

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему КОМП'ЮТЕРНА СИСТЕМА ТОВ "ПРОДІНВЕСТ" З ВИКОРИСТАННЯМ
ТЕХНОЛОГІЙ ІНТЕРНЕТУ РЕЧЕЙ

Виконав: студент 2 курсу, групи КНТ-512м
спеціальності _____

123 Комп'ютерна інженерія

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні системи та мережі

КУЧЕРЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Керівник ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент МАЛИЙ О.Ю.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти магістерський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) «Комп'ютерні системи та мережі»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“24” жовтня 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

КУЧЕРЕНКА Олексія Олександровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Комп'ютерна система ТОВ «ПродІнвест» з використанням технології інтернету речей.

керівник проєкту (роботи) к. т. н., доцент, ГОЛУБ Тетяна Василівна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом вищого навчального закладу від “24” жовтня 2023 року № 400

2. Строк подання студентом проєкту (роботи) 10 грудня 2023 року

3. Вихідні дані до проєкту (роботи) перелік мікроконтролерів, технічні параметри ТОВ «ПродІнвест»

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області;

2. Проєктування системи;

3. Розробка комп'ютерної системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ГОЛУБ Т.В., доцент		
нормоконтроль	ПОЛЬСЬКА О.В., ст. викл.		

7. Дата видачі завдання 01.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз наявних методів та алгоритмів використання інтернету речей	10.10.2023 р.	
2	Розробка вимог для системи	15.10.2023 р.	
3	Аналіз програмних середовищ	20.10.2023 р.	
4	Розробка схеми системи	05.10.2023 р.	
5	Розробка комп'ютерної системи	20.11.2023 р.	
6	Дослідження моделі інтелектуальної системи	25.11.2023 р.	
7	Оформлення отриманих результатів у ПЗ	02.11.2023 р.	
8	Оформлення допоміжного матеріалу	10.12.2023 р.	

Студент


(підпис) Олексій КУЧЕРЕНКО
(ім'я, ПРИЗВИЩЕ)

Керівник проєкту (роботи)


(підпис) Тетяна ГОЛУБ
(ім'я, ПРИЗВИЩЕ)

РЕФЕРАТ

ПЗ: 58 с., 38 рис., 12 джерел.

ІНТЕРНЕТ РЕЧЕЙ, МОНІТОРИНГ, УПРАВЛІННЯ, ОПТИМІЗАЦІЯ, АНАЛІТИКА, АВТОМАТИЗАЦІЯ, БЕЗПЕКА, АДАПТАЦІЯ, ЕФЕКТИВНІСТЬ.

Об'єкт дослідження - система моніторингу та керування елементами IoT.

Предмет - засоби передачі даних та управління в IoT.

Методи дослідження. Використовуються теоретико-емпіричний, системний аналіз та проектування комп'ютерних мереж для розробки системи.

Наукова новизна:

- порівняльний аналіз сучасних протоколів та засобів IoT дозволив виявити оптимальні для впровадження в систему ТОВ "ПродІнвест";
- визначено, що одноплатні комп'ютери найкраще підходять для систем моніторингу та управління IoT;
- за допомогою інженерії програмного забезпечення розроблено систему моніторингу та управління.

Практичне значення: Налаштовано обладнання та створено програмне забезпечення для ефективного моніторингу та управління IoT в рамках потреб ТОВ "ПродІнвест".

ABSTRACT

Explanatory note to the master's work: 58 p., 38 figures, 12 sources.

INTERNET OF THINGS, MONITORING, MANAGEMENT, OPTIMIZATION, ANALYTICS, AUTOMATION, SECURITY, ADAPTATION, EFFICIENCY.

The object of research is a system for monitoring and controlling IoT elements.

The subject is means of data transmission and management in IoT.

Research methods. Theoretical-empirical, system analysis and design of computer networks are used for system development.

Scientific novelty:

- a comparative analysis of modern IoT protocols and tools made it possible to identify the optimal ones for implementation in the ProdInvest LLC system;
- it is determined that single-board computers are best suited for IoT monitoring and control systems;
- a monitoring and management system was developed with the help of software engineering.

Practical significance: The equipment has been configured and the software has been created for effective monitoring and management of IoT within the needs of ProdInvest LLC.

ЗМІСТ

Скорочення та умовні позначки	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Системи Інтернету речей.....	10
1.2 Паттерни проєктування Інтернету речей.....	12
1.3 Дослідження апаратної складової IoT.....	15
1.4 Протоколи IoT	21
1.5 Висновки до розділу 1	30
2 Проєктування системи.....	30
2.1 Програмні середовища для роботи IoT.....	30
2.2 Мікропроцесор системи.....	32
2.3 Зовнішні модулі.....	36
2.4 Висновки до розділу 2	44
3 Розробка комп'ютерної системи.....	44
3.1 Структура розподіленої системи	44
3.2 Розробка системи	46
3.3 Використання комп'ютерної системи.....	51
3.4 Висновки до розділу 3	54
Висновки	56
Перелік джерел посилання	57

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API	- Application Programming Interface;
HTTP	- Hypertext Transfer Protocol;
IoT	- Internet of Things;
JSON	- JavaScript Object Notation;
JVM	- Java Virtual Machine;
MQTT	- Message Queuing Telemetry Transport;
REST	- Representational State Transfer;
SQL	- Structured Query Language;
TCP/IP	- Transmission Control Protocol/Internet Protocol;
UI	- User Interface.

ВСТУП

У епоху цифрової трансформації, використання інноваційних технологій у корпоративному середовищі стає не просто трендом, а необхідністю. Однією з таких технологій, що відіграє ключову роль у модернізації робочих процесів, є Інтернет речей (IoT). Дана робота присвячена створенню комп'ютерної системи для ТОВ "Продінвест" з використанням Інтернету речей, зосереджуючись на поліпшенні умов праці співробітників та оптимізації управління та контролю умов роботи в офісі.

Актуальність розробки полягає у впровадженні IoT-технологій, що дозволяють створити більш комфортні, безпечні та продуктивні умови праці. Застосування Інтернету речей у корпоративному середовищі відкриває широкі можливості для моніторингу та керування робочим простором, оптимізації споживання енергоресурсів, а також підвищення загального добробуту та задоволеності співробітників. Це сприяє не тільки покращенню продуктивності, але й позитивно впливає на корпоративну культуру та залученість персоналу.

Сучасні тенденції в сфері використання Інтернету речей в офісному середовищі включають автоматизацію та інтелектуалізацію управління офісними ресурсами, використання даних для аналітики та прийняття обґрунтованих рішень, а також забезпечення високого рівня безпеки та приватності даних. Ключовим аспектом є також інтеграція IoT-рішень з іншими корпоративними системами та платформами, що забезпечує гнучкість та масштабованість управління. Таким чином, інтеграція Інтернету речей в корпоративну інфраструктуру ТОВ "Продінвест" стає не лише актуальним кроком для підвищення ефективності бізнесу, але й важливим елементом стратегії цифрової трансформації компанії.

Актуальність дослідження. Концепція Інтернету речей (IoT) представляє собою глобальну мережу об'єктів з унікальними ідентифікаторами, здатних взаємодіяти один з одним автономно. Вже мільярди пристроїв по всьому світу

взаємодіють через Інтернет, формуючи складні системи, де датчики і пристрої обмінюються даними, проводять аналітику та автоматично управляють процесами без безпосередньої участі людини.

Ціль та завдання дослідження. Мета полягає у створенні та впровадженні системи моніторингу та управління для елементів IoT, відповідної потребам ТОВ "ПродІнвест". Для цього необхідно:

Аналізувати існуючі системи управління IoT;

- дослідити архітектуру, технології та протоколи IoT;
- вивчити та налаштувати апаратні ресурси;
- розробити програмне забезпечення для ефективного управління IoT;
- оцінити якість розробленого програмного забезпечення.

Об'єкт дослідження - система моніторингу та керування елементами IoT.

Предмет - засоби передачі даних та управління в IoT.

Методи дослідження. Використовуються теоретико-емпіричний, системний аналіз та проектування комп'ютерних мереж для розробки системи.

Наукова новизна:

- порівняльний аналіз сучасних протоколів та засобів IoT дозволив виявити оптимальні для впровадження в систему ТОВ "ПродІнвест";
- визначено, що одноплатні комп'ютери найкраще підходять для систем моніторингу та управління IoT;
- за допомогою інженерії програмного забезпечення розроблено систему моніторингу та управління.

Практичне значення: Налаштовано обладнання та створено програмне забезпечення для ефективного моніторингу та управління IoT в рамках потреб ТОВ "ПродІнвест".

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Системи Інтернету речей

Екосистема Інтернету речей (IoT) — це складне та багатогранне середовище, що охоплює широкий спектр інструментів, послуг і технологій, які працюють у гармонії, щоб забезпечити можливість комунікації та взаємодії розумних підключених пристроїв. За своєю суттю ця екосистема базується на апаратних компонентах – фізичних пристроях і датчиках, які збирають дані з навколишнього середовища. Ці пристрої можуть варіюватися від простих датчиків температури до складних промислових машин, кожна з яких відіграє вирішальну роль у зборі даних у реальному часі. Це апаратне забезпечення є основою екосистеми IoT, слугуючи очима та вухами системи, збираючи життєво важливу інформацію, яка живить усі подальші взаємодії та рішення.

Виходячи за межі матеріального обладнання, екосистема IoT значною мірою покладається на технології підключення та мереж, які дозволяють цим пристроям передавати свої дані. Це включає різноманітні протоколи зв'язку, мережі та стандарти, які забезпечують безперебійну та ефективну передачу даних між пристроями та центральними системами обробки. Через Wi-Fi, Bluetooth, стільникові мережі чи інші бездротові технології, що розвиваються, цей аспект екосистеми гарантує, що дані, зібрані обладнанням, не залишатимуться ізольованими. Натомість він стає частиною більшої взаємопов'язаної системи, де можна використати його справжню цінність. Цей зв'язок має життєво важливе значення, оскільки він усуває розрив між фізичним і цифровим світом, дозволяючи пристроям ділитися, аналізувати та діяти на основі зібраних даних.

Програмне забезпечення та платформи, які обробляють, аналізують і керують даними, утворюють останній, важливий компонент екосистеми IoT. Це включає в себе все: від платформ хмарних обчислень, які зберігають величезні обсяги даних, до аналітичного програмного забезпечення, яке обробляє ці дані,

витягаючи значущі ідеї та шаблони. Удосконалені алгоритми та моделі машинного навчання відіграють тут важливу роль, перетворюючи необроблені дані на практичну інформацію. На цьому рівні програмного забезпечення дані перетворюються на рішення, забезпечуючи автоматичні відповіді, оптимізуючи процеси та надаючи користувачам інформацію та контроль над своїми системами IoT. Саме цей рівень дійсно розкриває потенціал IoT, перетворюючи мережу підключених пристроїв на інтелектуальну систему, що швидко реагує, здатну адаптуватися та розвиватися відповідно до вимог середовища.

Загалом, компоненти інтернету речей можна розділити на чотири основні категорії: сервіси, апаратне забезпечення, правила та програмне забезпечення. Кожну з цих категорій можна далі поділити на підгрупи за функціональним призначенням (рис. 1.1).



Рисунок 1.1 – Система інтернету речей

Окрім цих базових рівнів, екосистема IoT також значно збагачена інтеграцією передових технологій, таких як штучний інтелект (AI) і периферійні обчислення. AI вводить елемент прогнозової аналітики та

можливості прийняття рішень, що дозволяє системам IoT не лише реагувати на поточні дані, але й передбачати майбутні тенденції та поведінку. Ця інтеграція особливо важлива в додатках, які вимагають прийняття рішень у режимі реального часу, таких як автономні транспортні засоби або інфраструктура розумного міста. З іншого боку, Edge Computing децентралізує обробку даних, дозволяючи пришвидшити час відповіді за рахунок обробки даних ближче до місця їх створення. Це особливо важливо в сценаріях, коли передача даних у центральну хмару для обробки займе надто багато часу чи пропускну здатності. Ці передові технології, працюючи в тандемі з основними компонентами екосистеми IoT, підвищують її ефективність, швидкість реагування та адаптивність, роблячи рішення IoT більш надійними, масштабованими та придатними для широкого спектру застосувань, від промислової автоматизації до персоналізованих рішень для охорони здоров'я.

1.2 Паттерни проєктування Інтернету речей

Структура Інтернету речей може бути різноманітною, залежно від її призначення та способу впровадження. Проте, вона зазвичай має багато спільного з традиційними автоматизованими системами управління технологічними процесами (АСУТП), як зображено на схемі 1.2. Взаємодія з оточенням здійснюється через датчики та виконавчі пристрої, подібно до того, як це відбувається в АСУТП для управління різними об'єктами. Ці елементи, разом із повною інфраструктурою, яка інтегрує їх з обробкою подій через мережу Інтернет, створюють так звану "граничну зону" (Edge).

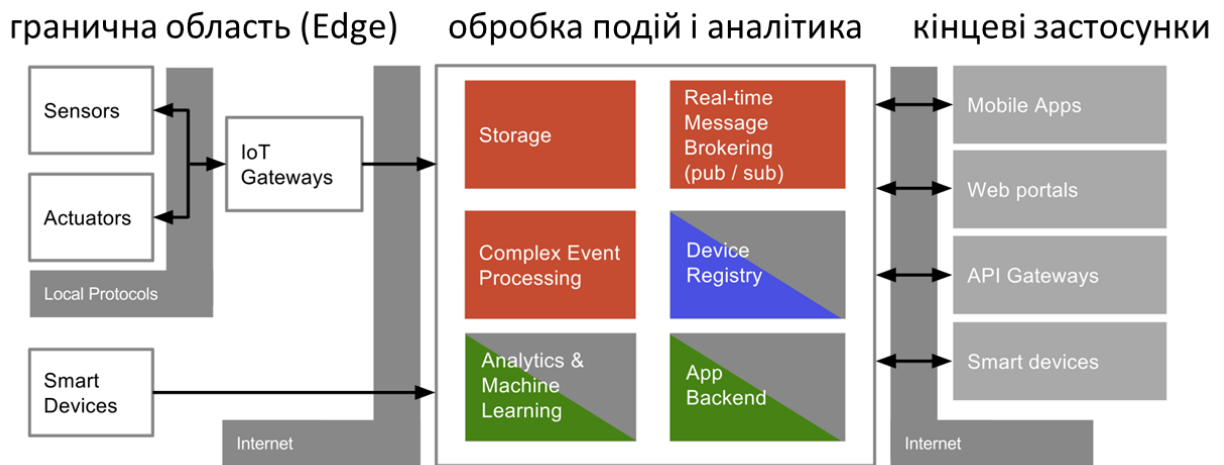


Рисунок 1.2 – Архітектура IoT

В критичній зоні отримані дані та події збираються та обробляються відповідно до заздалегідь встановленої мети на етапі обробки подій та аналітики (event processing, Platform). Цей процес відіграє ключову роль у ефективному функціонуванні систем Інтернету речей (IoT). На цій стадії відбувається зберігання (storage), обробка подій (Event Processing) та пересилання даних до потрібних додатків (Real-Time Message Brokering, Stream Processing). Окрім цього, здійснюється адміністрування та керування пристроями з критичної зони (Device Registry, Edge Device Management). На основі оброблених даних виконується машинне навчання (Machine Learning), яке використовує аналітичні сервіси (Analytics) для отримання висновків щодо об'єкту. Це зазвичай виконується за допомогою хмарних (Cloud) або туманних (Fog) обчислень. Процес можна порівняти з рівнем контролерів та систем SCADA у традиційних АСУТП, але без НМІ функцій. Контроль, дистанційне управління та адміністрування системи здійснюються через кінцеві додатки за допомогою Інтернету.

На рисунку 1.3 демонструється архітектура, аналогічна описаній, але в контексті сервісів. Вона включає зону Edge з датчиками та Device Hub/Gateway (збір і маршрутизація даних) і Device Management (управління пристроями). Ці компоненти реалізовані як через хмарні обчислення, так і на граничних пристроях. Функції зберігання та первинної обробки даних об'єднані в Data

Management. Всі інші процеси обробки, включаючи аналітику, представлені у вигляді PaaS-додатків, які взаємодіють з сервісами управління даними через API (Application Program Interface).

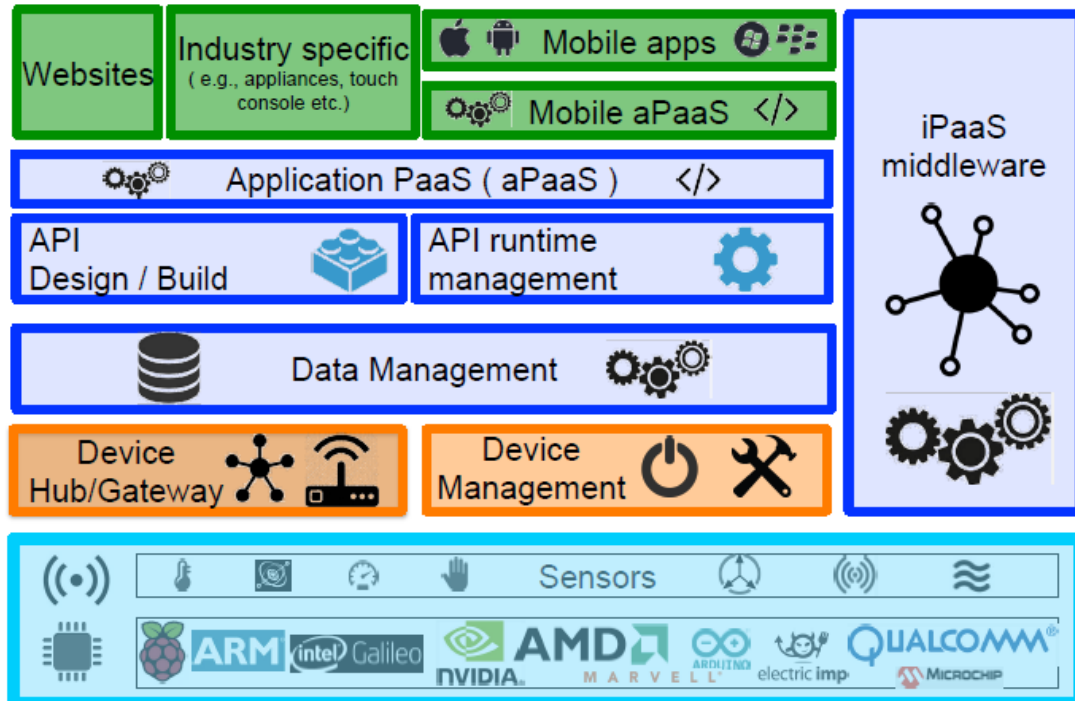


Рисунок 1.3 – Сервіси IoT

Рисунок 1.4 ілюструє альтернативний підхід до структуризації архітектури Інтернету речей, зосереджуючись на модульній організації. В цій архітектурі, різні функціональні елементи розподілені в модулі, що покращує гнучкість та масштабованість системи. Три основні рівні включають рівень граничних пристроїв (Edge Devices), центральний рівень обробки даних (Central Data Processing) та рівень інтеграції та управління (Integration and Management Layer). Кожен з цих рівнів виконує важливі функції, які в сукупності сприяють ефективній роботі всієї системи. Граничні пристрої збирають та передають дані, центральний рівень забезпечує обробку та аналіз даних, тоді як рівень інтеграції займається управлінням системою та забезпечує зв'язок з зовнішніми сервісами та додатками.

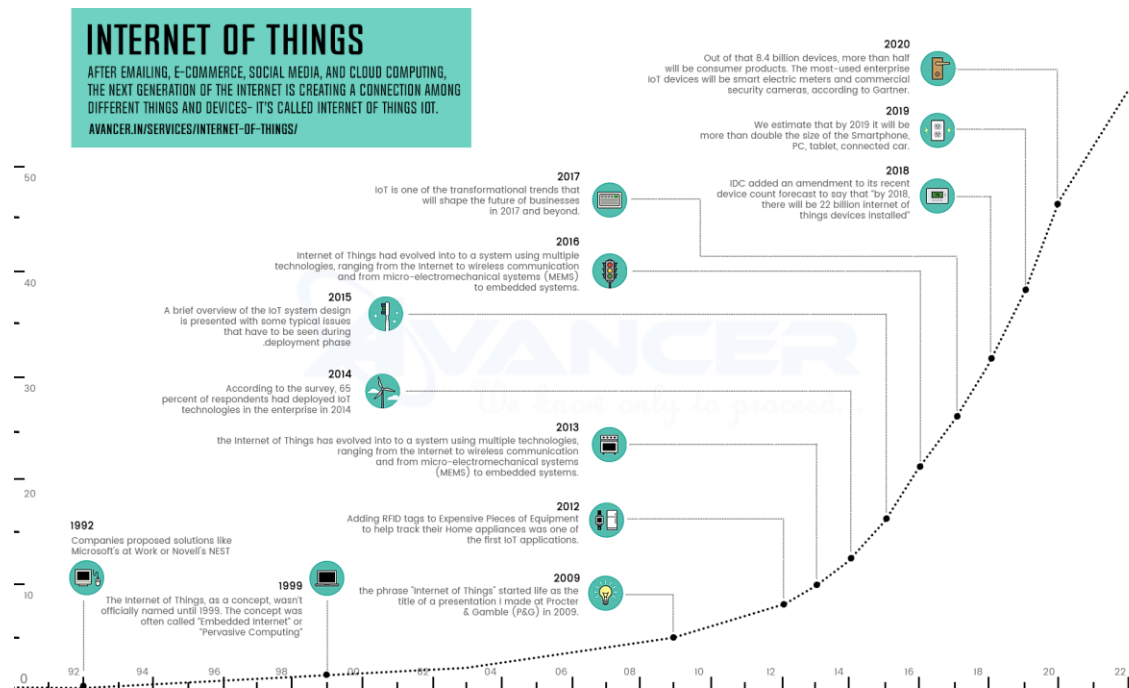


Рисунок 1.4 – Розвиток архітектури IoT

В рамках цієї модульної архітектури, хмарні обчислення та Інтернет відіграють роль інтеграційного рівня, забезпечуючи високу ступінь взаємодії та координації між різними компонентами системи. Хмарні платформи не лише дозволяють ефективно зберігати та обробляти великі обсяги даних, але й надають доступ до потужних аналітичних інструментів та сервісів управління. Це забезпечує швидке та ефективне реагування на зміни в даних, а також дозволяє автоматизувати багато процесів, підвищуючи загальну продуктивність системи. Таким чином, хмарні обчислення виступають як фундаментальний елемент архітектури Інтернету речей, забезпечуючи її масштабованість та гнучкість.

1.3 Дослідження апаратної складової IoT

Сенсори і енергія. Інтернет речей ініціюється або завершується з події: будь-то просте переміщення, коливання температури, чи навіть перемикання

важеля. На відмінну від стандартних ІТ-пристроїв, Інтернет речей переважно асоційований з фізичною активністю або зміною стану. Він реагує на зовнішні стимули з реального світу. При цьому, інколи навіть одиничний сенсор може генерувати значну кількість даних, як, наприклад, звуковий сенсор для діагностики обладнання. В інших ситуаціях, один біт інформації може нести ключову інформацію про стан пацієнта (рис. 1.5). Незалежно від контексту, системи сенсорів розвиваються, стаючи меншими за розмірами та більш доступними за ціною, відповідно до закону Мура. Саме цей прогрес спонукає прогнози про підключення мільярдів пристроїв до Інтернету речей, і саме тому ці прогнози мають всі шанси стати реальністю.

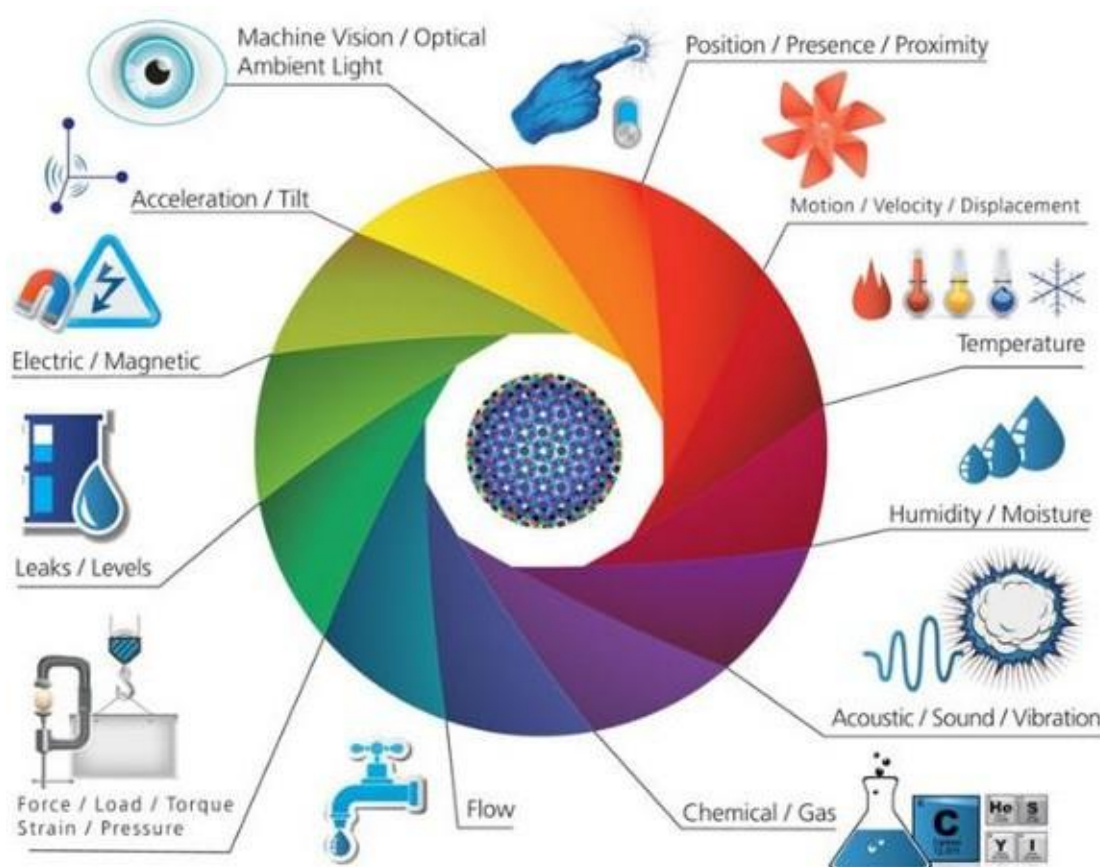


Рисунок 1.5 – Системи сенсорів та датчиків IoT

Оцінюючи сферу Інтернету речей, критично важливо враховувати роль мікроелектромеханічних систем, різноманітних датчиків та інших доступних периферійних пристроїв, а також їх електрофізичні властивості. Ключовим

аспектом також є системи живлення, які забезпечують енергією ці пристрої. Недооцінювати потребу в енергії для численних малогабаритних датчиків було б помилкою, оскільки вони вимагають значного обсягу енергії для своєї роботи. Це також має прямий зв'язок із розгортанням хмарних сервісів IoT.

Питання передачі даних є вирішальним у розробці Інтернету речей, з особливим акцентом на створення з'єднань та функціональності мереж. Інтернет речей не був би можливим без ефективних технологій передачі даних, які здатні транслювати інформацію з віддалених чи складних умов до головних центрів обробки даних великих компаній, таких як Google, Amazon, Microsoft та IBM. Оскільки термін "Інтернет речей" включає слово "Інтернет", вивчення мережевих технологій, обміну даними та навіть теорії сигналів є фундаментальним. Основу Інтернету речей становлять не стільки сенсори чи програмне забезпечення, скільки здатність встановлювати з'єднання.

Передача даних та створення мережевих з'єднань базуються на системах комунікації з коротким радіусом дії, таких як персональні мережі (PAN), які можуть бути як дротовими, так і бездротовими, і часто функціонують без використання IP-протоколів. В контексті бездротових IoT-мереж та протоколів зазвичай виділяють такі технології, як Bluetooth, mesh-мережі, Zigbee, Z-Wave, а в області промислового Інтернету речей (IIoT) використовуються також Wireless HART та ISA100.

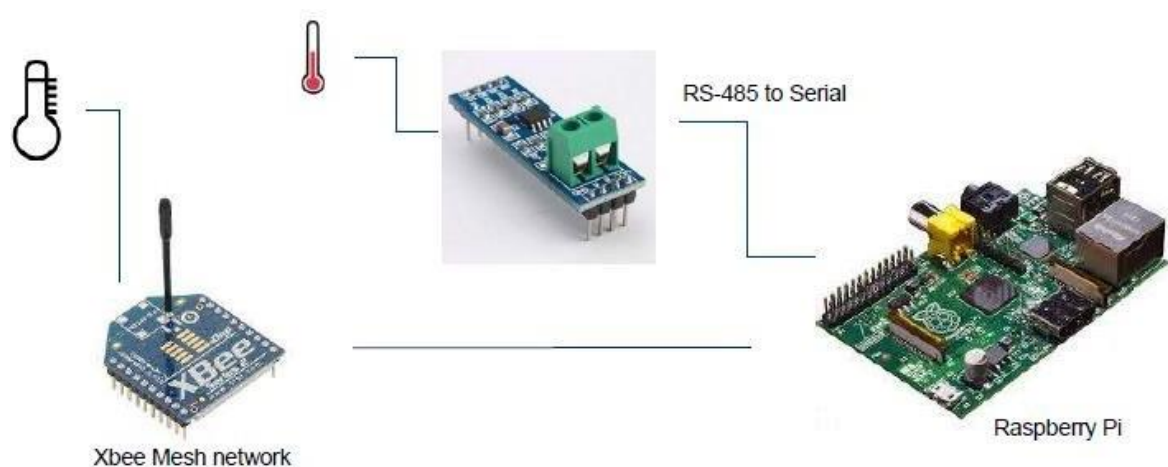


Рисунок 1.6 – Приклад підсистеми IoT

Крім PAN, також застосовують бездротові локальні мережі та системи зв'язку на основі IP-протоколів, включно з різноманітними Wi-Fi-мережами за стандартами IEEE 802.11, 6LoWPAN та технологією Thread. Часто використовуються телекомунікації на основі стільникових стандартів (3G, 4G LTE) і нові стандарти, спрямовані на підтримку роботи Інтернету речей (рис 1.7).

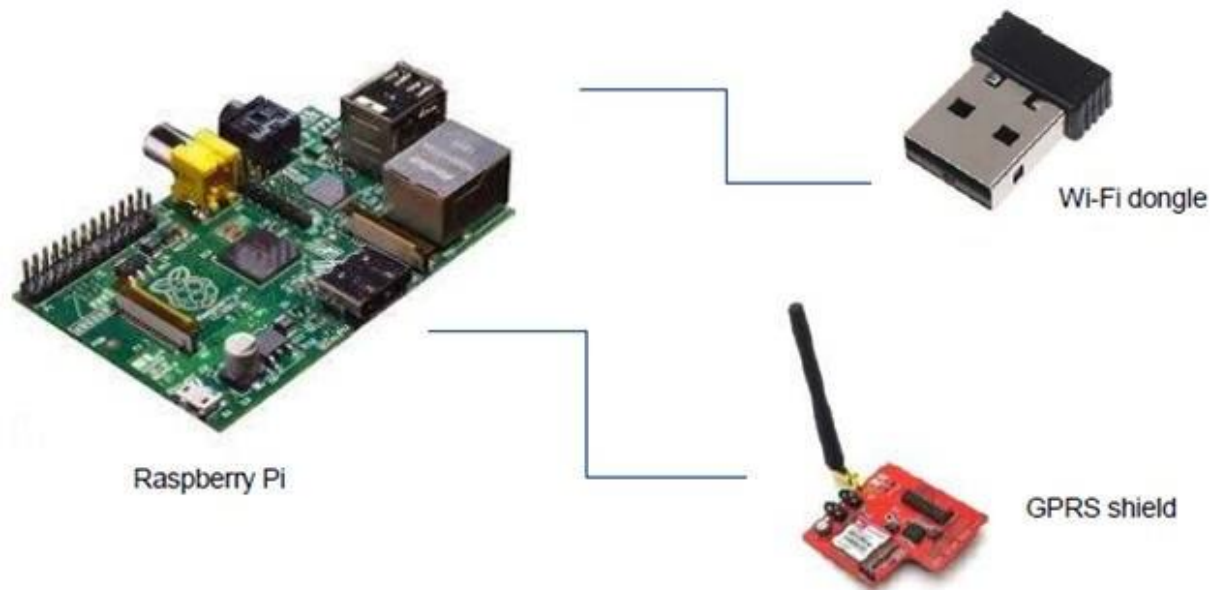


Рисунок 1.7 – Пристрої для підключення до мережі

Для передачі даних від сенсорів до Інтернет-простору використовуються дві ключові технології: маршрутизатор-шлюз та основні інтернет-протоколи для ефективного обміну даними. Маршрутизатор відіграє критичну роль у таких аспектах як безпека, керування та напрямлення даних. Граничні маршрутизатори (Edge routers) контролюють і моніторять стан відповідних mesh-мереж, забезпечуючи вирівнювання та підтримку якості даних. Вони також відіграють важливу роль у захисті конфіденційності та безпеки даних. Маршрутизатори є незамінними в створенні віртуальних приватних мереж, локальних мереж та програмно-визначених глобальних мереж, які можуть обслуговувати тисячі вузлів під керівництвом одного граничного маршрутизатора, що діє як місток до хмари.

На цьому рівні використовується ряд протоколів, необхідних для обміну даними між вузлами, маршрутизаторами та хмарними сервісами в межах IoT-системи. Інтернет речей сприяв розвитку нових IoT-протоколів, які стають нарівні з традиційними протоколами HTTP і SNMP, які використовуються вже кілька десятиліть. Для передачі IoT-даних потрібні ефективні, енергозберігаючі протоколи з мінімальною затримкою, здатні безпечно та ефективно пересилати дані до хмари та назад. Серед таких протоколів виділяють MQTT, AMPQ і CoAP.

Туманні і граничні обчислення, аналітика і машинне навчання. На цьому етапі розглядається питання щодо обробки потоку даних, що надходять у хмарний сервіс від граничного вузла. Важливо розуміти всі аспекти архітектури хмарних систем та вплив затримок на IoT-систему, щоб правильно оцінити, як система буде розвиватися і рости. Також не всі дані потрібно надсилати до хмари. Передача всіх IoT-даних часто виявляється значно дорожчою, ніж їх обробка на межі мережі (граничні обчислення, Edge Computing) або включення граничного маршрутизатора в зону обслуговування хмарного сервісу (туманні обчислення, Fog computing). Туманні обчислення також стандартизуються, зокрема, існує стандарт туманних обчислень, як наприклад, архітектура OpenFog (рис. 1.8).

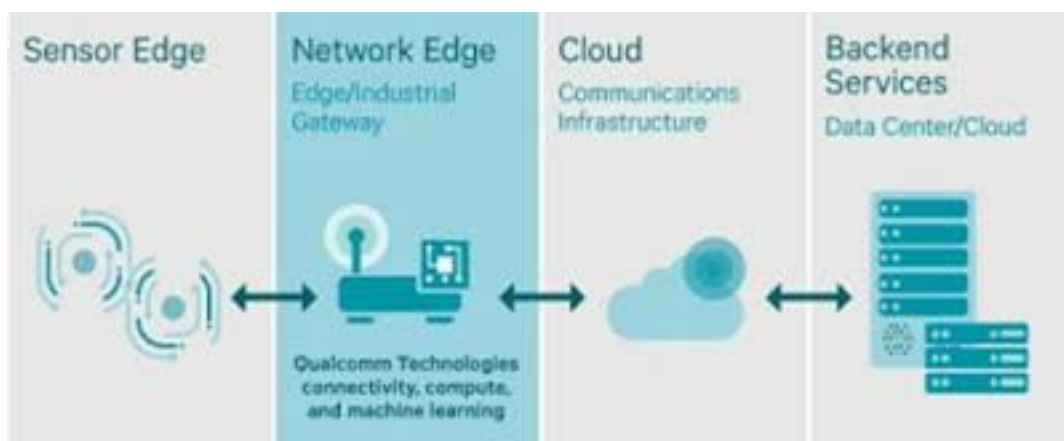


Рисунок 1.8 – Процес обчислень в IoT

Коли дані, перетворені з фізичних вимірювань в цифровий формат,

набувають значного обсягу, настає час для втручання аналітичних інструментів та процесорів правил в рамках IoT-систем. Рівень складності імплементації такої системи залежить від планованих застосувань. У деяких ситуаціях, реалізація може бути відносно простою, як у випадку з базовим правилочим процесором, встановленим на периферійному маршрутизаторі для керування невеликою кількістю датчиків, які відслідковують нестандартні коливання температури. Проте, в інших варіантах, може виникати потреба у передачі великої кількості структурованих та неструктурованих даних до хмарної інфраструктури.

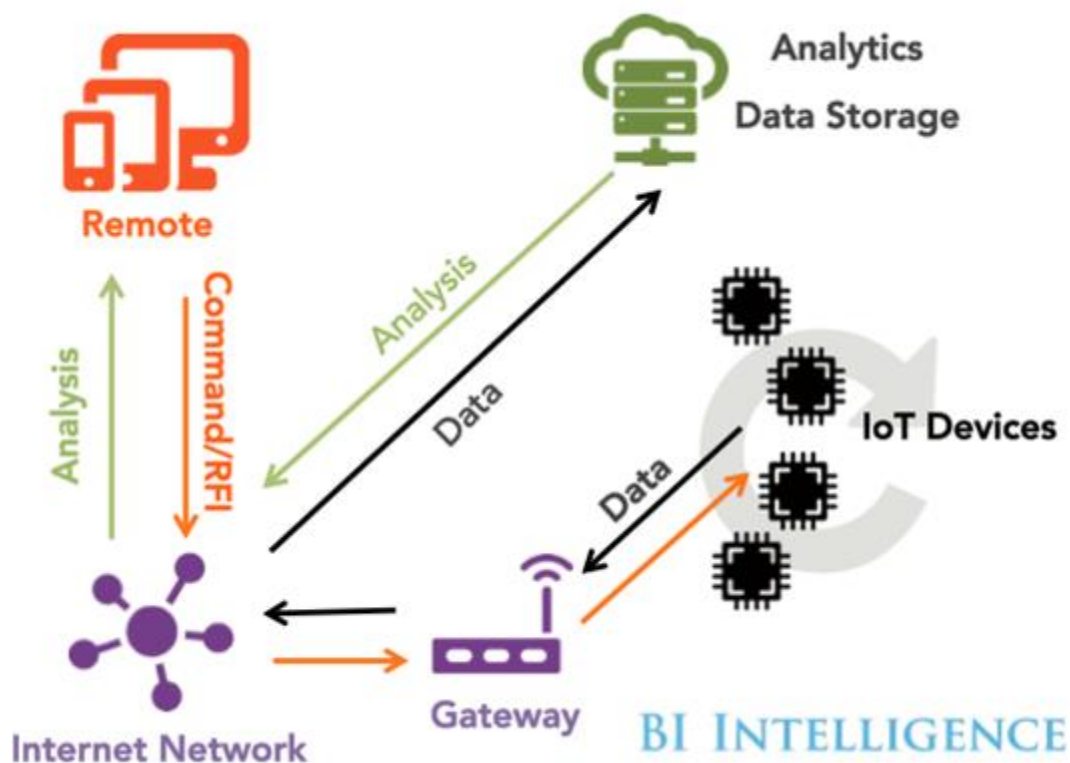


Рисунок 1.9 – Приклад обміну інформацією в IoT

У таких сценаріях, ефективне управління даними стає ключовим для успішного впровадження IoT-рішень. Передача даних в хмару для подальшої обробки та аналізу дозволяє використовувати більш потужні обчислювальні ресурси, що забезпечує більш глибокий аналіз та розуміння отриманої інформації. Це дозволяє не лише виявляти та реагувати на поточні умови, але й

прогнозувати майбутні тренди та поведінку системи. Такий підхід підкреслює важливість інтеграції IoT-систем з хмарними технологіями та аналітичними інструментами, що разом створюють більш гнучку, масштабовану та інтелектуальну екосистему.

1.4 Протоколи IoT

Технології та протоколи для короткодіапазонного зв'язку. Враховуючи концепцію Інтернету речей, де пристрої часто розташовані близько один до одного і можуть не мати постійного джерела енергії, існує потреба у спеціалізованих технологіях та протоколах для передачі даних. Ось деякі із зараз популярних технологій:

- Z-Wave - це радіотехнологія, що працює на частотах до 1 ГГц, ідеально підходить для передачі простих команд з мінімальними затримками;
- NFC - технологія, яка дозволяє передавати дані на відстані до 20 см;
- RFID - радіочастотна ідентифікація для зчитування інформації;
- BLE - варіація Bluetooth з оптимізованим енергоспоживанням;
- Wi-Fi HaLow - працює на частоті 900 МГц, має енергоспоживання порівнянне з Bluetooth, але забезпечує вищу швидкість передачі даних.

Далі ми розглянемо ці технології детальніше та з'ясуємо, як вони використовуються.

Z-Wave – це бездротова радіотехнологія, яка має низьке енергоспоживання і працює на частотах до 1 ГГц. Ця технологія оптимізована для передачі простих керуючих команд з мінімальними затримками, наприклад переключення каналів або вимкнення приладів. Вибір цих частот обумовлений прагненням уникнути перешкод від інших технологій, таких як Wi-Fi, які використовують частоти 2,4 ГГц і 5 ГГц. Незважаючи на свою складність, Z-Wave є простою в користуванні, енергозберігаючою та часозберігаючою

системою. Ця система, що працює на радіохвилях низької потужності, забезпечує майже 100% надійність, оскільки радіохвилі можуть проникати через стіни, поверхи та меблі, охоплюючи всі області будинку.

Near Field Communication (NFC) – це технологія для зв'язку на коротких відстанях, призначена для обміну різноманітною інформацією, включаючи зображення, музику, контактні дані або ключі цифрової авторизації між пристроями, що підтримують NFC, і розташованими поруч. Це можуть бути смарт-картки, портативні пристрої або зчитувачі RFID. NFC може бути використана як ключ доступу до даних або послуг. Технологія дозволяє передавати дані не тільки від активного до пасивного пристрою, але й між двома активними пристроями. NFC також взаємодіє з пристроями RFID. Для забезпечення сумісності між RFID картами та мобільними телефонами різних виробників проводяться тести цифрових протоколів та вимірювання ключових характеристик радіосигналу, таких як часові параметри, чутливість та амплітуда приймача в активному режимі, частота несучої та амплітуда сигналу.

При передачі від активного до пасивного пристрою використовується амплітудна маніпуляція ASK. При взаємодії двох активних пристроїв, обидва є рівноправними учасниками. Кожен пристрій має своє джерело енергії, тому сигнал несучої вимикається одразу після завершення передачі даних. В результаті індуктивного зв'язку між запитуючим та прослуховуючим пристроями, пасивний пристрій впливає на активний, змінюючи опір і спричиняючи зміни амплітуди або фази напруги на антені запитуючого пристрою. Це веде до з'єднання та передачі даних між пристроями, яка припиняється, якщо відстань між ними перевищує 20 см, розриваючи електромагнітний зв'язок.

NFC можна розглядати як розвиток відомої технології RFID. RFID застосовується в безконтактних картках та мітках. На відміну від RFID, NFC не тільки читає інформацію з пасивних електронних міток, але й забезпечує двосторонній бездротовий зв'язок між пристроями.

Ідентифікація радіочастотою (RFID) - це технологія для автоматизованої

ідентифікації предметів, що використовує радіохвилі для читання або запису інформації, яка зберігається в транспондерах, відомих як RFID-теги [19]. Структура RFID-системи включає пристрій для зчитування (читач) та транспондер (також названий RFID-тегом) [5].

RFID-теги зазвичай складаються з двох елементів: мікросхеми для обробки та зберігання даних, модуляції та демодуляції радіосигналів, а також інших функцій, і антени для передачі та прийому сигналів. Також потрібне програмне забезпечення для збору та аналізу інформації з RFID-тегів.

Існують два основних типи тегів: активні та пасивні:

- активні теги оснащені власним джерелом енергії, що дозволяє їм самостійно відправляти сигнали та бути зчитаними з великої відстані;
- пасивні теги працюють без власного джерела живлення, активуючись лише при отриманні сигналу від зчитувача, після чого передають збережені дані.

RFID-теги (рис. 1.10) можуть використовуватися для керування запасами або відстеження часу на спортивних заходах [5]. Вони не замінюють, а доповнюють штрих-коди, надаючи можливість дистанційного читання. Також використовуються для маркування худоби з метою реєстрації ветеринарних перевірок. У транспортній сфері допомагають ідентифікувати транспортні засоби, що рухаються на високій швидкості. Деякі авіакомпанії використовують їх для відстеження багажу. RFID також інтегровані в біометричні паспорти та кредитні картки для безпеки доступу.



Рисунок 1.10 - RFID-теги

BLE, Bluetooth Low Energy. Бездротовий протокол Bluetooth Low Energy (BLE) - це частина стандарту Bluetooth, яка була започаткована з версії Bluetooth 4.0 і продовжується у Bluetooth 5.0 [6].

BLE-пристрої характеризуються низьким споживанням енергії порівняно з попередніми поколіннями Bluetooth. Багато з них можуть функціонувати понад рік на одній маленькій батареї без необхідності перезарядки. Це робить можливим використання малогабаритних датчиків, які постійно працюють та взаємодіють з іншими пристроями.

BLE призначено для невеликих за розміром пристроїв, де важлива компактність і неможливість встановлення великих акумуляторів [6].

Bluetooth LE витрачає на 10-20 разів менше енергії, ніж традиційні Bluetooth рішення, здатний передавати дані в 50 разів швидше і на відстань більше 100 метрів. Він також відзначається високою безпекою, надійністю, малим часом затримки підключення та низькою споживчою потужністю.

Серед інших особливостей BLE - адаптивність в перенастроюванні частот, що забезпечує захист від помилок у передачі сигналів і дозволяє швидко змінювати робочу частоту для уникнення перешкод.

Специфікація Bluetooth 5.0 була розроблена з акцентом на Інтернет речей, демонструючи прагнення стандарту зайняти значну частку ринку. Вона значно швидша за Bluetooth 4.0, маючи при цьому аналогічний рівень споживання енергії, що є ключовим для розвитку мереж Інтернету речей. Нині ця специфікація ще не поширена через свою новизну. Bluetooth 5, як і його попередники, зворотно сумісний. Можливо, через кілька років кожен мобільний пристрій підтримуватиме цю версію, що є важливою перевагою.

Wi-Fi HaLow - це протокол бездротової мережі, представлений у 2017 році як доповнення до стандарту IEEE_802.11 [7]. Протокол працює на частоті 900 МГц, яка не потребує ліцензування, та пропонує розширений діапазон покриття порівняно зі звичайними Wi-Fi мережами на 2,4 та 5 ГГц. Низьке споживання енергії є його перевагою, дозволяючи створювати великі мережі зі станцій або датчиків для Інтернету речей (IoT). Wi-Fi HaLow конкурує з

Bluetooth, пропонуючи більш високі швидкості передачі даних та ширший діапазон покриття (рис. 1.10).

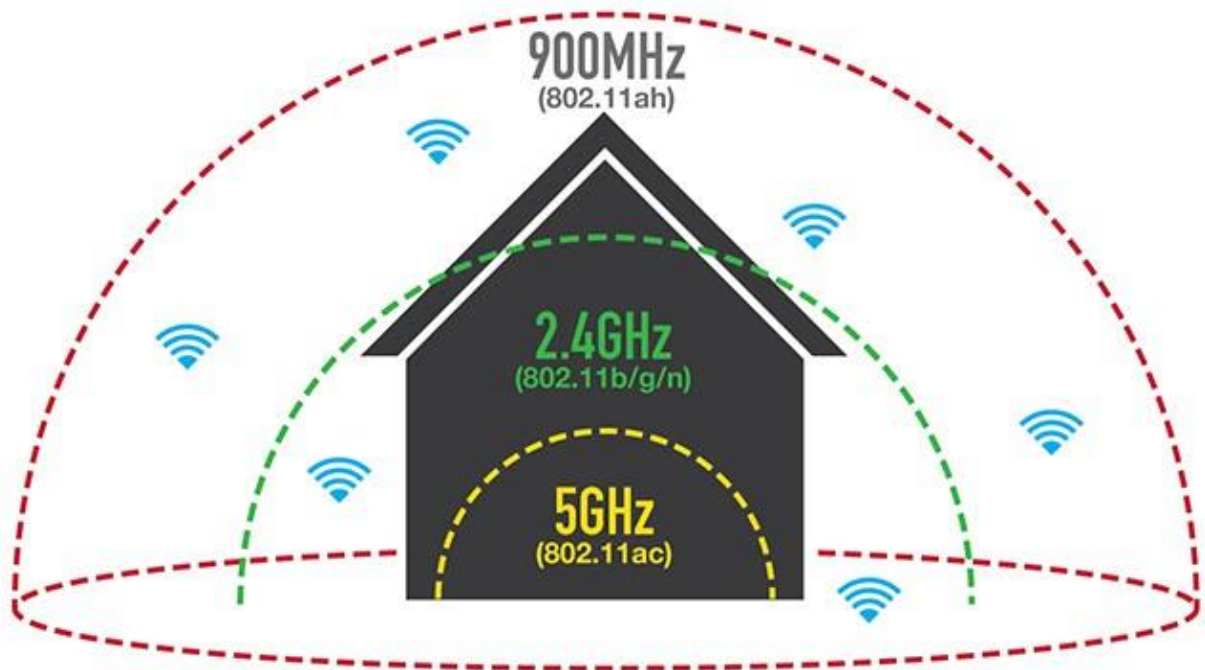


Рисунок 1.11 - Wi-Fi HaLow

Інформаційний обсяг, який генерується одним сенсорним вузлом, зазвичай є досить скромним. Однак, багато Інтернету речей сервісів засновані на обробці даних від великої кількості датчиків або клієнтів, що кардинально відрізняється від традиційних мережевих архітектур. Така ситуація породжує потребу в новій архітектурі, де багато джерел даних забезпечують інформацію численним одержувачам. Тут обсяг даних, який передає сенсорний вузол, може бути як надзвичайно малим, так і великим. Це ставить під сумнів можливість використання звичайних комунікаційних протоколів для передачі повідомлень.

Базова топологія для передачі повідомлень у мережах Інтернету речей має структуру "видавець-підписник" (Publisher-Subscriber, або pub/sub) (див. рис. 1.12).

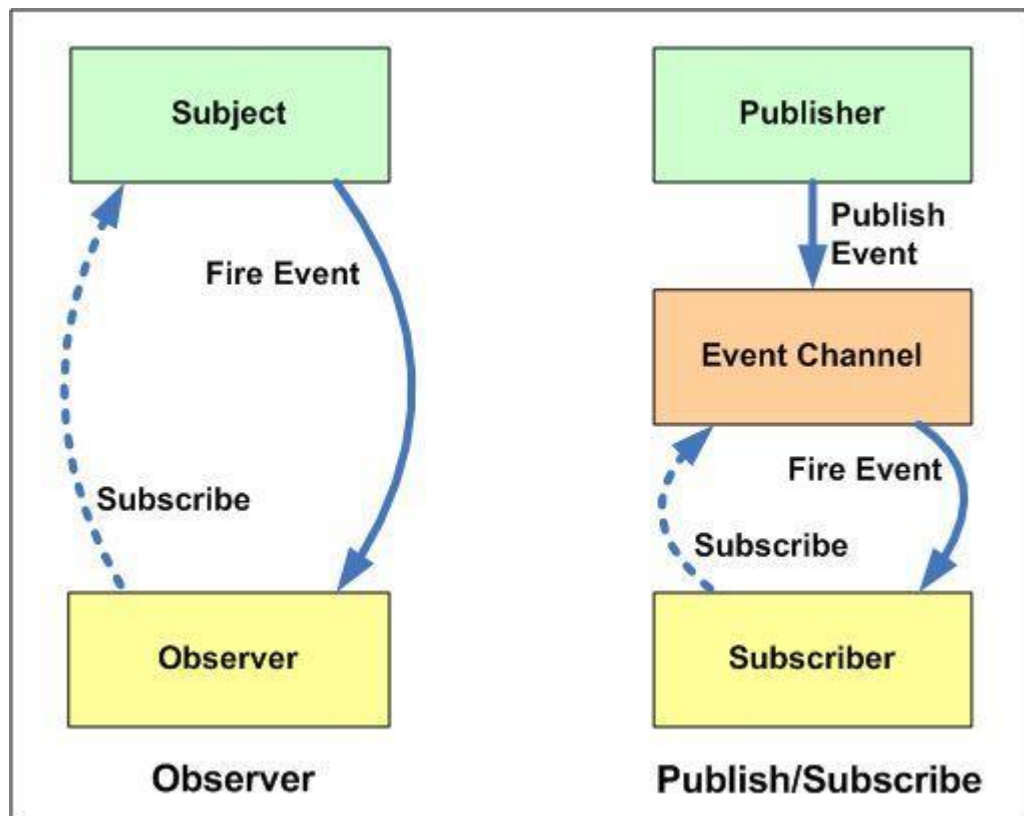


Рисунок 1.12 – Архітектура Publisher-Subscriber

У цій системі використовуються поняття видавця – джерела інформації та підписника – одержувача інформації. Підписка пов'язана з виконанням певних дій учасниками з метою отримання інформації від певного видавця, а також забезпеченням регулювання збору даних – наприклад, періодичності отримання та інших параметрів в залежності від специфіки системи.

У даному контексті розглядається ситуація, коли сенсорний вузол (Node) збирає інформацію від багатьох датчиків (наприклад, дані про вологість повітря) і відправляє її відповідно до визначених параметрів підписки або за запитом, або регулярно через встановлені інтервали часу. Оскільки самі датчики часто бувають простими, їхнє основне завдання – передавати інформацію про моніторинг параметрів. Тому виникає необхідність об'єднання датчиків у вузли з мікроконтролерами, які будуть відповідати за збір даних і їх передачу на сервер за певними алгоритмами.

Для взаємодії з системою клієнт часто потребує встановлення спеціалізованої клієнтської програми (Application) на персональний пристрій.

Ця програма дозволяє візуалізувати дані, отримані від датчиків, або вже оброблені сервером управління системою. У такій топології також присутній брокер (Broker). Брокер – це сервер, який отримує інформацію від видавців і передає її відповідним підписникам. У складних системах брокер може також займатися аналізом та обробкою надходження інформації. Він встановлює пріоритети для з'єднань і формує черги для передачі повідомлень. Так, брокер керує пересиланням повідомлень, їх зберіганням та фільтрацією. Черга повідомлень слугує як контейнер, де зберігаються повідомлення в процесі їх пересилання. Якщо канал зв'язку перевантажений, або одержувач недоступний у момент відправки повідомлення, черга зберігає повідомлення до моменту його доставки до одержувача.

Протоколи, які використовуються в мережах Інтернету речей, включають:

- DDS – протокол на рівні застосунків M2M для систем, що працюють у режимі реального часу;
- XMPP – протокол для миттєвого обміну повідомленнями та інформацією про присутність, працює майже у режимі реального часу;
- CoAP – розроблений спеціально для мереж та пристроїв з обмеженими ресурсами;
- MQTT – простий і відкритий протокол обміну даними, призначений для передачі даних у віддалених місцях;
- STOMP – використовується для встановлення зв'язку з брокером за принципом "запит-відповідь";
- SOAP – протокол обміну структурованими та неструктурованими повідомленнями у форматі XML у мережевому обчислювальному середовищі.

Далі розглянемо детальніше найпоширеніші з цих протоколів.

DDS (Data Distribution Service) є протоколом на рівні застосунків M2M для систем, що працюють у режимі реального часу, і базується на моделі "видавець-підписник". Основна мета протоколу - забезпечити зв'язок між пристроями через спільну шину обміну повідомленнями (рис. 1.13). DDS здатний швидко та синхронно доставляти мільйони повідомлень за секунду.

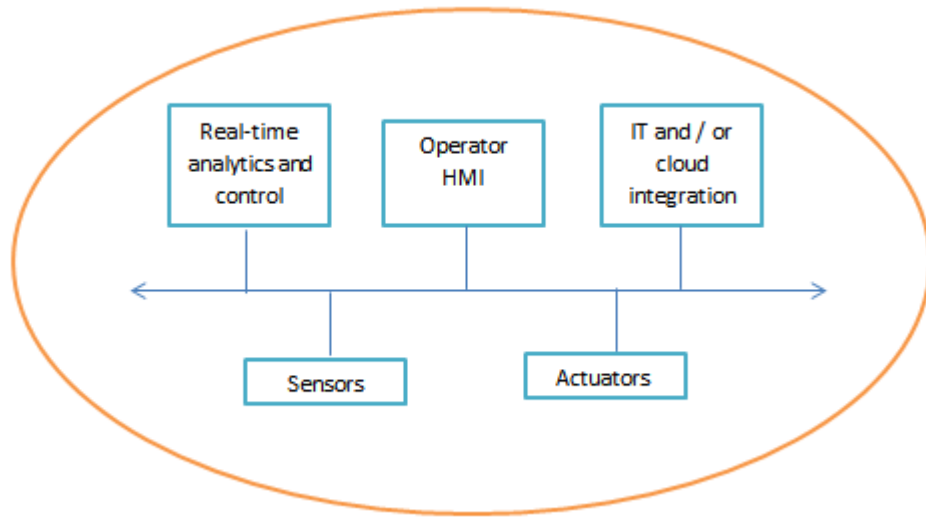


Рисунок 1.13 - DDS

Пристрої у мережах Інтернету речей вимагають інакшого підходу до зв'язку, ніж у звичайних ІТ мережах. Вони працюють швидко, з часовими рамками, вимірюваними в мікросекундах. Зв'язок між пристроями часто вимагає складних маршрутів, тому прості і надійні TCP-потoki не можуть задовольнити ці потреби.

DDS надає детальний контроль над якістю обслуговування (QoS), багатоадресну передачу, гнучкість у надійності та комплексність у резервуванні. Особливою перевагою DDS є можливість розширеного керування даними. Протокол дозволяє виконувати потужну фільтрацію і вибірку даних згідно з адресами призначення, де кількість синхронних одержувачів може налічувати тисячі. Ураховуючи, що деякі пристрої є компактними, існують спрощені версії протоколу DDS для роботи в умовах обмеженого обсягу.

Замість зіркоподібної мережі, DDS реалізує прямий шинний зв'язок між пристроями на основі реляційної моделі даних. Ця модель відома як шина даних (DataBus), оскільки вона є аналогом бази даних (database) в мережевому просторі.

DDS використовується у високопродуктивних пристроях. Це єдина технологія, що забезпечує необхідну гнучкість, надійність та швидкість для створення складних застосунків у реальному часі. До таких застосунків

належать військові системи, вітроелектростанції, інтегровані медичні системи, діагностичні системи візуалізації, системи управління ресурсами, автомобільні системи тестування та безпеки.

MQTT (Message Queue Telemetry Transport) представляє собою легкий, ефективний та відкритий протокол передачі даних, розроблений спеціально для обміну інформацією в умовах віддалених місцезнаходжень, де існують обмеження щодо обсягу коду і пропускної спроможності мережі. Ці характеристики роблять його ідеальним для використання в машина-машина (M2M) системах. Додатково, існує версія MQTT-SN (MQTT для сенсорних мереж), раніше відома як MQTT-S, що призначена для бездротових вбудованих систем без TCP/IP.

Процес роботи MQTT простий: виробник передає дані (наприклад, показники вологості) брокеру, зазначаючи тему (наприклад, "вологість"). Брокер визначає, які абоненти підписані на цю тему, та надсилає їм відповідні дані. Це дозволяє багатьом абонентам отримувати інформацію з різних джерел, не вступаючи в прямий контакт з джерелом даних.

STOMP (Simple Text Oriented Message Protocol) – це простий текстовий протокол для обміну повідомленнями, що підтримує широкий спектр мов, платформ та брокерів. Протокол організовує комунікацію з брокером через команди типу "запит-відповідь", такі як SEND (відправити), SUBSCRIBE (підписатися), UNSUBSCRIBE (відписатися) та інші. Схожий на HTTP, використовує TCP та забезпечує взаємодію між різними платформами та клієнтами.

Кожен протокол має своє призначення: DDS використовується для високошвидкісного обміну великими обсягами даних, MQTT - для важких мереж з обмеженою пропускною спроможністю і малим обсягом коду, а STOMP служить для спрощення та уніфікації систем обміну повідомленнями між різними протоколами.

1.5 Висновки до розділу 1

У сфері інтернету речей для обміну даними використовуються різноманітні технології та протоколи, що залежать від конкретних умов та середовища, де розгортається мережа. У домашніх мережах найбільш підходящими є технологія Wi-Fi HaLow та протокол MQTT. Для створення персональних мереж (PAN), найкраще використовувати технології BLE та RFID, які забезпечують комунікацію між пристроями на коротких дистанціях, а також протоколи MQTT та DDS для передачі команд та даних. Протокол STOMP, у свою чергу, забезпечує з'єднання з різними пристроями та вирішує проблему сумісності з протоколами нижчих рівнів, сприяючи уніфікації доступу до сервісів.

Цей розділ охоплює детальний аналіз архітектури Інтернету речей та його мережевих протоколів. Описано ключові протоколи, що використовуються для створення таких мереж. Для короткодистанційного зв'язку між пристроями оптимальними є технології BLE та NFC, які забезпечують найкращий баланс швидкості передачі даних і енергоефективності. Wi-Fi HaLow є ідеальним варіантом для передачі даних на середні відстані, оскільки він поєднує низьке енергоспоживання, аналогічне Bluetooth, з вищою швидкістю передачі даних та більшим радіусом дії.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Програмні середовища для роботи IoT

The Thing System – це система для моніторингу IoT екосистеми, створена американськими розробниками разом із Денні Гудманом, відомим автором у сфері JavaScript та HTML. Цей проект має на меті інтегрувати IoT пристрої в

системах домашньої автоматизації під одним централізованим контролем. Зазвичай, пристрої від різних виробників не сумісні та працюють окремо. Для вирішення цієї проблеми розробники створили програмне забезпечення, сумісне з різними мережевими протоколами, гаджетами та клієнтськими програмами. Інформацію про сумісні пристрої можна знайти на веб-сайті проекту.

Софтвер Steward, являючись ключовим компонентом The Thing System, розроблено на основі node.js, що забезпечує його портативність та гнучкість. Це ПЗ може працювати на ноутбуках, віддалених серверах або навіть на одноплатних комп'ютерах типу Raspberry Pi. Переваги цього ПЗ включають його компактність, сумісність з багатьма платформами, придатність для IoT пристроїв та простоту установки та налаштування. Однак, система має свої обмеження: вона оптимізована для домашніх мереж і може мати проблеми з конфігурацією та продуктивністю при розширенні на більш складні мережі, особливо через обмеження платформи node.js та підтримку обмеженої кількості пристроїв. Таким чином, система найкраще підходить для домашніх та простих локальних мереж, але може зіткнутися з проблемами при масштабуванні.

Domoticz – це система домашньої автоматизації, створена для керування різноманітними пристроями та обробки сигналів з сенсорів. Ця система підтримує широкий спектр пристроїв від різних виробників та має відкритий код, більшість якого написана на C++, і розповсюджується під ліцензією GNU GENERAL PUBLIC LICENSE. Її користувацький інтерфейс – це адаптивний HTML5 веб-інтерфейс, сумісний з різними пристроями. Domoticz працює на Windows, Apple, Cubieboard, Raspberry Pi та інших Unix-подібних операційних системах, та має мобільні додатки для дистанційного керування. Система надає можливість програмувати обробку подій та сигналів візуальним способом через додаток Blockly.

Переваги Domoticz включають її сумісність з різними платформами, зручний веб-інтерфейс, підтримку великої кількості пристроїв, можливість користувацького програмування, простоту установки та налаштування,

гнучкість у масштабуванні та програмуванні поведінки пристроїв, а також відкритість коду для налаштування та розширення функціоналу. Однак, система може мати проблеми з продуктивністю на пристроях з обмеженою оперативною пам'яттю та може бути несумісною зі старими версіями браузерів через використання HTML5. Особливість системи полягає в можливості асинхронного управління через JSON запити, при цьому вона містить детальну документацію щодо усіх можливих запитів для керування системою.

2.2 Мікропроцесор системи

Raspberry Pi model B - це компактний одноплатний комп'ютер, який спочатку був розроблений для навчання основ комп'ютерних наук у шкільному середовищі (рис. 2.1). Завдяки прогресу в області мобільних процесорів, він швидко став популярним вибором серед любителів, що працюють над проектами для інтернету речей. Головні переваги цієї платформи включають доступність за ціною, багатофункціональність та відкритість.

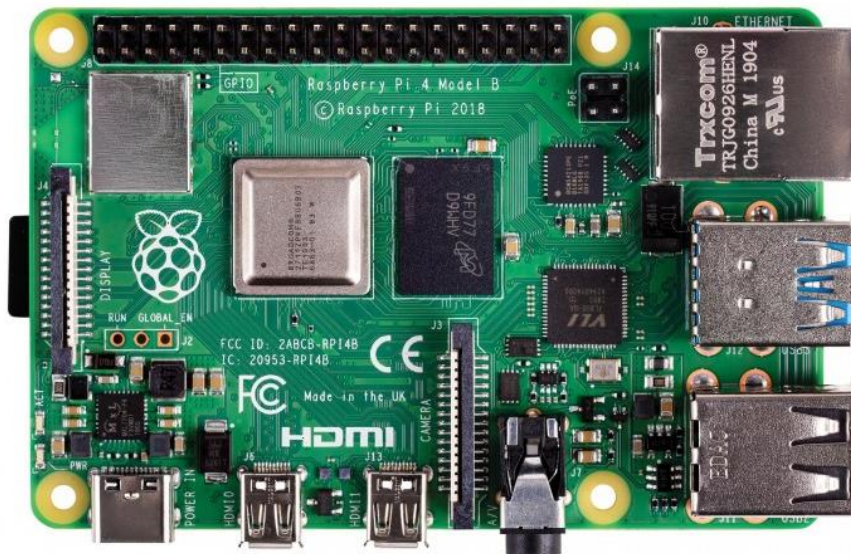


Рисунок 2.1 - Raspberry Pi model B

Основні особливості Raspberry Pi включають:

- підтримку операційних систем на основі Unix;
- можливість відтворення відео у Full HD якості;
- мінімальний об'єм оперативної пам'яті 512 МБ;
- габарити плати становлять 85,6 * 54 * 17 мм;
- наявність композитного відеовиходу RCA та HDMI для підключення монітора, а також 3.5-міліметрового аудіовиходу.

Плата Raspberry Pi може живитися від декількох джерел: через micro-USB роз'єм на 5В, універсальні піни або через адаптер на 5В і 2А. Останній метод забезпечує найбільш стабільне живлення, особливо при підключенні великої кількості периферійних пристроїв.

Плата підтримує широкий спектр інтерфейсів (рис 2.2), включаючи UART, слот для карт SD, MMC, SDIO, роз'єм HDMI, 3.5мм стерео, композитний відеовихід, 2x USB 2.0 порти, 10/100Mb RJ45 Ethernet, Wi-Fi 802.11n, Bluetooth 4.1, 16 портів input/output 3.3В, інтерфейси I2C і SPI, ARM JTAG, DSI та MIPI CSI-2. Wi-Fi 802.11n і Bluetooth 4.1 забезпечуються за допомогою чіпа Broadcom BCM43438, який може бути інтегрований на плату через інтерфейс Shield або встановлений окремо.

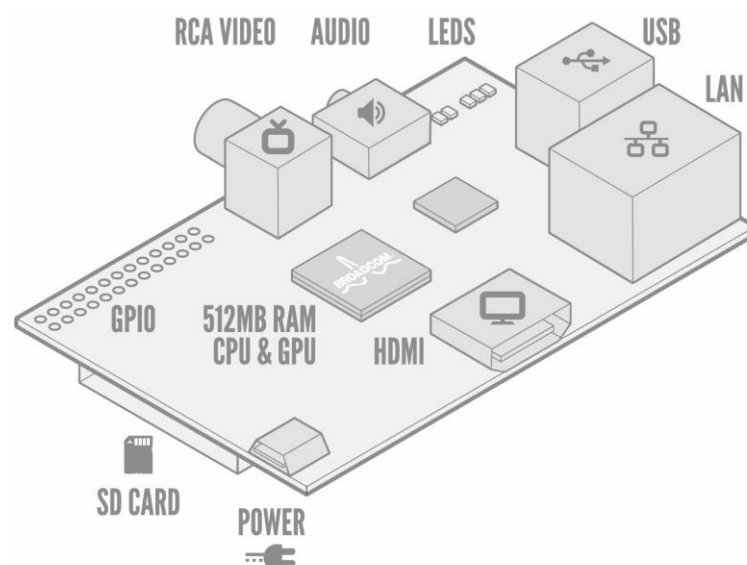


Рисунок 2.2 – Інтерфейси плати

Процесор Raspberry Pi B - 32-розрядний ARM Cortex-A7 з тактовою частотою 900 МГц, обладнаний кеш-пам'яттю першого (L1) на 16 Кб та другого (L2) рівнів на 128 Кб, останній з яких використовується переважно графічним процесором. SoC розміщений під мікросхемою оперативної пам'яті, тому не видимий.

З точки зору продуктивності, на частоті 700 МГц Raspberry Pi досягає продуктивності близько 0,041 GFLOPS, що приблизно еквівалентно продуктивності Pentium II 300 МГц 1997-99 років. GPU плати забезпечує обробку графіки зі швидкістю 1 Gpixel/s або 1.5 Gtexel/s і загальною обчислювальною продуктивністю 24 GFLOPS.

У якості програмного забезпечення, Raspberry Pi використовує операційні системи на базі Linux, а також підтримує установку Windows 10 IoT. Для установки ОС на Raspberry Pi можна використовувати інструменти на кшталт NOOBS та інших аналогічних програм для запису образів ОС на карту пам'яті.

Оскільки Raspberry Pi не має внутрішньої пам'яті для зберігання даних, потрібно попередньо налаштувати SD карту, завантаживши на неї образ операційної системи, що бажаєте використовувати [12-13]. Для цього процесу потрібно:

- SD (MMC/SDIO) карта об'ємом між 8 та 32Гб, із високою швидкістю читання, щонайменше класу 8;
- картридер для з'єднання карти з комп'ютером;
- програмне забезпечення Win32DiskImager;
- образ операційної системи Raspbian.

Згодом потрібно вставити карту у картридер, підключений до комп'ютера, та за допомогою Win32DiskImager записати на неї образ операційної системи.

Перше підключення Raspberry Pi та первинне налаштування. Для ініціації та конфігурації Raspberry Pi, крім самої плати та підготовленої карти пам'яті, вам знадобиться USB-клавіатура, монітор із HDMI входом або телевізор із RSA роз'ємом та відповідним кабелем для з'єднання, а також джерело живлення на 5В та щонайменше 0.5А[13].

Після підключення всіх компонентів та включення живлення, відразу розпочнеться завантаження операційної системи (рис. 2.3).

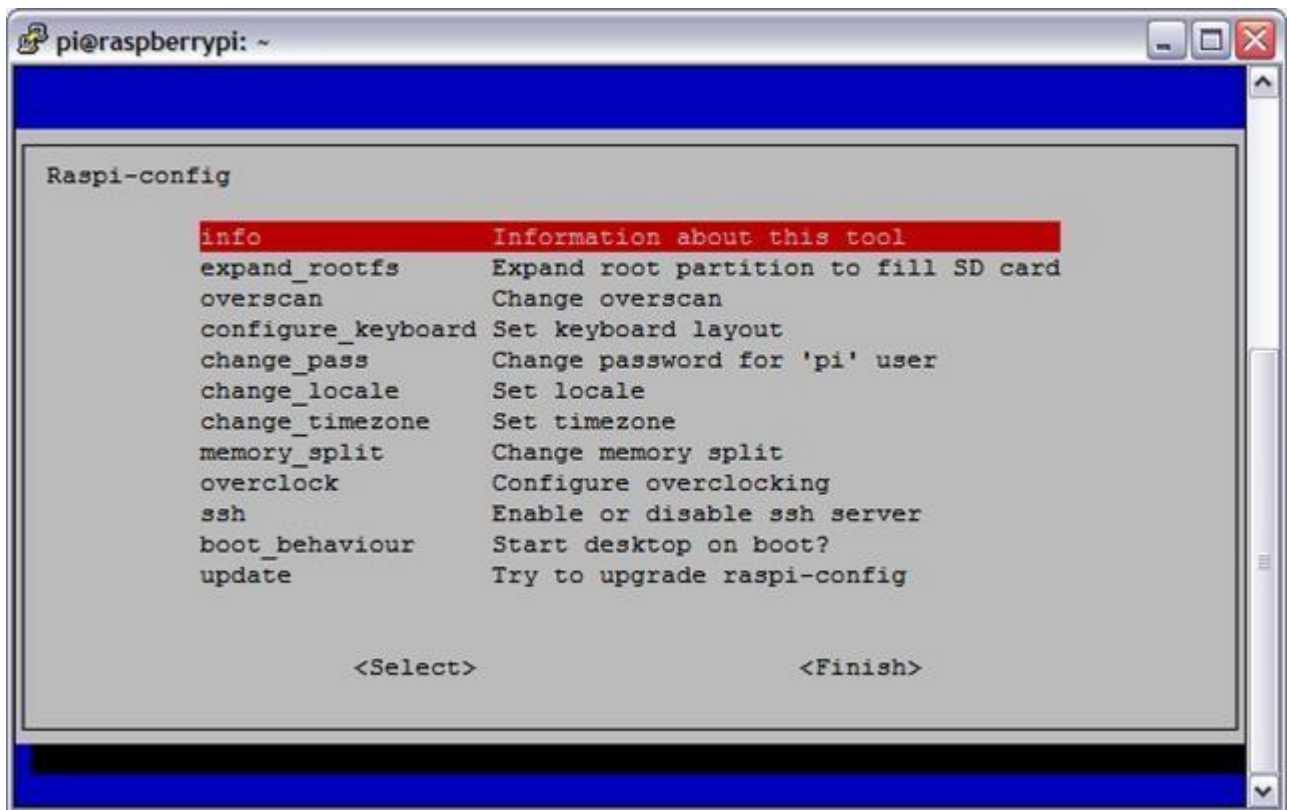


Рисунок 2.3 – Середовище програмування плати

Основні опції конфігурації включають:

- expand_rootfs: збільшення розміру root-розділу до максимального обсягу карти;
- configure_keyboard: вибір драйверу клавіатури, якщо стандартний варіант не влаштовує;
- change_pass: зміна пароля для користувача “pi” (пароль не відображається при вводі та потребує підтвердження);
- change_locale: зміна мови системи;
- change_timezone: налаштування часового поясу, який синхронізується через інтернет;
- memory_split: розподіл оперативної пам'яті;
- overclock: налаштування розгону процесора;

- ssh: увімкнення чи вимкнення сервера SSH, корисне для віддаленого доступу;
- boot_behaviour: налаштування автозавантаження графічного інтерфейсу при старті системи.

Після завершення налаштувань, потрібно перезапустити систему, натиснувши Ctrl+F і підтвердивши свій вибір. Система автоматично перезавантажиться.

Після завантаження системи необхідно встановити пароль для користувача “root”. В командному рядку вводиться “sudo passwd root” і вводиться пароль двічі. Щоб переглянути поточно активні процеси в системі, використовується команда “top” (рис 2.4).

```

pi@raspberrypi: ~
top - 13:32:51 up 18:08, 1 user, load average: 2,05, 2,33, 1,83
Tasks: 73 total, 1 running, 72 sleeping, 0 stopped, 0 zombie
%Cpu(s): 20,4 us, 2,3 sy, 0,0 ni, 63,9 id, 12,4 wa, 0,0 hi, 1,0 si, 0,0 st
KiB Mem: 189100 total, 175640 used, 13460 free, 2960 buffers
KiB Swap: 102396 total, 22272 used, 80124 free, 56084 cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 23225 eggdrops  20   0 13380 3468 1304 S 15,8   1,8   1:43.20 eggdrops
 25116 debian-t 20   0 46588 13m 1244 S  4,6   7,3  38:55.79 transmission-da
 7958  pi        20   0  6348 1368 1040 R  1,0   0,7   0:00.14 top
    32 root      20   0     0     0     0 D  0,7   0,0  30:01.47 mmccqd/0
    33 root      20   0     0     0     0 S  0,3   0,0   0:03.88 jbd2/mmcblk0p2-
   355 root      20   0     0     0     0 S  0,3   0,0   0:09.13 flush-179:0
  1722 root      20   0 32044  660  100 S  0,3   0,3   0:06.63 php5-fpm
  2252 mysql     20   0  309m 22m  196 S  0,3  12,0  4:21.85 mysqld
  3202 root      20   0 19936  748  360 S  0,3   0,4   0:01.43 smbd
  3623 pi        20   0  8640  944  520 S  0,3   0,5   0:33.93 eggdrops
    1 root      20   0  2136   88   68 S  0,0   0,0   0:03.91 init
    2 root      20   0     0     0     0 S  0,0   0,0   0:00.04 kthreadd
    3 root      20   0     0     0     0 S  0,0   0,0   0:00.32 ksoftirqd/0
    6 root      0 -20     0     0     0 S  0,0   0,0   0:00.00 khelper
    7 root      20   0     0     0     0 S  0,0   0,0   0:00.01 kdevtmpfs
    8 root      0 -20     0     0     0 S  0,0   0,0   0:00.00 netns
    9 root      20   0     0     0     0 S  0,0   0,0   0:00.58 sync_supers
  
```

Рисунок 2.4 - Виконання команди top

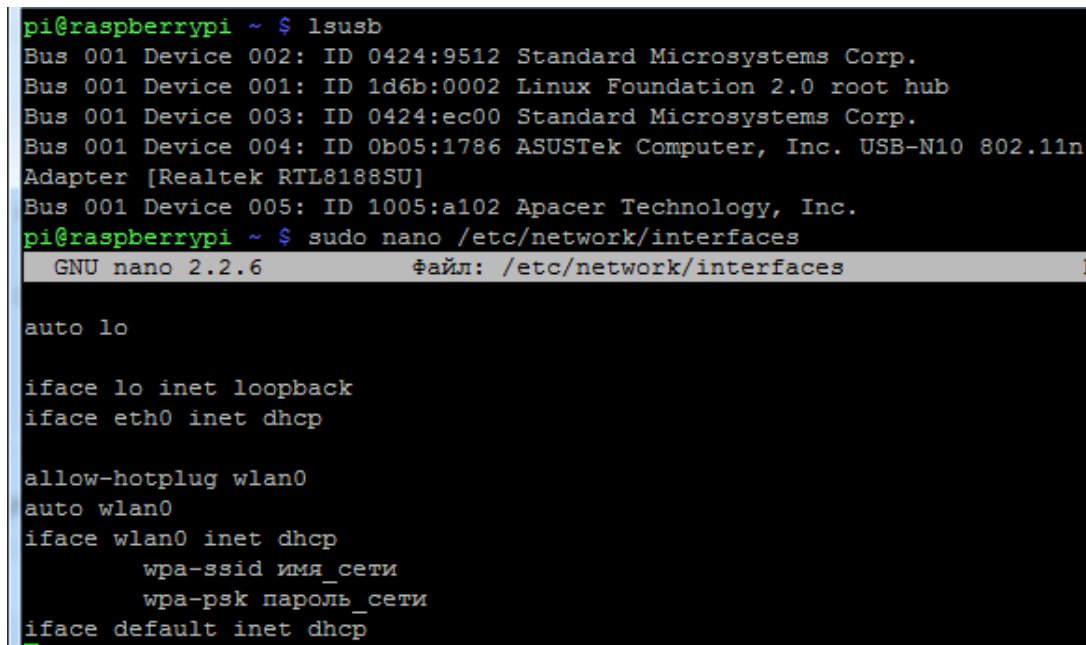
2.3 Зовнішні модулі

Для забезпечення бездротового з'єднання використовується Wi-Fi адаптер для Raspberry Pi, модель Auspi Wireless Adaptor – 802.11 B/G/N. Цей адаптер

вірізняється підтримкою діапазонів b, g, n, сумісністю з USB і не вимагає додаткового джерела живлення[13].

Будь-який зовнішній диск, який можна підключити через USB, підійде для використання.

Після підключення пристроїв першочергово перевіряємо їх наявність у системі за допомогою команди “lsusb”, оскільки вони підключені через USB. Якщо пристрої успішно виявлені, необхідно внести зміни в конфігурацію мережі wlan0 у файл /etc/network/interfaces. Для цього відкриваємо файл за допомогою “sudo nano /etc/network/interfaces” і робимо зміни (рис. 2.5).



```

pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 0b05:1786 ASUSTek Computer, Inc. USB-N10 802.11n
Adapter [Realtek RTL8188SU]
Bus 001 Device 005: ID 1005:a102 Apacer Technology, Inc.
pi@raspberrypi ~ $ sudo nano /etc/network/interfaces
GNU nano 2.2.6      файл: /etc/network/interfaces

auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0
iface wlan0 inet dhcp
        wpa-ssid имя_сети
        wpa-psk пароль_сети
iface default inet dhcp

```

Рисунок 2.5 -Виконання команди lsusb та редагування файлу налаштувань мережі

Далі, для оновлення налаштувань мережі використовується команда “service networking restart”, а потім послідовно оновлюємо систему командами “apt-get update” та “apt-get upgrade”.

Наступний крок - підключення зовнішнього жорсткого диску. Спочатку створюємо папку “data” у директорії “/media” для монтування файлових розділів. У файлі “/etc/fstab” додаємо UUID розділу та шлях до директиви, як показано на рис. Після збереження конфігураційного файлу, монтуємо розділ за

допомогою “`sudo mount /media/data`”. Для кожного наступного розділу цю процедуру повторюємо. Після цього перезавантажуємо Raspberry Pi та від'єднуємо клавіатуру та монітор, оскільки подальше управління здійснюватиметься через SSH.

```

pi@raspberrypi ~ $ sudo blkid
/dev/mmcblk0p1: SEC_TYPE="msdos" LABEL="boot" UUID="5D2D" TYPE="vfat"
/dev/mmcblk0p2: UUID="41cd5baa-7a62-4706-b8e8" TYPE="ext4"
/dev/sda1: LABEL="DATA" UUID="2C38225A2C38225A" TYPE="ntfs"
pi@raspberrypi ~ $ sudo nano /etc/fstab
GNU nano 2.2.6 Файл: /etc/fstab

proc            /proc          proc           defaults      0            0
/dev/mmcblk0p1  /boot          vfat           defaults      0            2
/dev/mmcblk0p2  /              ext4           defaults,noatime 0            1
# a swapfile is not a swap partition, so no using swapon/off from here on, use $
UUID=2C38225A2C38225A /media/data ntfs-3g users,defaults 0 0
  
```

Рисунок 2.6 - Пошук зовнішнього диску та налаштування файлу /etc/fstab

У налаштуваннях маршрутизатора потрібно встановити статичну IP-адресу для Raspberry Pi та налаштувати параметри DDNS провайдера. Після цих кроків налаштування плати завершені, і її можна використовувати як сервер, керуючи подальшими діями через SSH-з'єднання.

Щоб створити SSH-з'єднання з операційною системою Windows, необхідно використовувати безкоштовний інструмент PuTTY[14]. Необхідно завантажити наступні компоненти з веб-сайту PuTTY:

- PuTTY.exe – основний клієнт для з'єднань Secure Shell;
- PuTTYgen.exe – утиліта для генерації публічних і приватних SSH-ключів;
- Pagent.exe – агент аутентифікації для зберігання ключів;

– PSCP.exe – інструмент для безпечного копіювання файлів через командний рядок;

– PSFTP.exe – інструмент для безпечного копіювання файлів через FTP.

Для належної роботи цих інструментів у Windows, потрібно встановити системні змінні (рис. 2.7):

– назва змінної – PATH;

– значення змінної - C:\BIN; %PATH%;

Це необхідно, щоб програма могла знаходити всі необхідні системні файли.

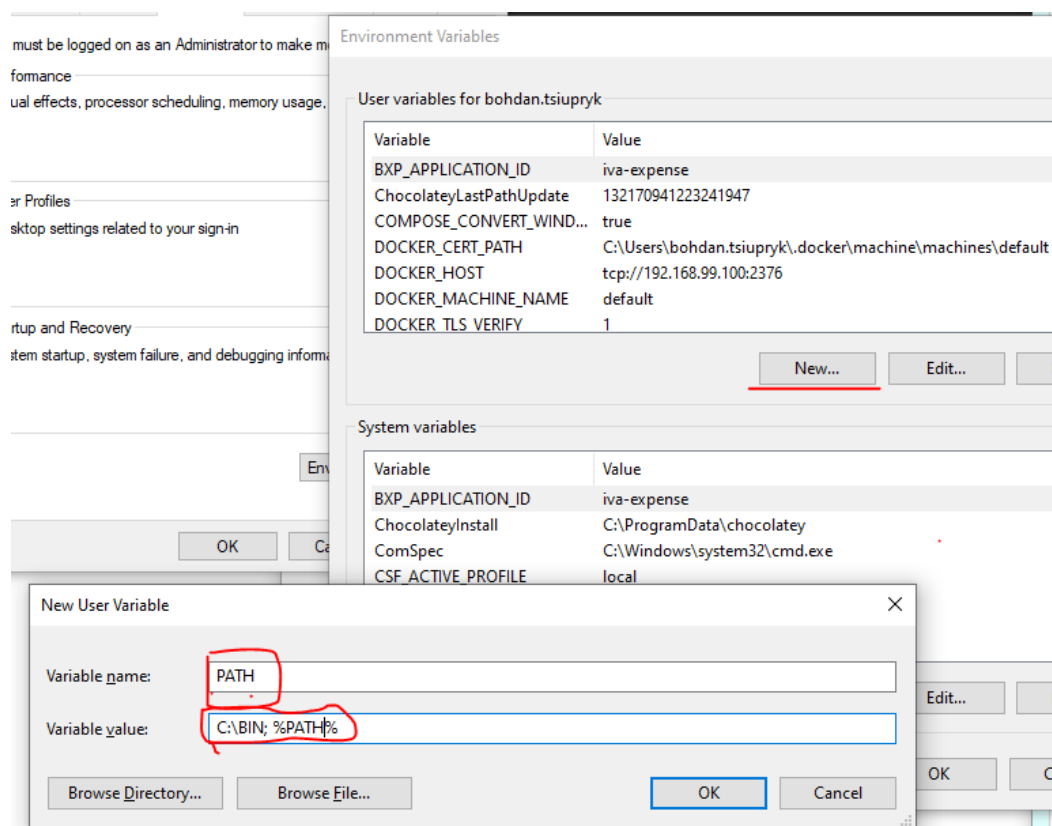


Рисунок 2.7 - Встановлення системних змінних для PuTTY

Далі розглянемо інтерфейс програми PuTTY, її налаштування та процедуру підключення[14]. На рисунку 2.8 показано, як налаштовувати сесію для підключення.

Основні параметри налаштування підключення:

- Host name (or IP address) - ім'я хоста або його IP-адреса, до якого буде виконуватись підключення;
- Port - порт для SSH-з'єднання, зазвичай це 22;
- Connection type - вибір типу з'єднання, у цьому випадку SSH;
- Saved session - можливість збереження сесій для швидкого доступу, вимагає введення назви сесії та натискання кнопки Save.

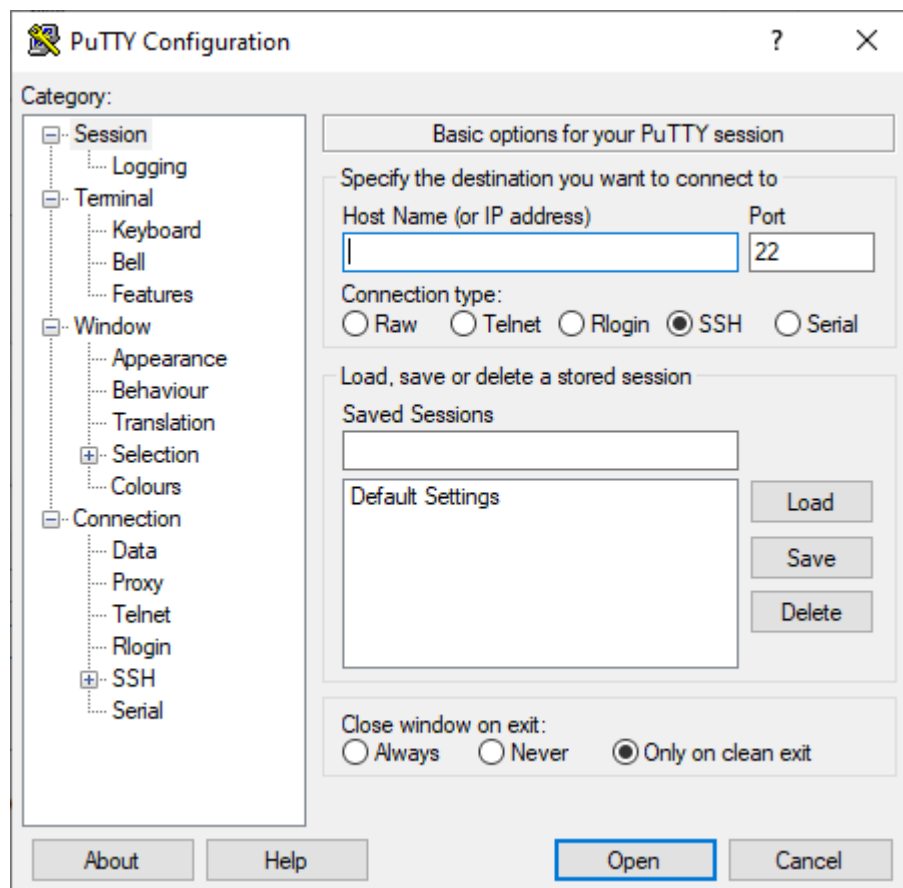


Рисунок 2.8 - Інтерфейс програми PuTTY, вкладка Session

Після введення усіх необхідних даних, потрібно натиснути кнопку Open для відкриття терміналу. У терміналі необхідно ввести логін та пароль, які були встановлені під час первинної конфігурації Raspberry Pi. В результаті буде встановлене з'єднання з платою.

Для інсталяції Domoticz спочатку потрібно перейти у домашній каталог, використовуючи команду `“cd /home/”`. Потім завантажити та ініціювати встановлення за допомогою команди `“curl -L https://install.domoticz.com |`

bash”[15]. Завершивши встановлення, переходимо у веб-браузер за адресою “IP-адреса-Raspberry-Pi:8080”, де відкриється головна сторінка Domoticz (рис. 2.9). На початку вона буде порожньою, оскільки до системи ще не підключено жодних пристроїв.

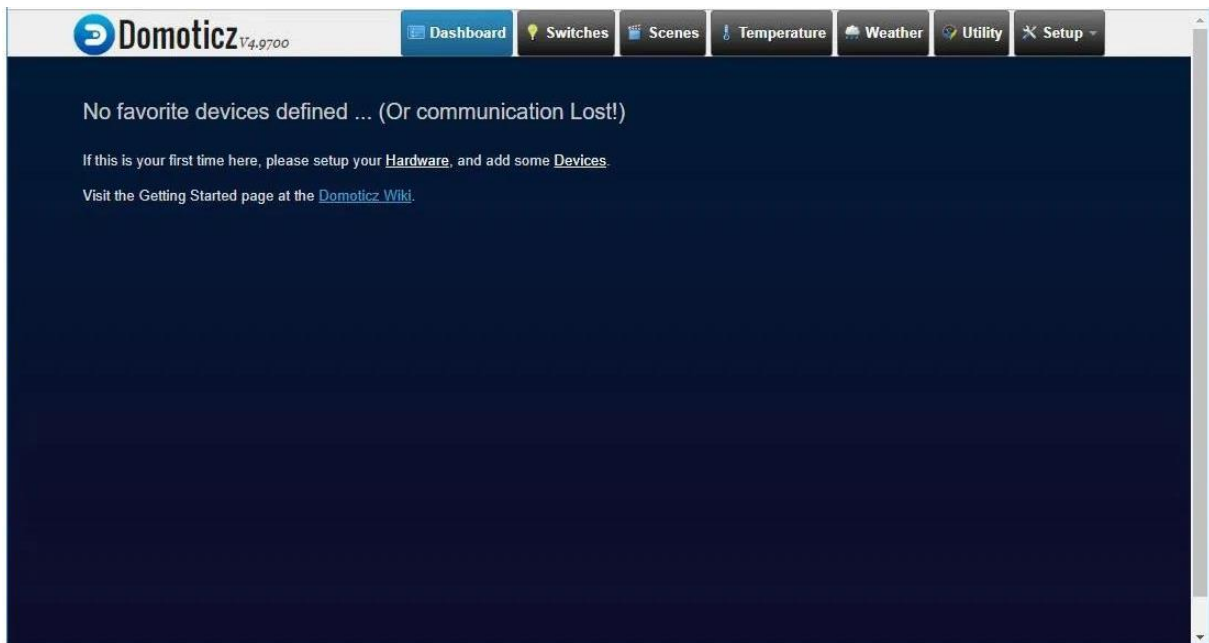


Рисунок 2.9 - Головна сторінка Domoticz

Для налаштування системи переходимо в Setup - Settings (рис. 2.10) та налаштовуємо параметри, такі як мова, тема, локація, захист сайту та інше. Зміни зберігаємо, натиснувши Apply Setting.

Тепер, коли початкове налаштування завершено, можна приступити до додавання датчиків для тестування.

Підключаємо два датчики температури DS18B20 через інтерфейс 1-wire, датчик температури та атмосферного тиску BMP180, а також датчик освітленості TSL2561 через інтерфейс I2C. Схема підключення GPIO на Raspberry Pi представлена на рисунку 2.10.

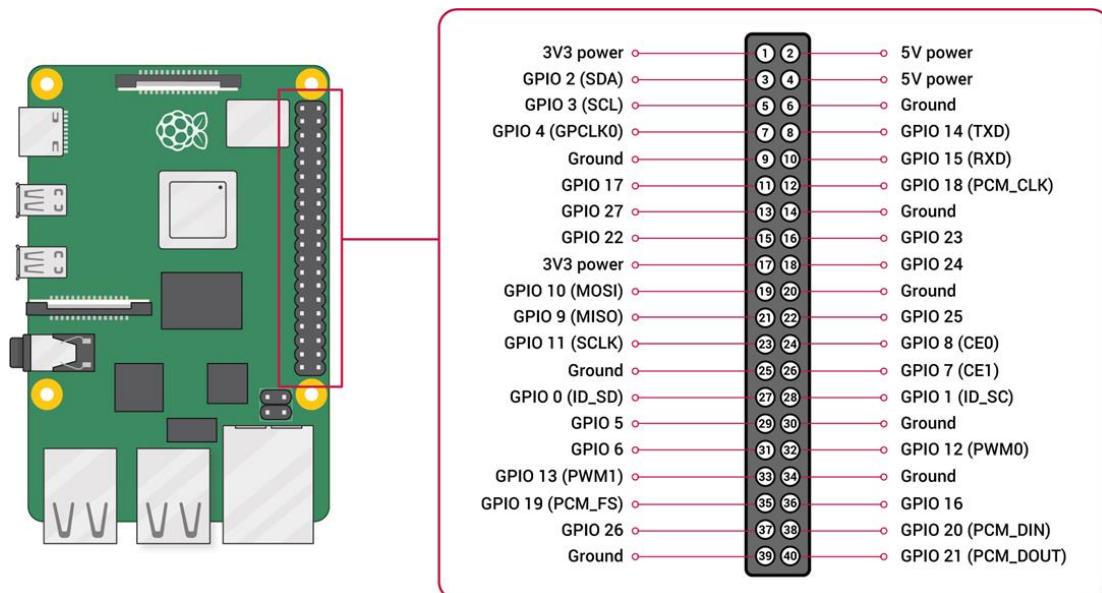


Рисунок 2.10 – Схема підключення

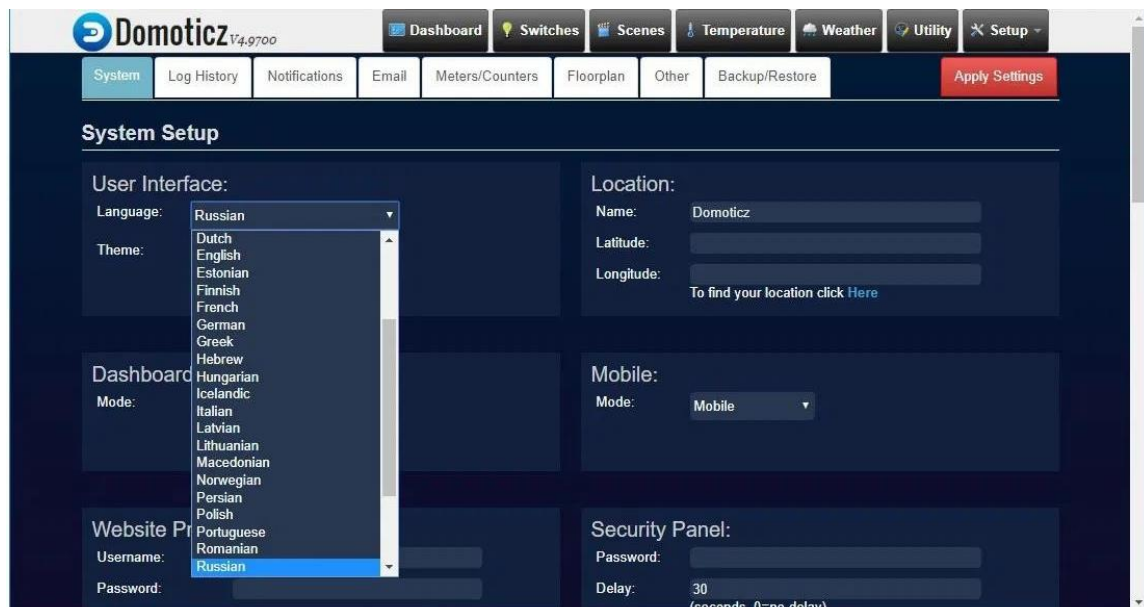


Рисунок 2.11 - Панель налаштувань Domoticz

Підключення датчиків у Domoticz виконується через веб-інтерфейс. Переходимо в Setup, вибираємо Hardware, де бачимо список пристроїв та форму для додавання нових (рис. 2.12). Для датчика DS18B20 вводимо назву у поле Name, вибираємо тип “1-wire (System)” і видаляємо стандартний шлях в полі OWFS Path. Інші параметри залишаємо без змін. Додавання пристрою здійснюється натисканням на кнопку Add. Так само додається і другий датчик DS18B20[18].

Для датчика BMP180 обираємо у Type “I2C sensors” та в SubType - “I2C sensor BMP085/180 Temp+Baro” (рис. 2.12). Датчик TSL2561 налаштовується аналогічно, але у SubType вибираємо “I2C sensor TSL2561 Illuminance”[19].

Idx	Name	Enabled	Type	Address	Port	Data Timeout
2	DS18B20 #1	Yes	1-Wire (System)			Disabled

Showing 1 to 1 of 1 entries

Update Delete

Enabled: ☒

Name: DS18B20 #1

Type: 1-Wire (System)

Data Timeout: Disabled

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.
Do not enable this option for devices that do not receive data!

OWFS Path:

Sensor Poll Period: 300000 (milliseconds)

Switch Poll Period: 100 (milliseconds)

Add

Рисунок 2.12 - Список пристроїв та форма додавання нового в Domoticz

Idx	Name	Enabled	Type	Address	Port	Data Timeout
4	BMP180	Yes	I2C sensor BMP085/180 Temp+Baro	I2C		Disabled
3	DS18B20 #2	Yes	1-Wire (System)			Disabled
2	DS18B20 #1	Yes	1-Wire (System)			Disabled

Showing 1 to 3 of 3 entries

Update Delete

Enabled: ☒

Name: BMP180

Type: I2C sensors

Data Timeout: Disabled

Specifying a Data Timeout will restart the hardware device if no data is received for the specified time.
Do not enable this option for devices that do not receive data!

SubType: I2C sensor BMP085/180 Temp+Baro

Path to I2C bus: (etc. /dev/i2c-1, or empty for autodetection)

Add

Рисунок 2.13 - Налаштування датчика BMP180

Після підключення датчики знаходяться в неактивному стані. Щоб активувати датчики, потрібно перейти в Setup, вибрати Settings і активувати датчик, натиснувши на зелене кільце біля потрібного датчика та вказавши його

назву. У системі датчики розподілені за функціональністю, наприклад, датчики температури DS18B20 та BMP180 (температура) будуть відображатись у вкладці Temperature, BMP180 (атмосферний тиск) - у Weather, а TSL2561 - в Utility. Для зручності, найчастіше використовувані датчики можна додати на головну панель Dashboard, натиснувши на зірочку поруч із пристроєм.

Domoticz також має систему логування, що дозволяє відстежувати динаміку показників датчиків. Щоб переглянути графік, достатньо вибрати датчик і натиснути на його іконку.

2.4 Висновки до розділу 2

У цьому розділі розглядається одноплатний комп'ютер Raspberry Pi як ідеальний варіант для створення системи моніторингу та керування в контексті інтернету речей.

Проведено ініціацію та первинне налаштування одноплатного комп'ютера Raspberry Pi. Також була виконана інсталяція системи Domoticz на Raspberry Pi, аналіз процедур налаштування та досліджені функції його користувацького програмного інтерфейсу (API).

3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Структура розподіленої системи

В цьому розділі розглянуто використання архітектури клієнт-сервер для розробки. Клієнтська частина являє собою додаток, розгорнутий на окремому комп'ютері або в мережі, що взаємодіє з програмою Domoticz. Сервер, у свою чергу, може бути розташований на віддаленому сервері або хостингу. Ця

архітектура забезпечує незалежність сервера від клієнтів, забезпечуючи йому автономність. Клієнти, в свою чергу, мають простий метод підключення до сервера і не залежать один від одного.

Додаток було розроблено на Java та за допомогою фреймворку Spring Boot. В якості системи керування базами даних використовується PostgreSQL. Також у процесі розробки застосовується ORM-фреймворк Hibernate, який має свою мову запитів HQL. HQL є об'єктно-орієнтованою альтернативою SQL і спрощує роботу з базами даних, зменшуючи необхідність ручного написання SQL-запитів. Веб-інтерфейс було створено за допомогою шаблонізатора Freemarker.

Середовище розробки обрано IntelliJ IDEA від компанії JetBrains. Це середовище має безкоштовну версію із потужними плагінами та інструментами, що спрощують розробку на Java.

Серверний додаток розроблений за шаблоном MVC (model, view, controller), що дозволяє слабе зв'язування модулів, полегшуючи їх підтримку і заміну. Деталі модулів:

- Model - це робота з базою даних, обробка та валідація даних;
- View - приймає вхідні дані та надсилає оброблені;
- Controller - основна бізнес-логіка, обробка та виконання операцій з даними.

Model відповідає за обробку основних даних, View відображає ці дані, а Controller відстежує події від користувача та керує даними.

Клієнтський додаток розроблено за монолітною архітектурою через відсутність потреби у підключенні до баз даних або веб-інтерфейсу. Основна функція - підтримка з'єднання з віддаленим сервером та передача повідомлень.

Java та Spring Boot. Java - це об'єктно-орієнтована мова, що використовує JVM для забезпечення крос-платформеності. Spring Boot спрощує розробку Java-додатків, надаючи автоматичну конфігурацію та вбудовані контейнери сервлетів, серед інших переваг.

PostgreSQL - це реляційна, об'єктна СУБД з відкритим кодом, що підтримує багатий функціонал і типи даних. Hibernate служить для взаємодії Java-класів з базами даних, використовуючи HQL як спрощений аналог SQL.

. Freemarker - це шаблонізатор для створення гнучких веб-інтерфейсів без глибоких знань у веб-програмуванні, з використанням макросів для зменшення кількості коду.

3.2 Розробка системи

Система містить наступні реалізовані моделі даних:

- Device, яка представляє пристрій, підключений до системи;
- DeviceDTO, модель пристрою для серіалізації та передачі, без зайвих даних;
- DeviceState, яка відображає стан пристрою;
- User, що представляє користувача системи;
- UserRole, відображає роль користувача у системі;
- Diagram, модель для графіків даних;
- Client, модель для клієнтської частини системи;
- Command, модель команд, які передаються або зберігаються для історії пристрою.

Управління моделями здійснюється через репозиторії "Repository", які є інтерфейсами фреймворку Hibernate, приклади імплементації яких показані на рис. 3.1. Такий підхід дозволяє уникнути повторення коду, оскільки без цього для кожної моделі потрібно було б реалізовувати окремі методи доступу до бази даних[24].

```
import org.springframework.data.jpa.repository.JpaRepository;

public interface DeviceRepo extends JpaRepository<Device, Long> {
}
|
```

Рисунок 3.1 - Приклад імплементації інтерфейсів для доступу до бази даних

Інтерфейси `JpaRepository` та `CrudRepository` надають базові методи доступу до бази даних, такі як пошук об'єктів за ідентифікатором, видалення або оновлення. Для розширення функціоналу, JPA дозволяє додавати власні методи керування базою даних через спеціальні назви методів. Наприклад, пошук пристроїв у стані OFF потребує правильного формулювання назви методу, такого як `"List<Device> findAllByStatus"` і передачу в параметрах статусу (рис. 3.2). Аналогічний підхід використовується для інших моделей.

```
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface DeviceRepo extends JpaRepository<Device, Long> {
    List<Device> findAllByStatus(DeviceStatus status);
}
```

Рисунок 3.2 - Приклад створення запиту для пошуку пристроїв за статусом

Розробка бізнес-логіки додатку. Вся бізнес-логіка зосереджена у пакеті "service". Основна мета додатку - збір та систематизація даних з декількох серверів Domoticz, аналіз статистики та контроль. Domoticz має API для запитів JSON, для відправки яких використовується вбудований інструмент Spring Boot - `RestTemplate`. Клас `HttpUtil`, що є сервісом для відправки запитів, демонструється на рис. 3.3.

```

public static HttpClientResponse executeRequest(HttpUriRequest request, CloseableHttpClient closeableHttpClient) {
    String jsonText = null;
    int statusCode;

    try (CloseableHttpResponse response = closeableHttpClient.execute(request)) {
        statusCode = response.getStatusLine().getStatusCode();
        Log.debug("Status code: {}", statusCode);
        HttpEntity entity = response.getEntity();
        if (entity != null) {
            jsonText = extractJsonStringFromHttpResponse(entity);
            EntityUtils.consume(entity);
        }
    }
}

```

Рисунок 3.3 - Приклад методу для надсилання запитів

Основні сервіси включають:

- HttpUtil для відправки запитів;
- DeviceService для управління пристроями;
- CommandService для створення та обробки команд;
- UserService для управління користувачами;
- DiagramService для обробки діаграм;
- MailService для розсилки листів;
- StatisticService для збору статистики;
- InitializationService для ініціалізації та налаштування системи.

Розробка користувацького інтерфейсу та обробка даних від користувачів. Інтерфейс користувача створено з допомогою Freemarker, використовуючи макроси для ефективного повторного використання коду. Для відображення діаграм використано Google Chart, який є легким та простим у використанні. Стилзація здійснена за допомогою Bootstrap 4. Контролери обробляють дані, що надходять від користувацького інтерфейсу, описуючи шляхи обробки даних та передаючи їх сервісам для подальших дій (рис. 3.4, 3.5).

```

<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
  google.charts.load('current', {'packages':['corechart']});
  google.charts.setOnLoadCallback(drawChart);

  function drawChart() {
    var data = google.visualization.arrayToDataTable([
      ['Year', 'Sales', 'Expenses'],
      ['2004', 1000, 400],
      ['2005', 1170, 460],
      ['2006', 660, 1120],
      ['2007', 1030, 540]
    ]);
  }

```

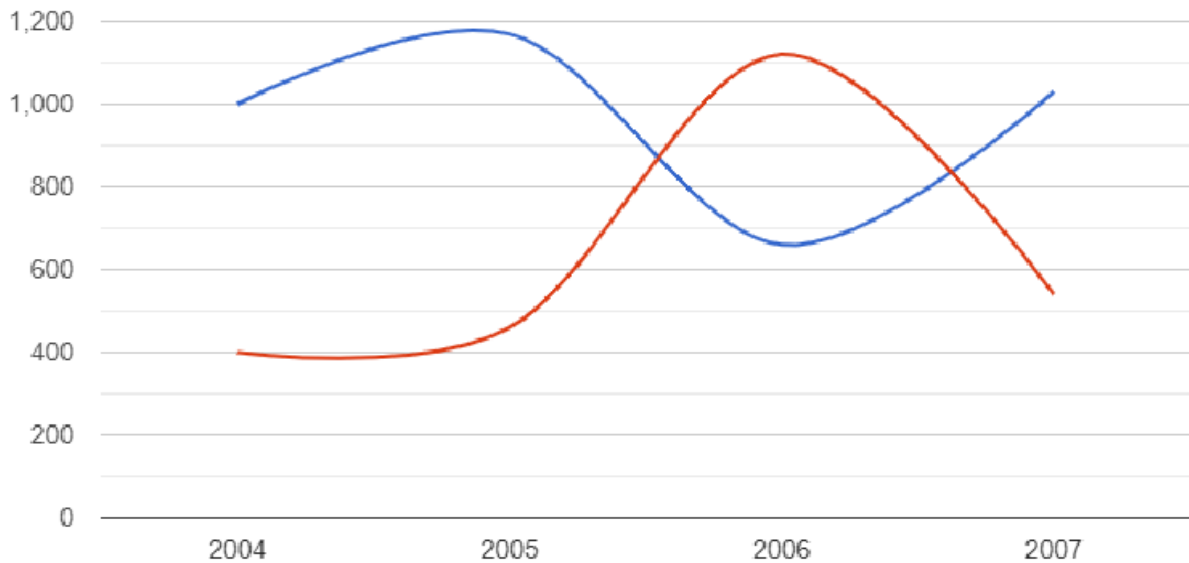


Рисунок 3.4 - Приклад коду діаграм

```

@RestController
@RequiredArgsConstructor
@RequestMapping("/device")
public class DeviceController {
    private final DeviceService deviceService;

    @GetMapping("all")
    public ResponseEntity<List<Device>> getAll() {
        List<Device> all = deviceService.getAll();
        return ResponseEntity.ok(all);
    }
}

```

Рисунок 3.5 - Приклад оголошення контролера та вказівки оброблюваного шляху

Конфігурація доступу до бази даних здійснюється через файл `db.properties` (рис. 3.6). Зміни конфігурації потребують перезапуску додатку, оскільки налаштування ініціюються при його старті.

Ключові параметри налаштування включають:

- `datasource.url` - адреса бази даних;
- `datasource.username` - ім'я користувача бази даних;
- `datasource.password` - пароль для доступу до бази даних.

```
datasource.url=jdbc:postgresql://localhost/iot-control-system
datasource.username=postgres
datasource.password=123
```

Рисунок 3.6 - Приклад налаштувань для доступу до бази даних

Конфігурація клієнтської частини проводиться у файлі `application.properties`, де встановлюються наступні параметри:

- `remote-server.ip` - IP-адреса віддаленого сервера для відправки даних;
- `remote-server.port` - опціональний параметр, використовується, якщо порт віддаленого сервера відрізняється від стандартного 8080;
- `remote-server.key` - ключ безпеки, без якого запити відхиляються;
- `remote-server.local-id` - ідентифікатор клієнта;
- `domoticz.login` - логін для доступу до Domoticz;
- `domoticz.password` - пароль для доступу до Domoticz;
- `domoticz.port` - опціональний параметр, потрібен, якщо порт Domoticz відрізняється від 8080.

Після налаштування запускається клієнтська частина, яка автоматично здійснює підключення до Domoticz, знаходить підключені пристрої та намагається підключитися до віддаленого сервера. У разі недоступності віддаленого сервера, програма переходить у режим очікування та регулярно намагається відновити з'єднання.

3.3 Використання комп'ютерної системи

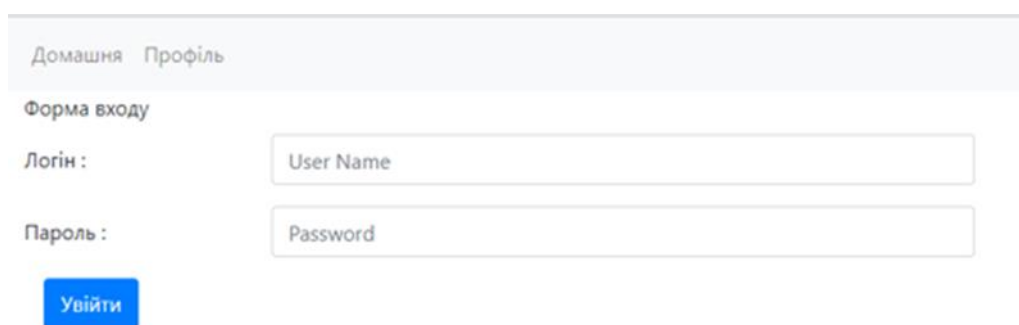
Процес встановлення та запуску системи, де ключовим елементом є наявність Java Virtual Machine (JVM) версії 8 або вище на комп'ютері. Це важливо, адже без такої версії JVM система не зможе працювати ефективно.

Для роботи клієнтського додатку потрібно перенести файл IoT-control-system-client.jar на комп'ютер, де вже встановлено програму Domoticz, та ініціювати його роботу. Запуск додатку виконується через введення команди “java -jar IoT-control-system-client.jar” у командний рядок, що дозволяє активувати клієнтську частину системи.

Процедура запуску серверного додатку, для якої необхідно мати встановлену базу даних із створеним користувачем. Після цього користувач має скопіювати файл IoT-control-system-server.jar та запустити його, використовуючи команду “java -jar IoT-control-system-server.jar”, що дозволить активувати серверну частину системи.

Системою можна користуватися як з персональних комп'ютерів та ноутбуків, так і з мобільних пристроїв. Основна вимога – наявність підтримки браузера. Для ініціації використання системи слід перейти за адресою <https://<ip-адреса-сервера>:8080>, після чого відкриється інтерфейс для входу в систему, візуалізований на рисунку 3.8.

На рисунку 3.7 представлена форма входу в систему. При першому вході в систему необхідно використовувати стандартні креденціали “admin:admin”.



Домашня Профіль

Форма входу

Логін :

Пароль :

Рисунок 3.7 – Вхід в систему

Після успішного входу користувачеві слід змінити пароль та при необхідності додати нових користувачів. Для цього в системі передбачена спеціальна панель управління користувачами, яка зображена на рисунку 3.8.

Список користувачів					
Логін	Номер телефону	Електронна пошта	Ролі		
Bohdan	+380987278916	bod9.hom9k@gmail.com	MODERATOR, USER	Редагувати	Видалити
Ivan	+380987278916	rewq@gmail.com	MODERATOR, USER	Редагувати	Видалити

Рисунок 3.8 – Список користувачів

Рисунок 3.9 демонструє панель адміністратора. На цій панелі адміністратор може редагувати інформацію про користувачів, змінювати їх права доступу, а також додавати нових користувачів. Це дозволяє забезпечити гнучке управління доступом та конфігурацією користувачів у системі.

Ім'я клієнта (також використовується як ідентифікатор)	Home
Додати клієнта та згенерувати ключ	Згенерувати
Ключ безпеки	a6d4e5e4-82ec-45b9-8382-ca489b1e52ee

Рисунок 3.9 – Панель адміністратора

Для інтеграції клієнтських додатків із системою, спочатку їх потрібно створити через спеціалізовану панель, зображену на рисунку 3.10. На цій панелі необхідно ввести назву клієнта, яка використовуватиметься в системі як ідентифікатор, та згенерувати ключ безпеки, що є важливою частиною процесу конфігурації безпеки.

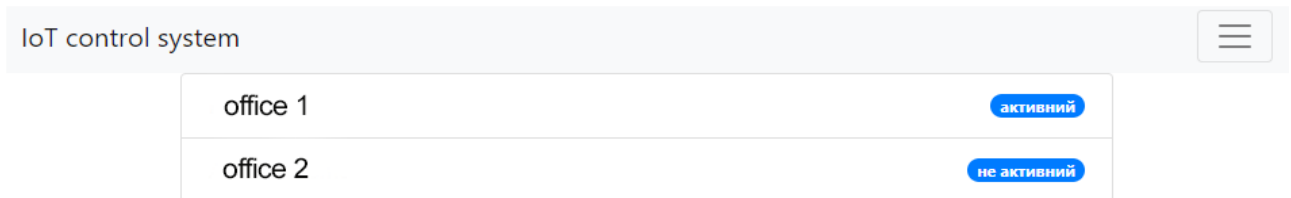


Рисунок 3.10 – Панель клієнтів

Рисунок 3.10 ілюструє процес додавання нового клієнта. Після створення нового клієнта наступним кроком є його конфігурація та запуск. Важливо використовувати ідентифікатор та ключ безпеки, отримані під час процесу створення, для правильної налаштування клієнта.

Після успішного підключення клієнта в системі, він відображатиметься у списку зі статусом “активний”, як показано на рисунку 3.10. Якщо клієнт ще не був підключений, він матиме статус “новий”. У випадках, коли підключення було перервано або пристрій вимкнено, статус клієнта зміниться на “не активний”.

Для перегляду детальної інформації про певного клієнта достатньо клікнути по його запису. Відкриється окреме вікно з усією необхідною інформацією про клієнта, включаючи назву пристрою, його IP-адресу, час роботи, кількість підключених пристроїв, а також можливість переглянути деталі про кожен з підключених пристроїв (рис. 3.11).

IoT control system Домашня Список користувачів Профіль	
	
Температура	Температура
Клієнт - Office 1	Клієнт - Office 2
10 C	24 C
Оновлено: 11.12.2023 07:00	Оновлено: 11.12.2023 07:00
Детальніше Вимкнути	Детальніше Вимкнути

Рисунок 3.11 – Детальна інформація

На вкладці “Домашня” представлено перегляд усіх пристроїв, які наразі підключені до системи. Тут можна відсортувати пристрої за клієнтом або типом пристрою, наприклад, датчики температури чи лампочки.

Рисунок 3.11 демонструє, як на домашній сторінці відображаються датчики температури та інші пристрої. З цієї сторінки також можна перейти до детальнішого огляду окремих пристроїв. Якщо пристрій вимкнено, він не буде відображатися на домашній сторінці, і для його активації потрібно буде перейти до налаштувань відповідного клієнта.

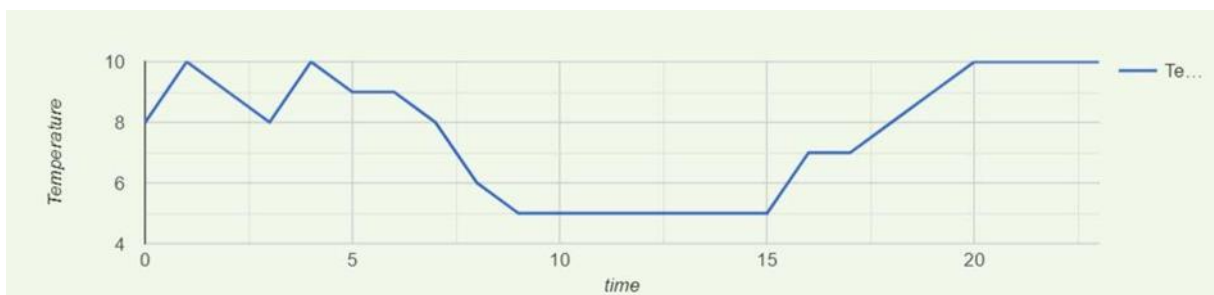


Рисунок 3.12 – Графіки показників

Для отримання детальної інформації про конкретний пристрій, необхідно скористатися посиланням "Детальніше" поруч з пристроєм. Це відкриє вікно з розгорнутою інформацією про пристрій, включно з його назвою, характеристиками, списком можливих команд, а також, наприклад, для датчиків температури, графіком коливань температури за останній день, як зображено на рисунку 3.12.

3.4 Висновки до розділу 3

Розробка комп'ютерної системи на основі клієнт-серверної архітектури є ефективним рішенням, яке гарантує автономність сервера і простоту підключення для клієнтів. Використання Java, Spring Boot, PostgreSQL та Hibernate сприяє ефективній розробці за допомогою зручних інструментів і

фреймворків. Веб-інтерфейс, створений за допомогою Freemarker, дозволяє розробляти гнучкі веб-сервіси, в той час як MVC-шаблон забезпечує слабе зв'язування модулів для полегшення їх підтримки.

Ефективне управління моделями даних, такими як Device, User, Command, через репозиторії Hibernate сприяє оптимізації доступу та управління базами даних. Використання JpaRepository та CrudRepository для базових операцій із базами даних, а також можливість розширення функціоналу через спеціальні методи, підвищує гнучкість системи. Розробка бізнес-логіки зосереджена у пакеті "service", що дозволяє чітко структурувати і оптимізувати обробку даних.

Користувацький інтерфейс, реалізований з використанням Freemarker і Bootstrap 4, забезпечує інтуїтивно зрозуміле управління та візуалізацію.

Завдяки використанню модульної архітектури, система легко інтегрується з додатковими серверами-клієнтами без необхідності переписування існуючого коду.

Також у розділі демонструється процес налаштування клієнтів: як їх додавати, конфігурувати та налаштовувати в системі.

ВИСНОВКИ

У цій магістерській роботі було проведено дослідження інструментів для моніторингу та керування пристроями Інтернету речей (IoT), щоб спростити взаємодію з мережею IoT та її компонентами в підприємстві ТОВ «ПродІнвест». Основні досягнення та висновки цього дослідження включають:

- вивчення сучасних методів передачі даних у мережі Інтернету речей;
- аналіз протоколів та технік передачі даних в мережі Інтернету речей, що дозволило визначити сильні та слабкі сторони кожного протоколу та вибрати найбільш ефективні для створення мережі;
- детальне дослідження системи Domoticz та її програмного інтерфейсу, які стали фундаментом для розробки системи моніторингу та управління IoT;
- розробка на базі Domoticz системи за допомогою мови Java та фреймворку Spring Boot, яка забезпечує можливість дистанційного моніторингу та керування IoT-елементами з різних локацій;
- демонстрація функціональності та роботи розробленої системи.

Всі задачі роботи були виконані та мета досягнута.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Integration of the ASR Toolkit Kaldi Into a Domoticz Home Automation System / M. Aboulhir et al. Transactions on Machine Learning and Artificial Intelligence. 2017. Vol. 5, no. 4. URL: <https://doi.org/10.14738/tmlai.54.3424> (date of access: 15.12.2023).
2. Wireless Home Automation System / P. Hamsagayatri et al. IOP Conference Series: Materials Science and Engineering. 2021. Vol. 1084, no. 1. P. 012075. URL: <https://doi.org/10.1088/1757-899x/1084/1/012075> (date of access: 15.12.2023).
3. IOT Based Home Automation System / Prof. P. P. Likhitar et al. International Journal of Advanced Research in Science, Communication and Technology. 2022. P. 863–868. URL: <https://doi.org/10.48175/ijarsct-5173> (date of access: 15.12.2023).
4. Android Based Home Automation System / P. D. Yelorkar et al. International Journal for Research in Applied Science and Engineering Technology. 2022. Vol. 10, no. 5. P. 4489–4491. URL: <https://doi.org/10.22214/ijraset.2022.43441> (date of access: 15.12.2023).
5. Костенко О. В. ІДЕНТИФІКАЦІЯ ІОТ. Знання європейського права. 2021. № 1. С. 77–83. URL: <https://doi.org/10.32837/chern.v0i1.177> (дата звернення: 15.12.2023).
6. Мушинський А. О., Бурак Н. Є. Інформаційна безпека пристроїв IoT : thesis. 2020. URL: <http://hdl.handle.net/123456789/6995> (дата звернення: 15.12.2023).
7. Архітектурна концепція моніторингової системи на основі нейронного модуля IoT аналітики даних / І. Клименко та ін. Адаптивні системи автоматичного управління. 2022. Т. 2, № 41. С. 111–123. URL: <https://doi.org/10.20535/1560-8956.41.2022.271355> (дата звернення: 15.12.2023).

8. Нікольський С. С., Клименко І. А. Розробка засобів для аналізу даних для систем IoT моніторингу стану дорожнього покриття. *Problems of Informatization and Management*. 2023. Т. 1, № 73. С. 45–56. URL: <https://doi.org/10.18372/2073-4751.73.17644> (дата звернення: 15.12.2023).

9. Maidaniuk N. V. Application of IoT Technologies for Cyber-physical Systems of a Sensible Enterprise. *Control Systems and Computers*. 2020. No. 2 (286). P. 30–40. URL: <https://doi.org/10.15407/csc.2020.02.030> (date of access: 15.12.2023).

10. Можливості авторизації та захисту даних користувача під час розробки хмарних веб-додатків для IoT / М. Polishchuk та ін. *Computer technologies*. 2023. № 52. С. 94–103. URL: <https://doi.org/10.36910/6775-2524-0560-2023-52-12> (дата звернення: 15.12.2023).

11. Олійник Т. М. Методи і засоби проектування архітектури комп'ютерних систем з використанням IoT компонентів : master's thesis. 2018. URL: <http://elartu.tntu.edu.ua/handle/lib/26478> (дата звернення: 15.12.2023).

12. Trends in the development of systems for monitoring the state of water bodies and the development of a Geo-information system Dniester on the basis of an industrial IoT / L. Zamikhovskyi та ін. *Pidvodni tehnologii*. 2021. № 11. С. 67–77. URL: <https://doi.org/10.32347/uwt2021.11.1502> (дата звернення: 15.12.2023).