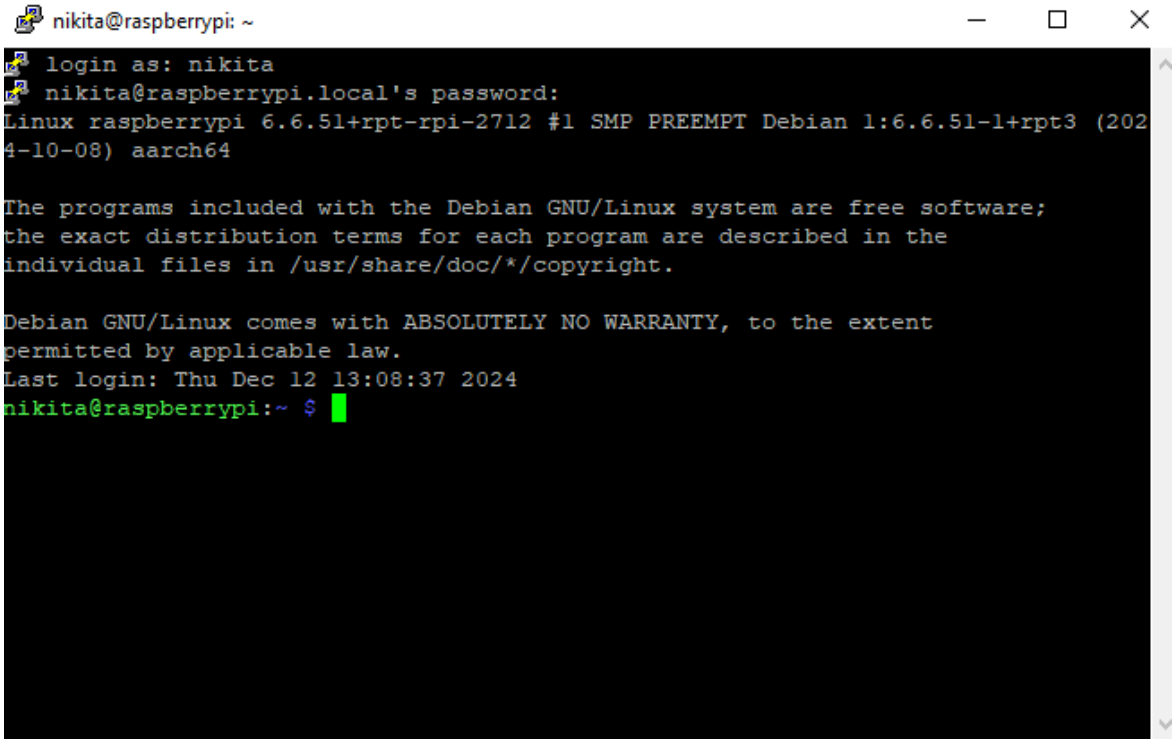

Рис. 3.4 - Термінал Raspberry Pi 5

В подальшому можна використовувати цей варіант підключення, але він потребує прямого з’єднання з мінікомп’ютером. Оскільки мінікомп’ютер підключений до мережі Wi-Fi, то можна підключатись до нього використовуючи IP-адресу. Вводим команду ifconfig та шукаємо розділ wlan0, де і буде вказано IP-адрес. Вводим IP-адресу замість хоста raspberrypi.local та можемо знов підключитичь до Raspberry.

3.3 Налаштування програмного забезпечення

Налаштування Docker

Першою програмною, яку буде встановлено це Docker. Docker — це платформа з відкритим кодом, створена для розробки, доставки та виконання програм. Вона забезпечує можливість ізолювати програми від

інфраструктури, що дозволяє запускати їх швидше та безпечніше. Docker дозволяє упаковувати програми в ізольоване середовище, яке називається контейнером. Завдяки ізоляції та безпеці можна одночасно запускати багато контейнерів на одному хості. Контейнери є легкими та містять усе необхідне для роботи програм, причому кожен контейнер виконує окрему програму. Така структура робить систему більш стійкою, оскільки якщо трапиться якийсь збій в одному контейнері, то це не вплине на всю систему, та інші контейнери і вони продовжать працювати.

Спочатку потрібно додати всі офіційні репозиторії до нашої машини, щоб APT не захоплював неофіційні пакунки в стандартних репозиторіях Debian. Для встановлення репозитаріїв виконаємо наступні команди:

```
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/asc -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
echo \"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \ $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Далі ми використовуємо apt для встановлення нових пакетів за допомогою наступної команди:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

За замовчуванням лише користувач root може запускати команди докерів. За необхідності можна запускати команди докерів зі стандартного облікового запису користувача. Отже, команда додасть обліковий запис користувача до групи докерів:

```
sudo usermod -aG docker ${USER}
```

Для перевірки вводимо команду docker –version

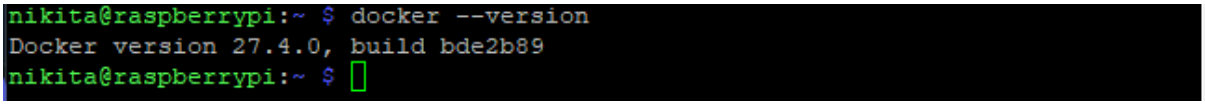

Рис. 3.5 - Перевірка версії Docker

Далі будуть встановлені необхідні програми Zigbee2MQTT і Mosquitto для використання Zigbee та MQTT протоколу, а також Node-red для роботи з даними з пристроїв.

Налаштування Node-Red

Node-Red — це інструмент програмування з відкритим вихідним кодом для творчого та легкого підключення апаратних пристроїв, API та онлайн-сервісів. В першу чергу, це візуальний інструмент, розроблений для IoT, але також може використовуватися для інших програм для дуже швидкого збирання потоків різноманітних послуг. Він об’єднаний за допомогою блоків програмного коду на основі JavaScript, які називаються вузлами. Вузли візуально перетягуються, щоб робить розробку простішою, легшою для повторення та швидшою для масштабування. Node-Red працює на комп’ютерах Windows, Mac і Linux, а також вбудованих пристроях. [31]

Для встановлення Node-Red потрібно в терміналі ввести наступну команду: docker run -it -p 1880:1880 -v node_red_data:/data --name mynodered -e TZ=Europe/Athens --restart unless-stopped nodered/node-red

Після введення команди ми побачимо процесс налаштування, який все зробить автоматично. Для початку роботи з графічною частиною Node-red можна в браузері Raspberry ввести лінк http://localhost:1880/, після чого відкриється сторінка для роботи вузлами Node-red. Якщо не зручно використовувати VNC Viewer для роботи з Node-red, то можна відкрити його через браузер на іншому комп’ютері використовуючи лінк з IP-адресою, замість localhost.

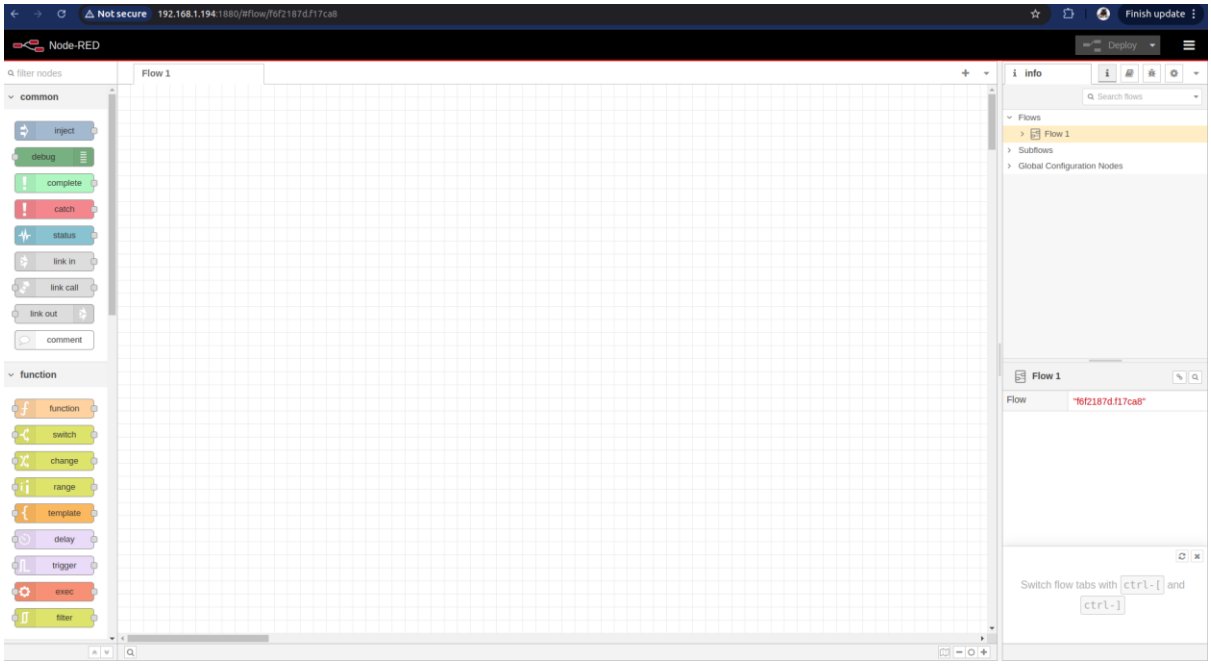

Рис. 3.6 - Сторінка для роботи з вузлами

Налаштування Mosquitto

Eclipse Mosquitto — це відкритий брокер повідомлень, що підтримує протокол MQTT версій 5.0, 3.1.1 і 3.1. Завдяки своїй компактності, Mosquitto підходить для використання як на малопотужних одноплатних комп'ютерах, так і на повноцінних серверах. Крім того, проєкт Mosquitto пропонує C-бібліотеку для реалізації клієнтів MQTT, а також популярні клієнтські утиліти командного рядка mosquitto_pub і mosquitto_sub для роботи з MQTT. Щоб встановити Mosquitto, потрібно створити два файли і папку ось цією командою. При цьому потрібно перебувати в домашній директорії, перейти в домашню директорію можна за допомогою команди:

```
mkdir mosquitto && \ touch mosquitto/mosquitto.conf mosquitto/mosquitto_password_file && \ echo -e "listener 1883\nallow_anonymous false\npassword_file /mosquitto/config/mosquitto_password_file" > mosquitto/mosquitto.conf
```

Потім виконуємо команду установки контейнера з Mosquitto:

```
docker run -it -p 1883:1883 -p 9001:9001 -v `pwd`/mosquitto:/mosquitto/config --name mosquitto --restart unless-stopped eclipse-mosquitto
```

І в кінці треба ввести команду де буде вказано пароль для користувача від імені якого підключатиметься Zigbee2MQTT та інші пристрої:

docker exec -it mosquitto mosquitto_passwd -c /mosquitto/config/mosquitto_password_file iot_user

Перезапускаем контейнер та перевіряємо роботу брокера. Для цього переходимо до Node-red та додамо вхідну ноду для mqtt. В налаштуваннях вводимо сервер та дані користувача, які були створені в попередньому кроці. Після цього натискаємо кнопку Deploy і під нодою буде статус connected, що означає успішне з'єднання з брокером (рисунок 3.7).

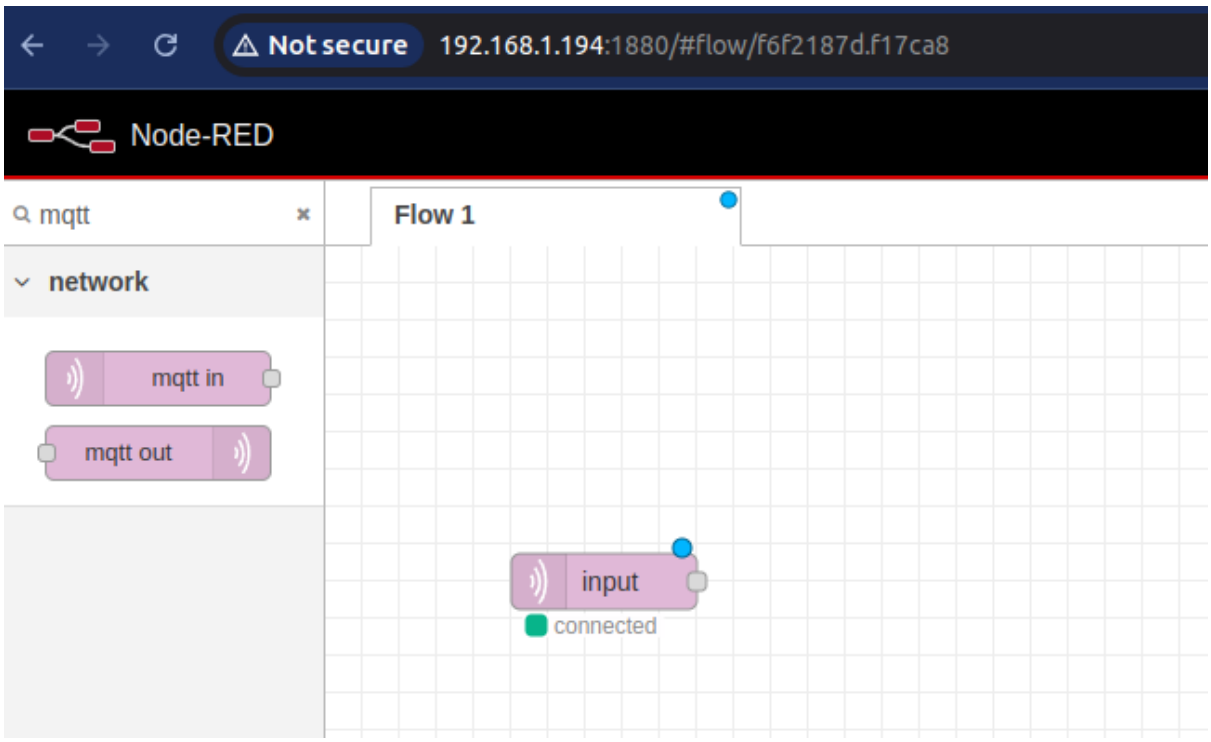

Рис. 3.7 - Перевірка налаштування Mosquitto

Налаштування Zigbee2MQTT

Наступним етапом треба підключити Zigbee адаптер до Zigbee2MQTT, щоб мати змогу отримувати інформацію з пристроїв та передавати її до Node-red. Zigbee2MQTT - це проект, який використовує протокол MQTT для передачі даних з пристроїв Zigbee до MQTT брокера.

Ключовою перевагою Zigbee2MQTT є те, що він усуває потребу у власних мостах або шлюзах Zigbee. Спочатку введемо команду для скачування конфіг файлу:

wget https://raw.githubusercontent.com/Koenkk/zigbee2mqtt/master/data/configuration.yaml -P data

Вимкористовуюємо команду nano data/configuration.yaml, щоб змінити налаштування для лінку на сервер, додати дані користувача для MQTT та додати порт для frontend. В результаті маємо такі налаштування:

```
# Home Assistant integration (MQTT discovery)
homeassistant: false

# allow new devices to join
permit_join: true

# MQTT settings
mqtt:
  # MQTT base topic for zigbee2mqtt MQTT messages
  base_topic: zigbee2mqtt
  # MQTT server URL
  server: 'mqtt://[IP-адреса]:1883'
  # MQTT server authentication, uncomment if required:
  user:  [ім’я користувача MQTT]
  password: [пароль користувача MQTT]

# Serial settings
serial:
  # Location of CC2531 USB sniffer
  port: /dev/ttyUSB0

frontend:
```

port: 8080

Далі виконуємо команду ls /dev/serial/by-id, яка покаже id zigbee адаптера вставленого в мінікомп'ютер, приклад результату на рисунку 3.8.

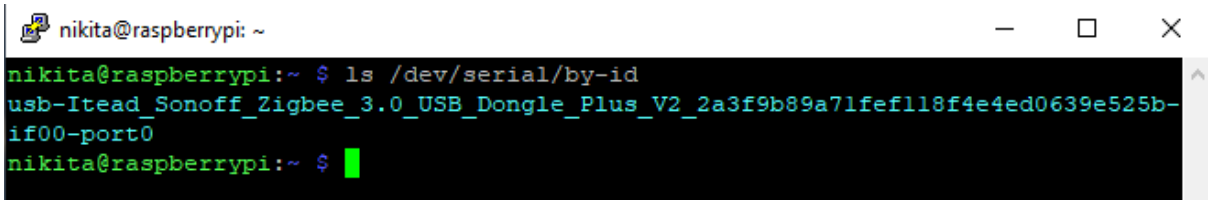

Рис. 3.8 - Айді адаптера в системі

Тепер змінюємо в цій команді id адаптера на той, що тільки-но дізналися і встановлюємо Zigbee2MQTT:

```
docker run \
  --name zigbee2mqtt
  --restart=unless-stopped \
--device=/dev/serial/by-id/usb-
Itead_Sonoff_Zigbee_3.0_USB_Dongle_Plus_V2_2a3f9b89a71fef118f4
e4ed0639e525b-if00-port0:/dev/ttyUSB0 \
  -p 8080:8080 \
  -v $(pwd)/data:/app/data \
  -v /run/udev:/run/udev:ro \
  -e TZ=Europe/Amsterdam \
  koenkk/zigbee2mqtt
```

Після звершення налаштування переходимо за адресою з IP як у пунктах вище, та використовуємо порт 8080. В результаті отримаємо доступ до панелі керування пристроями (рисунок 3.9).

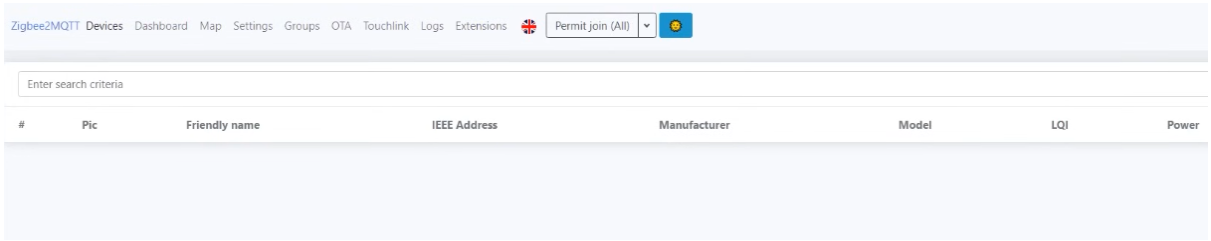

Рис. 3.9 - Панель керування пристроями в Zigbee2MQTT

Тепер всі основні програми встановлені та готові до роботи. В наступному розділі буде проведена демонстрація роботи системи.

3.4 Демонстрація роботи системи

В даному розділі проведена демонстрації роботи прототипу IoT-системи для моніторингу якості повітря. Апаратне та програмне забезпечення налаштоване та готове до роботи. Після додавання датчика до Zigbee2MQTT до нього можна отримати доступ через ноду mqtt in.

Додаємо вхідну ноду для отримання даних з датчика, після додаємо ноду function, яка дозволяє взаємодіяти з даними які отримує. В даному випадку її буде використано щоб дістати необхідні дані з об’єкта. Додаємо ноду debug для перевірки об’єкту даних який приходить від датчика. Для відображення даних у вигляді графіків встановлюємо плагін node-red-dashboard. Додаємо вузол chart для відображення даних у вигляді графіку. В результаті маємо такий вигляд потік даних (рисунок 3.10):

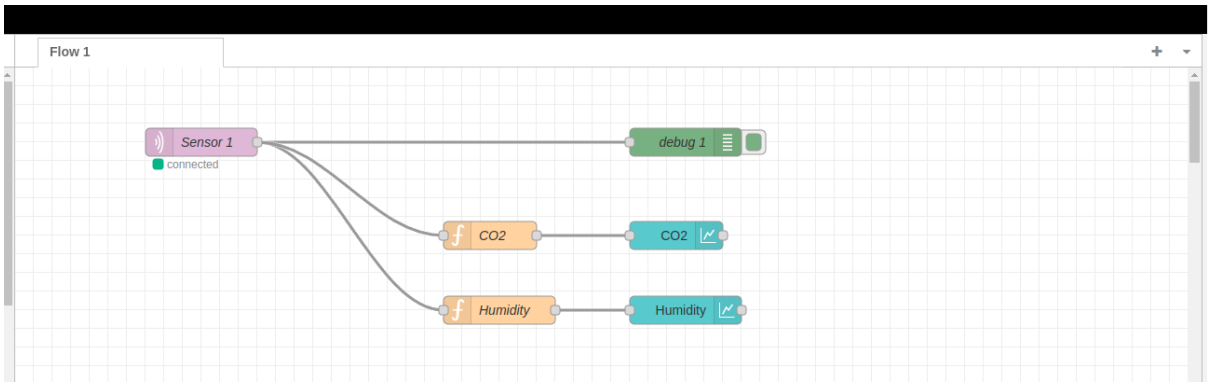

Рис. 3.10 - Приклад обробки та відображення даних

Графіки будуть відображатись на окремій сторінці за лінком http://(IP-адрес):1880/ui (рисунок 3.11).

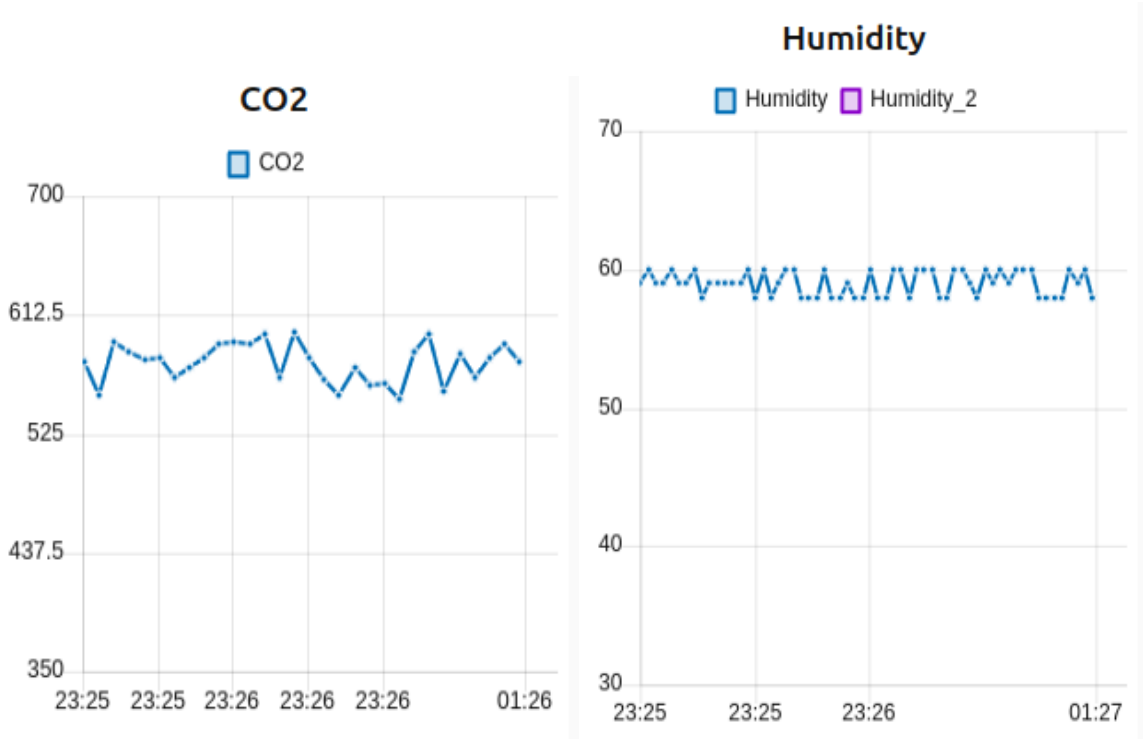

Рис. 3.11 - Приклад відображення графіків рівня вологи та CO2

Система може не тільки відображати дані, а й використовувати їх для аналізу, зробимо простий приклад. Додамо ноду switch яка виконує дію якщо умова виконується. В якості умови задамо, що дія має виконуватись якщо рівень VOC більше або дорівнює 0.5. Якщо умова вірна, то на сторінці з графіками буде з’являтись повідомлення про високий рівень VOC (рисунок 3.12).

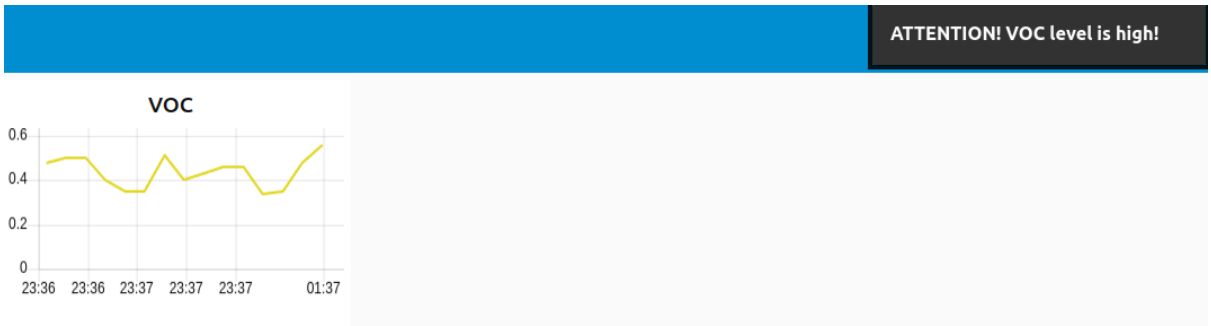

Рис. 3.12 - Відображення повідомлення при перевищенні рівня VOC

Нода show notification відповідає за відображення повідомлення та його налаштування, загальний вигляд схеми наразі як на рисунку 3.13/

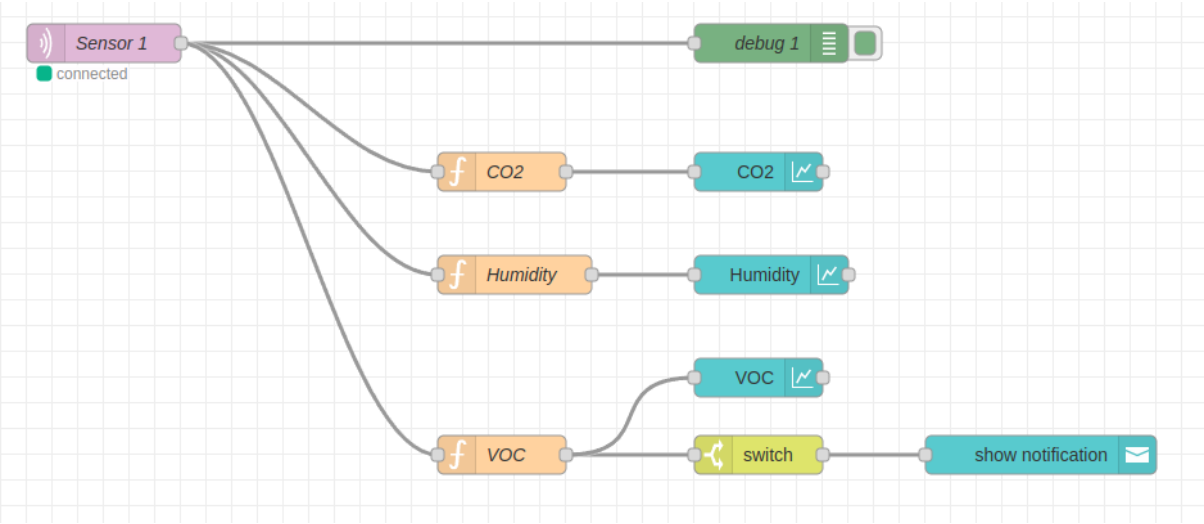

Рис. 3.13 - Схема системи з можливістю робити дію на основі значень

Проте зараз ці дані ніяк не зберігаються, якщо перезапустити Node-red або весь мінікомп'ютер, то вони будуть втрачені. Щоб зберегти дані в файлі додамо функцію для їх отримання з об'єкта, csv ноду щоб налаштувати формат файлу та ноду для запису де буде вказано шлях до файла.

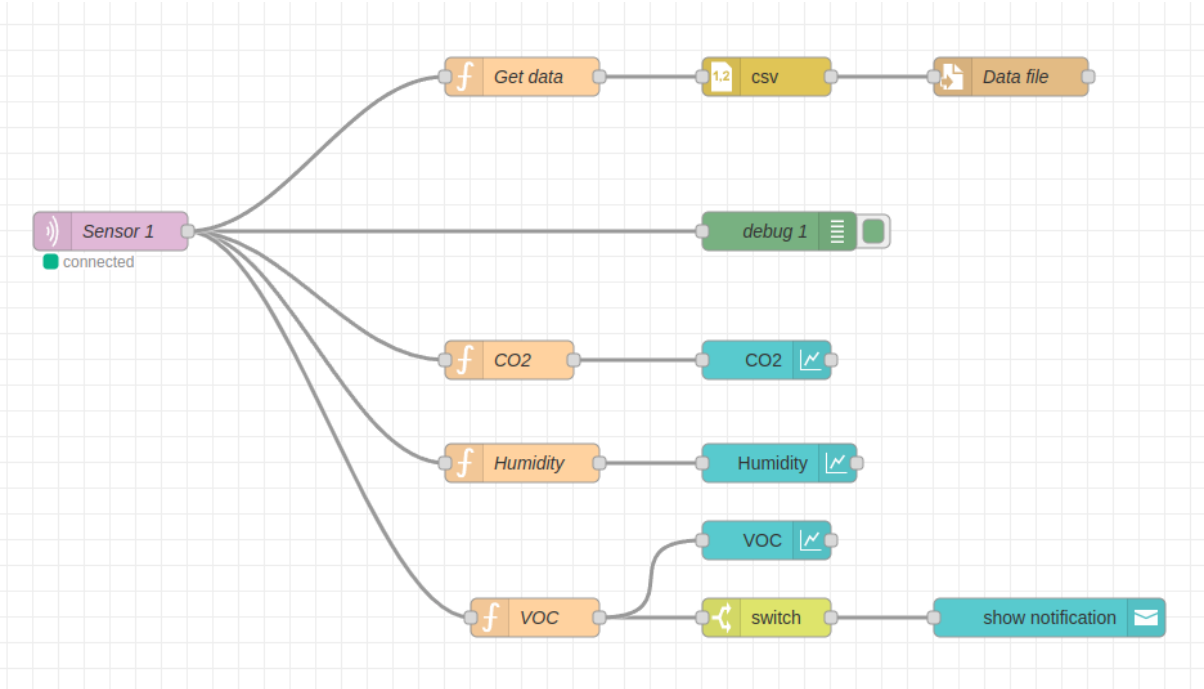

Рис. 3.14 - Схема з нодами для збереження даних

Функція для отримання даних з об'єкту має наступний вигляд:

```
var co2 = msg.payload.co2;
```

```
    var formaldehyd = msg.payload.formaldehyd;
    var humidity = msg.payload.humidity;
    var temperature = msg.payload.temperature;
    var voc = msg.payload.voc;
    var linkquality = msg.payload.linkquality;
    var data = {
       'co2': co2,
       'formaldehyd': formaldehyd,
       'humidity': humidity,
       'temperature': temperature,
       'voc': voc,
       'linkquality': linkquality,
    };
return {payload:data};
```

В csv ноді вказуємо стовпчики у які треба записувати дані: co2,formaldehyd,humidity,temperature,voc,linkquality.

Вказуємо шлях до файлу та його ім’я, також треба встановити шифрування на utf-8. Запускаємо деплой та чекаємо якийсь час для запису даних в файл.

	A	B	C	D	E	F	G
1	co2	formaldehyd	humidity	temperature	voc	linkquality	
2	559	315	59	26.8	0.36	96	
3	592	332	60	26.8	0.32	97	
4	570	342	60	26	0.31	96	
5	589	301	58	26.9	0.29	97	
6	590	308	59	27	0.27	96	
7	556	318	58	26.6	0.23	99	
8	595	323	59	27	0.25	97	
9	591	311	60	26.8	0.22	95	
10	589	318	60	26.1	0.25	94	
11	580	318	59	26.5	0.22	93	
12	598	314	60	26.8	0.21	93	
13	558	304	58	26.6	0.21	92	
14	573	305	58	26.4	0.21	91	
15	589	310	58	26.3	0.29	96	
16	564	335	58	26.9	0.3	92	
17	565	348	60	26.2	0.32	94	
18	555	335	59	26.6	0.3	99	
19	565	341	58	26.9	0.28	93	
20	554	342	59	26.7	0.26	94	
21	565	346	60	26.1	0.27	95	
22	583	346	59	26.2	0.26	96	
23	587	344	59	26.5	0.26	96	
24	567	344	58	26.3	0.23	100	
25	590	322	60	26.8	0.21	104	
26	600	329	59	26.2	0.27	108	
27	575	329	59	26.6	0.28	109	
28	550	322	58	26.7	0.28	112	
29	553	323	60	27	0.25	108	
30	560	327	60	26.9	0.22	109	
31	555	328	60	26.7	0.2	108	

Рис. 3.15 - Дані записані в файл

В результаті було розроблено робочий прототип IoT-системи для моніторингу якості повітря на виробництві. Система отримує параметри та може відображати їх у вигляді графіку. Для збереження даних використовується файл в CSV форматі, що дозволяє потім зручно переглядати дані. Для подальшого розвитку можна додати систему сповіщення, яка буде надсилати інформація на телефон у разі критичної норми якогось з параметрів.

Висновки до розділу 3

За допомогою прототипу системи моніторингу якості повітря на виробництві було продемонстровано практичну реалізацію побудови бюджетної IoT-системи моніторингу якості повітря з агрегацією та обробленням даних у хмарі. Вибір обладнання забезпечив прийнятну швидкодію та коректність функціонування системи.

Отримані результати підтверджують, що описана структурна схема у розділі 2 надає можливість для подальшого масштабування системи та адаптації до специфічних умов різних виробничих процесів.

Розділ 4
РЕКОМЕНДАЦІЇ ЩОДО ВПРОВАДЖЕННЯ ІОТ-СИСТЕМИ

4.1 Алгоритм вибору системи IoT

Впровадження системи моніторингу та керування IoT пристроями на виробництві доволі непроста задача, як правило для будь-яких компаній. Велике різноманіття різних протоколів, платформ, апаратного та програмного забезпечення не дають чіткого представлення який варіант краще обрати. Для вирішення цієї проблеми було розроблено алгоритм (рисунок 3.1) для вибору системи моніторингу та керування IoT пристроями на виробництві, який має допомогти вирішити це питання. Перш ніж його використовувати необхідно виконати декілька пунктів:

1. чітко розуміти яку систему треба зробити і що вона має робити
2. визначити кількість пристроїв, яка необхідна для цієї системи

Нижче представлений опис блок-схеми алгоритму для більш чіткого розуміння його роботи та логіки.

- Перше запитання “Компанія велика ?”: необхідно для того, щоб визначити чи треба відкинути варіант з малою кількістю пристроїв. Оскільки як правило великі компанії використовують велику кількість пристроїв. Якщо відповідь “Так”, то переходимо до запитання про типи пристроїв. Якщо відповідь “Ні”, то до запитання про їх кількість.
- Запитання “Пристрої одного типу ?”: необхідне для того, щоб визначити, чи варто розглядати варіант з автоматизацією. Оскільки в наш час деякі компанії використовують комбінацію SCADA та IoT, то варто це теж врахувати. Якщо ж прилади одного типу, то система буде простою. Відповівши “Ні” переходимо до запитання про автоматизацію, якщо відповідь “Так”, то переходимо до перевірки готових рішень.

- Запитання ”Автоматизація виробництва ?”: як і було описано вище, потрібне для того щоб врахувати можливість використання комбінованого варіанту з SCADA. Якщо відповідь “Ні” то можна переходити до розгляду різних IoT рішень, якщо “Так” то необхідно звертатись до відповідних спеціалістів.
- Запитання “Кількість приладів > 7 ?”: має важливий момент. Кількість пристроїв було обрана з огляду на рекомендації по кількості пристроїв для протоколів BLE та Wi-Fi. Якщо відповідь “Ні”, то варто розглянути можливість керування через мобільний застосунок. Якщо відповідь “Так”, то краще обирати інший протокол та переходити до огляду різних рішень.
- Запитання “Чи є готове рішення ?”: є ключовим в даному алгоритмі. Якщо відповідь “Так”, то переходимо перевірки цього рішення. Якщо “Ні”, то переходимо до розробки своєї платформи. Важливо зазначити, що це не розробка з нуля, а розробка з використанням існуючих опенсоурс програм.
- Запитання “Чи можливе пряме керування ?”: треба для перевірки наявності мобільного застосунку, або веб-додатку для керування пристроями. Якщо відповідь “Так” і застосунок підходить для всіх пристроїв, то можна використовувати його, та не витрачати час і кошти на впровадження більш складної системи. Якщо “Ні”, то доведеться розглянути використання готових рішень.
- Запитання “Готове рішення задовольняє більшу частину потреб ?”: варто перевірити чи може готова система закрити більше ніж 90% потреб. Якщо відповідь “Так”, то варто купити та встановити готову систему. Якщо відповідь “Ні”, то переходимо до пункту з хмарними платформами.

- Запитання “Чи підходить хмарна платформа ?”: варіанти відповідей переходять або до розробки власної платформи, або ж буде обрана якась хмарна платформа.

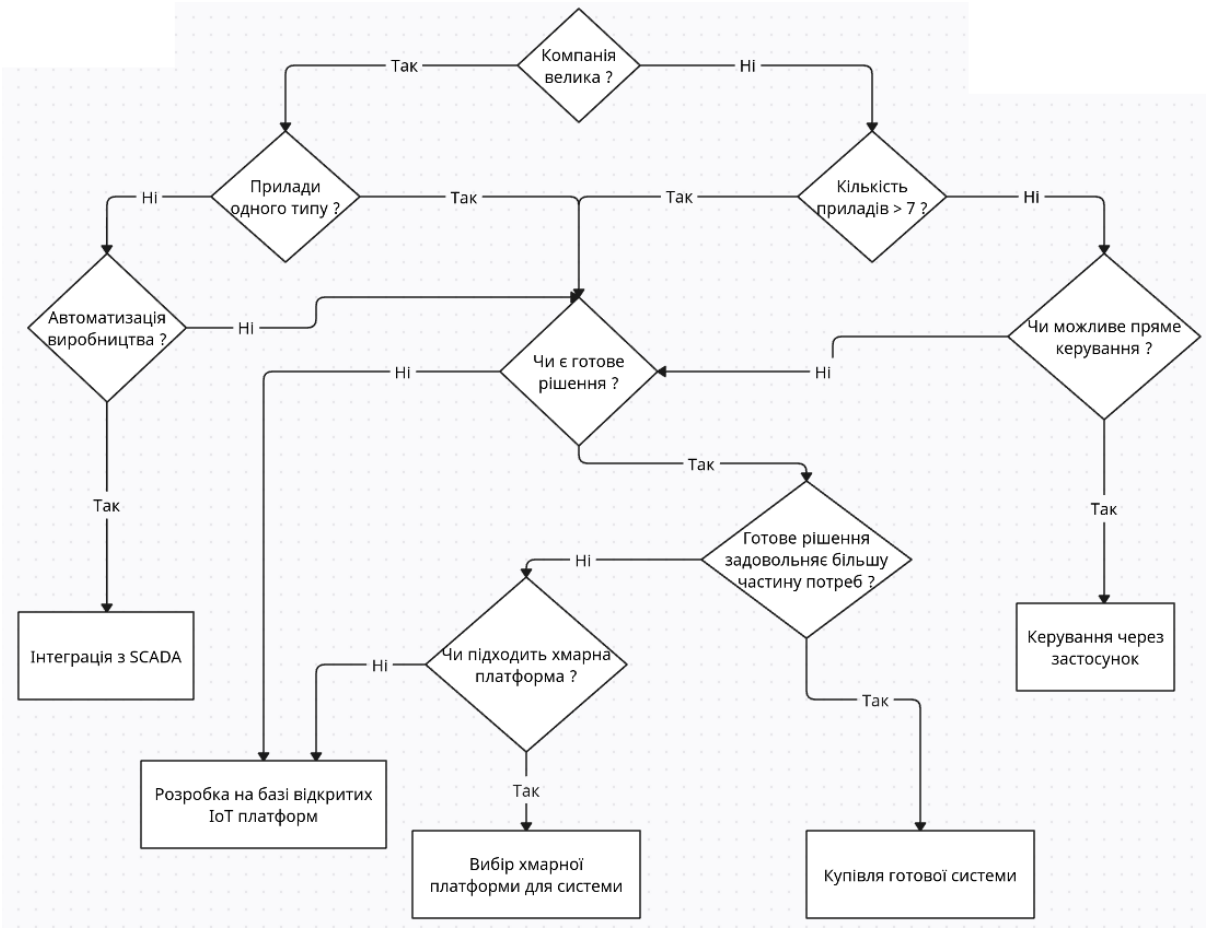

Рис. 3.1 – Алгоритм вибору системи

У підсумку можна визначити наступне, для маленьких IoT систем на кілька пристроїв можна використовувати пряме управління через застосунок. В інших випадках потрібна система, яка може працювати з більшою кількістю пристроїв. Для великих виробництв, яким треба промисловій автоматизації краще використовувати рішення типу SCADA. Розробка систем за базі готових платформах в певній мірі складніша, але дає більше перспектив та гнучкості.

Окремо варто виділити варіант який є найскладнішим - це повністю своя розробка. Його не було додано до алгоритму, оскільки такий підхід має

сенс лише для деяких державних компаній та амбітних проектів які мають достатньо фінансів для цього. При цьому для швидких демонстрацій та копіювання найкращих практик буде корисно паралельно зі своєю розробкою зробити ескізний проект на хмарній платформі.

4.2 Етапи впровадження системи IoT на виробництві

Завжди є великі очікування від проектів IoT. Організація може дозволити собі людські ресурси та технічні інструменти, але все одно може зазнати краху без належних кроків для впровадження систем IoT. Тому важливо знати, для чого потрібна реалізація IoT і як впроваджувати системи IoT. Для успішного впровадження системи моніторингу та керування IoT пристроями слід виконати наступні етапи:

1. Аналіз потреб виробництва, та як саме система моніторингу та керування IoT пристроями це вирішить. Це буде само IoT система, чи необхідна автоматизація. Чітке формулювання цілей є важливим етапом, адже від нього залежить правильність виконання всіх наступних етапів.
2. Перевірити як були інтегровані схожі рішення. З огляду на досвід інших компаній можна буде виділити якісь мінуси, які не було помітно на початку.
3. Оберіть який варіант системи вам підійде на основі блок-схеми алгоритму представленій у розділі 4.1. В залежності від варіанта можна обирати протокол, апаратне та програмне забезпечення. За результатами аналізу протоколів, якщо система не велика (до 7 пристроїв) і не планується масштабування, то можна обрати Wi-Fi або BLE. Якщо ж пристроїв більше, то вибір залежить від площі яку треба охопити, для більшості випадків найкращим варіантом буде Z-Wave.

4. Оберіть протокол для рівня додатків. В результаті аналізу у розділі 2.4 було визначено, що MQTT краще ніж інші протоколи, тому варто обрати саме його.
5. Виберіть потрібні інструменти IoT. З огляду на те який протокол буде використовуватись підберіть необхідні пристрої. Якщо планується використовувати власний сервер для обробки даних, то краще обирати Orange Pi 5. Проте якщо потрібно щось більш дешевше, то Raspberry Pi 5 є прийнятним варіантом для більшості задач.
6. Вибір програмної частини. Якщо було обрано хмарне рішення, або керування через застосунок, то цей етап можна пропустити, бо він вже виконаний. Якщо було обрано розробку на базі готових платформ, то треба обрати серед найбільш популярних програм.
7. Впровадження та створення прототипу. Якщо проводится власна розробка, перед впровадженням створіть прототип, щоб впевнитись що все працює як ви очікуєте. Якщо ж з якихось причин не виходить швидко підготувати прототип, то за можливістю на якийсь час варто розгорнути інфраструктуру на хмарній платформі.
8. Перевірка безпеки, підтримка та оновлення за необхідності Турбота про безпеку є важливим пунктом. Переконайтесь, що використовуються ефективні заходи безпеки, які можуть захистити прилади від хакерських атак. Важливо не нехтувати оновленнями для програмного забезпечення та періодично перевіряти, чи немає якоїсь підозрілої активності в системі.

Підсумовуючи, можна зазначити, що ефективне впровадження системи моніторингу та керування IoT пристроями на виробництві є багаторівневим процесом. Це вимагає ретельного планування та структурованого підходу, що в свою чергу гарантує можливість уникнути більшості потенційних проблем.

Висновки до розділу 4

Дані рекомендації визначають послідовний та структурований підхід до впровадження систем моніторингу та керування IoT пристроями на виробництві. Запропонований алгоритм вибору методу впровадження IoT-системи дозволяє більш точно та швидко вибрати кращу архітектуру та компоненти. Описані етапи забезпечують поетапне впровадження системи з мінімізацією ризиків та оптимізацією ресурсів.

ЗАГАЛЬНІ ВИСНОВКИ

1. Визначено основні безпровідні протоколи для обміну даними в IoT-системі. Проведено практичне порівняння протоколів HTTP та MQTT. В результаті MQTT був обраний як найбільш ефективний і легкий протокол для взаємодії з більшою частиною необхідних програм та хмарних платформ.
2. У процесі розгляду специфікації вимог до системи моніторингу та керування IoT пристроями на виробництві було створено структурну схему. Структурна схема може бути адаптована для роботи хмарною платформою.
3. У якості практичного застосування теорії було розроблено прототип IoT-системи для моніторингу якості повітря на виробництві. Демонстрація роботи системи підтвердила її функціональність, стабільність передачі даних та відповідність визначеним вимогам. Отже систему на основі цього прототипу можна впровадити повноцінну систему моніторингу якості повітря на виробництві.
4. Надано рекомендації для більш простого впровадження IoT-системи на виробництві малих підприємств. Запропоновано методику та алгоритм швидкої розробки та впровадження пропрієтарних IoT-систем на основі вихідних даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Weagro [Електронний ресурс]. - Режим доступу: https://weagro.com.ua/blog/internet-rechej-iot-shho-cze-ta-jogo-vykorystannya-v-silskomu-gospodarstvi/
2. Kyivstar Hub [Електронний ресурс]. - Режим доступу: https://hub.kyivstar.ua/articles/shho-take-iot-tehnologiya-ta-yak-vona-vplyvaye-na-rizni-galuzi
3. Termin [Електронний ресурс]. - Режим доступу: https://termin.in.ua/internet-rechey-iot/
4. Blog seo evolution [Електронний ресурс]. - Режим доступу: https://seo-evolution.com.ua/blog/poleznye-sovety/iot-v-biznesi-zastosuvannya-ta-perevagi
5. Blog imena [Електронний ресурс]. - Режим доступу: https://www.imena.ua/blog/internet-of-things/
6. Geek for geeks [Електронний ресурс]. - Режим доступу: https://www.geeksforgeeks.org/bluetooth/
7. Архітектура і технології IoT [Електронний ресурс]. - Режим доступу: https://learn.ztu.edu.ua/pluginfile.php/205968/mod_resource/content/2/%D0%9B%D0%B5%D0%BA_1.pdf
8. A Survey on Sensor-based Threats to Internet-of-Things (IoT) Devices and Applications [Електронний ресурс]. - Режим доступу: https://arxiv.org/pdf/1802.02041
9. Geek for geeks [Електронний ресурс]. - Режим доступу: https://www.geeksforgeeks.org/what-is-wi-fiwireless-fidelity/
10. The Internet of Things for Smart Manufacturing: [Електронний ресурс]. - Режим доступу: https://www.researchgate.net/publication/330408457_The_Internet_of_Things_for_Smart_Manufacturing_A_Review

11. E-server [Електронний ресурс]. - Режим доступу: https://e-server.com.ua/uk/poradi/merezi-iot-porivniannia-4-tipiv-i-prikladi-vikoristannia?srsltid=AfmBOopcg43UV6AUx0GAEuX6Abe63dHg7kLx7XzaBNcK7e7mazkuyYJq
12. Z-Wave документація [Електронний ресурс]. - Режим доступу: https://www.wccandm.services/pdf/page_producten/ZWave/Z-Wave%20Protocol%20Overview.pdf
13. Zigbee документація [Електронний ресурс]. - Режим доступу: https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/protocols/zigbee/index.html
14. Технологія LoRaWAN [Електронний ресурс]. - Режим доступу: https://deps.ua/ua/knowegable-base/reference-information/66634.html
15. Emnify [Електронний ресурс]. - Режим доступу: https://www.emnify.com/iot-glossary/guide-iot-protocols
16. AWS iot code docs [Електронний ресурс]. - Режим доступу: https://docs.aws.amazon.com/iot/
17. Microsoft Azure iot documentation [Електронний ресурс]. - Режим доступу: https://learn.microsoft.com/en-us/azure/iot/
18. Kaa iot docs [Електронний ресурс]. - Режим доступу: https://www.kaaiot.com/docs
19. Raspberry Pi docs [Електронний ресурс]. - Режим доступу: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html
20. Orange Pi 5 docs [Електронний ресурс]. - Режим доступу: http://www.orangepi.org/orangepiwiki/index.php/Orange_Pi_5
21. Banana Pi 5 docs [Електронний ресурс]. - Режим доступу: https://wiki.banana-pi.org/Banana_Pi_BPI-M5
22. Odroid C4 docs [Електронний ресурс]. - Режим доступу: https://wiki.odroid.com/odroid-c4/odroid-c4

23. NanoPi NEO4 docs [Електронний ресурс]. - Режим доступу: https://wiki.friendlyelec.com/wiki/index.php/NanoPi_NEO4
24. Emqx [Електронний ресурс]. - Режим доступу: https://www.emqx.com/en/blog/what-is-the-mqtt-protocol
25. HTTP protocol [Електронний ресурс]. - Режим доступу: https://www.javatpoint.com/computer-network-http
26. Tuya air sensor options [Електронний ресурс]. - Режим доступу: https://www.zigbee2mqtt.io/devices/TS0601_air_quality_sensor.html
27. Sonoff documentation [Електронний ресурс]. - Режим доступу: https://sonoff.tech/wp-content/uploads/2022/07/%E4%BA%A7%E5%93%81%E5%8F%82%E6%95%B0%E8%A1%A8-ZBDongle-E-20220714.pdf
28. Документація Raspberry Pi OS[Електронний ресурс]. - Режим доступу: https://www.raspberrypi.com/documentation/computers/getting-started.html
29. Docker forum [Електронний ресурс]. - Режим доступу: https://forums.docker.com/t/installation-steps-for-the-latest-raspberry-pi-os-64-bit/138838
30. Jpaul docker on raspberry pi 5[Електронний ресурс]. - Режим доступу: https://www.jpaul.me/2024/07/how-to-install-docker-on-a-raspberry-pi-5/
31. Geek for geeks Node-red [Електронний ресурс]. - Режим доступу: https://www.geeksforgeeks.org/node-red/
32. Node-red setup with Docker docs [Електронний ресурс]. - Режим доступу: https://nodered.org/docs/getting-started/docker
33. Mosquitto setup docs [Електронний ресурс]. - Режим доступу: https://hub.docker.com/_/eclipse-mosquitto
34. Zigbee2MQTT setup docs [Електронний ресурс]. - Режим доступу: https://www.zigbee2mqtt.io/guide/installation/02_docker.html

35. The Core Nodes [Електронний ресурс]. - Режим доступу: https://nodered.org/docs/user-guide/nodes
36. Aerostart [Електронний ресурс]. - Режим доступу: https://aerostar.ua/ua/news/novosti/co2-u-primischennjahl-na-scho-vplivae-dioksid-vuglecju-u-povitri-ta-jak-zmenshiti-jogo-riven.html
37. Iceoom [Електронний ресурс]. - Режим доступу: https://iceoom.com.ua/blog/formaldehyde_is_ubiquitous_and_dangerous_companion_of_a_modern_home/
38. Tecamgroup [Електронний ресурс]. - Режим доступу: https://tecamgroup.com/what-are-acceptable-voc-levels-in-the-air/