НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
Комп'ютерних систем, мереж та кібербезпеки

_____ Касаткін Д.Ю., доц., к.пед.н.

(підпис) (ПІБ, вчене звання і ступінь)

«___»_____2025 р

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка засобів інтелектуалізації граничних пристроїв IoT»

Спеціальність 123 «Комп'ютерна інженерія»

Гарант освітньої програми

_____ _____ Нікітенко Є.В.
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ _____ Коваленко О,Є.
(Науковий ступень та вчене звання) (підпис) (ПІБ)

Виконав _____ Масло Д.А
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І
ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
Комп'ютерних систем, мереж та кібербезпеки
_____/ Касаткін Д.Ю, доцент, к.пед.н /
підпис
"_"_____2025 р.

ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

Масло Дмитро Андрійович
(прізвище, ім'я, по батькові)
Спеціальність (напрям підготовки): комп'ютерна інженерія

Тема кваліфікаційної бакалаврської роботи: «Розробка засобів інтелектуалізації граничних пристроїв IoT»
затверджена наказом ректора НУБіП України від "___" _____2025 р. №____
Термін подання завершеної роботи на кафедру 28.05.2025

Вихідні дані до кваліфікаційної бакалаврської роботи
Дослідження та розробка системи керування засобами моніторингу та управління елементами граничних пристроїв IoT.
Перелік питань, що підлягають розробці::
- дослідити сучасні системи управління елементами граничних пристроїв інтернету речей;
- проаналізувати архітектуру мережі інтернету речей, технології та протоколи передачі даних;
- дослідити апаратне забезпечення та провести його налаштування;
- реалізувати програмне забезпечення з метою покращення управління елементами IoT;
- здійснити оцінку якості розробленого програмного забезпечення.

Дата видачі завдання "_____"_____2024 р.

Керівник бакалаврської роботи _____ Коваленко О.Є., д.т.н., професор
(підпис) (прізвище та ініціали)
Завдання прийняв до виконання _____ Масло Д.А.
(підпис) (прізвище та ініціали студента)

КАЛЕНДАРНИЙ ПЛАН

п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту(роботи)	Примітка
1	Аналіз технічного завдання	14.04.2025 р.	Виконано
2	Дослідження практичних рішень	18.04.2025 р.	Виконано
3	Реалізація системи	28.04.2025 р.	Виконано
4	Тестування системи	11.05.2025 р.	Виконано
5	Оформлення пояснювальної записки	23.05.2025 р.	Виконано
6	Оформлення графічного матеріалу	24.05.2025 р.	Виконано

Студент _____ __Масло Д.А___

(підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

_____ Коваленко О.Є.

(підпис) (ПІБ)

ЗМІСТ

ВСТУП

РОЗДІЛ 1 ОГЛЯД АРХІТЕКТУРИ МЕРЕЖІ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Екосистема інтернету речей

1.2 Огляд архітектури інтернету речей

1.3 Огляд компонентів архітектури інтернету речей

1.4 Огляд технологій та протоколів передачі даних у мережі інтернету речей

1.5 Протоколи для передачі повідомлень в мережі інтернету речей

1.6 Огляд існуючих засобів моніторингу та управління елементами інтернету речей

1.7 Висновок до розділу

РОЗДІЛ 2 ОПИС І АНАЛІЗ АПАРАТНОЇ ЧАСТИНИ ТА НАЛАШТУВАННЯ ЗАСОБІВ МОНІТОРИНГУ ТА УПРАВЛІННЯ ЕЛЕМЕНТАМИ ГРАНИЧНИХ ПРИСТРОЇВ ІОТ

2.1 Опис вибраної платформи Raspberry Pi

2.2 Установка та налаштування Raspberry Pi

2.3 Підключення зовнішнього диска та модуля Wi-Fi

2.4 Підключення до Raspberry Pi з допомогою ssh. Налаштування з’єднання та встановлення Domoticz

2.5 Аналіз Domoticz API, як способу управління

2.6 Висновок до розділу

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МОНІ- ТОРИНГУ ТА УПРАВЛІННЯ ЕЛЕМЕНТАМИ ГРАНИЧНИХ ПРИСТРОЇВ ІОТ

3.1 Реалізація системи автоматизації збору даних

3.2 Налаштування системи

3.3 Демонстрація роботи

3.4 Висновки до розділу

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ВСТУП

Актуальність теми. Інтернет речей (Internet of Things) – по своїй сутті є мережею мереж, що складається з об’єктів (речей), які мають індивідуальні ідентифікатори та можуть взаємодіяти між собою без втручання ззовні.

На сьогоднішній день до пристроїв IoT відносяться мільярди пристроїв по всьому світу, які підключенні до мережі інтернет. Слід зазначити, що термін інтернет речей, давно перейшов в складніше явище ніж просто набір давачі, які виводили інформацію на екран приймального пристрою. На даний час, давачі об’єднуються в єдину мережу де відбувається обмін даними, обробка, аналітика та керування системою, таким чином формується самостійна система, що потребує мінімального втручання людини, або і взагалі не потребує останнього. Система здатна сама приймати рішення та керувати об’єктом, або системою об’єктів, в залежності від того для чого була розроблена мережа.

Мета і завдання дослідження. Дослідження та розробка системи керування засобами моніторингу та управління елементами граничних пристроїв IoT.

Для досягнення поставленої цілі, необхідно роз’вязати наступні задачі:

- дослідити сучасні системи управління елементами граничних пристроїв
- інтернету речей;
- проаналізувати архітектуру мережі інтернету речей, технології та протоколи передачі даних;
- дослідити апаратне забезпечення та провести його налаштування;
- реалізувати програмне забезпечення з метою покращення управління елементами інтернету речей;
- здійснити оцінку якості розробленого програмного забезпечення.

Об’єкт дослідження - система моніторингу та керування елементами інтернету речей.

Предметом дослідження є засоби та протоколи передачі даних в мережі інтернету речей та системи моніторингу та управління її елементами;

Практичне значення отриманих результатів. Налаштовано апаратні засоби та розроблено програмне забезпечення, яке використовується для моніторингу та керування елементами граничних пристроїв інтернету речей.

РОЗДІЛ 1
ОГЛЯД АРХІТЕКТУРИ МЕРЕЖІ ІНТЕРНЕТУ РЕЧЕЙ

Термін «інтернет речей» завдячує своєю появою Кевіну Ештону, який у 1997 р, застосував технологію радіочастотної ідентифікації (RFID) для керування системою поставок. Завдяки цій роботі його запросили в Масачусетський технологічний інститут, де він з групою колег організував дослідницький консорціум Auto-ID Center. З того часу інтернет речей перейшов від простих радіо-міток до екосистеми.

Першим пристроєм інтернету речей можна вважати автомат з газованою водою, підключений до інтернету в університеті Карнегі-Меллон у 1982 році. Поступово до IoT почали підключати все більше речей. У 1991 році компанія HP представила перший мережевий принтер, згодом у 1993 році до інтернету підключено першу камера у Кембриджському університеті. Поява організації Bluetooth SIG у 1998 році сильно вплинула на розвиток IoT, оскільки технологія Bluetooth споживала набагато менше енергії, що допомогло досягти ліпшої автономності пристроїв. 2000 рік – перші прояви розробленої компанією HP концепції всепроникної комп'ютеризації (Cooltown): HP Labs, система обчислювальних і комунікаційних технологій, які у поєднанні один з одним створюють підключення до Інтернету для людей, місць і об'єктів. У 2005 сформовано звіт міжнародним союзом електрозв’язку (спеціалізована установа ООН), у якому вперше сформульовано прогнози розвитку інтернету речей. Успішна розробка напівпровідникових світлодіодних ламп у 2008 призвела до розвитку концепції «розумного» освітлення, та у свою чергу, «розумного» дому.

Прогнозується, що в найближчі 5-7 років, Інтернет речей захопить кожен сегмент в сфері промисловості, бізнесу, охорони здоров’я і споживчих товарів.

1.1 **Екосистема інтернету речей**

До екосистеми інтернету речей відносять засоби, сервіси і технології, які застосовують в інтернеті речей. Їх можна поділити на такі категорії:

- Сенсори, розумні датчики, давачі: вбудовані системи, операційні системи реального часу, джерела безперебійного живлення, мікро-електромеханічні системи;
- Системи зв’язку з датчиками: система охоплення бездротових персональних мереж становить від 0 см до 100 метрів. Для обміну даними між датчиками застосовують низькошвидкісні малопотужні інформаційні канали, які часто побудовано не на протоколі IP;
- Локальні обчислювальні мережі (LAN): зазвичай, це системи обміну даними на основі протоколу IP, наприклад, 802.11 Wi-Fi-мережу для швидкої радіозв'язку, часто це пірінгові або зіркоподібні мережі;
- Агрегатори, маршрутизатори (routers), шлюзи (gateways), пограничні пристрої (Edge Device): постачальники вбудованих систем, самі бюджетні складові (процесори, динамічна оперативна пам'ять і система зберігання даних), виробники модулів, виробники пасивних компонентів, виробники тонких клієнтів, виробники стільникових і бездротових радіосистем, постачальники міжплатформового програмного забезпечення, розробники інфраструктури туманних обчислень, інструментарій для граничної аналітики, безпеку граничних пристроїв, системи управління сертифікатами;
- Глобальна обчислювальна мережа: оператори стільникового зв'язку, оператори супутникового зв'язку, оператори малопотужних глобальних мереж (Low- Power Wide-Area Network, LPWAN).

Зазвичай застосовуються транспортні протоколи Інтернету для IoT і мережевих пристроїв (MQTT, CoAP і навіть HTTP);

- Хмара: інфраструктура в якості постачальника послуг, платформа в якості постачальника послуг, розробники баз даних, постачальники послуг потокової і пакетної обробки даних, інструменти для аналізу даних, програмне забезпечення в якості постачальника послуг, постачальники озер даних, оператори програмно-визначених мереж / програмно-визначених параметрів, сервіси машинного навчання;

- Сервіси аналізу даних: величезні масиви інформації передаються в хмару. Робота з великими обсягами даних і отримання з них користі - це завдання, що вимагає комплексної обробки подій, аналітики і прийомів машинного навчання;

- Безпека (security): при зведенні всіх елементів архітектури воєдино постають питання кібербезпеки. Безпека стосується кожного компонента: від датчиків фізичних величин до ЦПУ і цифрового апаратного забезпечення, систем радіозв'язку і самих протоколів передачі даних. На кожному рівні необхідно забезпечити безпеку, достовірність і цілісність. У цьому ланцюзі не повинно бути слабких ланок, оскільки Інтернет речей стане головною мішенню для атак хакерів в світі.

Загалом, елемент інтернету речей можна поділити на чотири категорії[1-2]. Це сервіси (Services), апаратна частина (Hardware), набір правил (Rules) та програмна частина (Software). Ці категорії також можна розділити на підгрупи за призначенням (див. рис. 1.1).

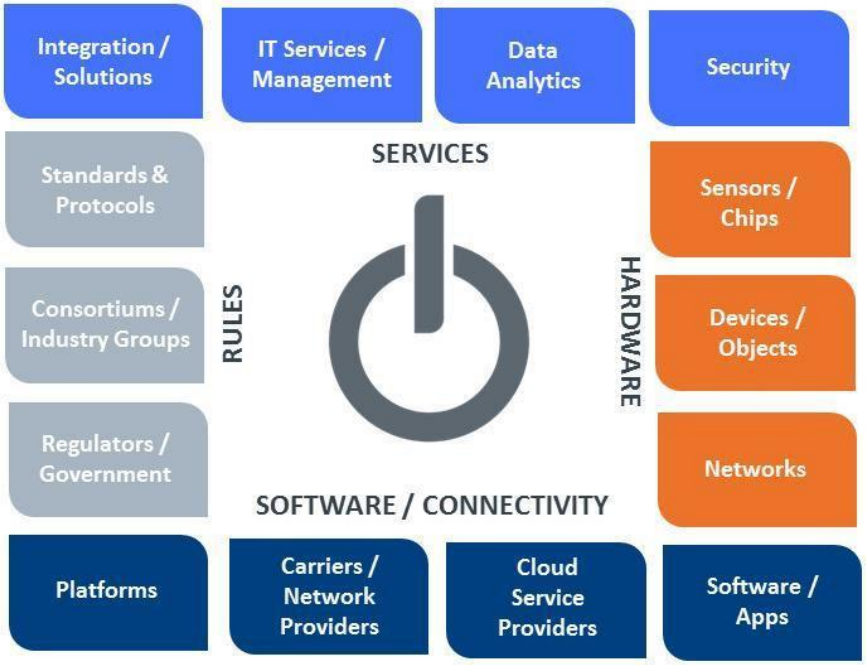


Рис. 1.1. зв’язок між елементами інтернету речей

1.2 **Огляд архітектури інтернету речей**

Архітектура IoT може відрізнятись в залежності від призначення та реалізації. Та загалом вона схожа на типову АСУТП (Автоматизована Система Управління Технологічними Процесами) (рис 1.2.).

Взаємодія з середовищем відбувається через датчики (sensors) та виконавчі механізми (actuators), аналогічно, як це робиться в АСУТП для будь-якого об’єкту керування. Ці датчики разом з усією інфраструктурою для інтеграції з рівнем обробки подій через мережу Internet формують так звану граничну область (Edge).

Події (дані), котрі надходять із граничної області, зберігають і опрацьовують відповідно до задачі (рівень обробки подій і аналітики, event processing, Platform).

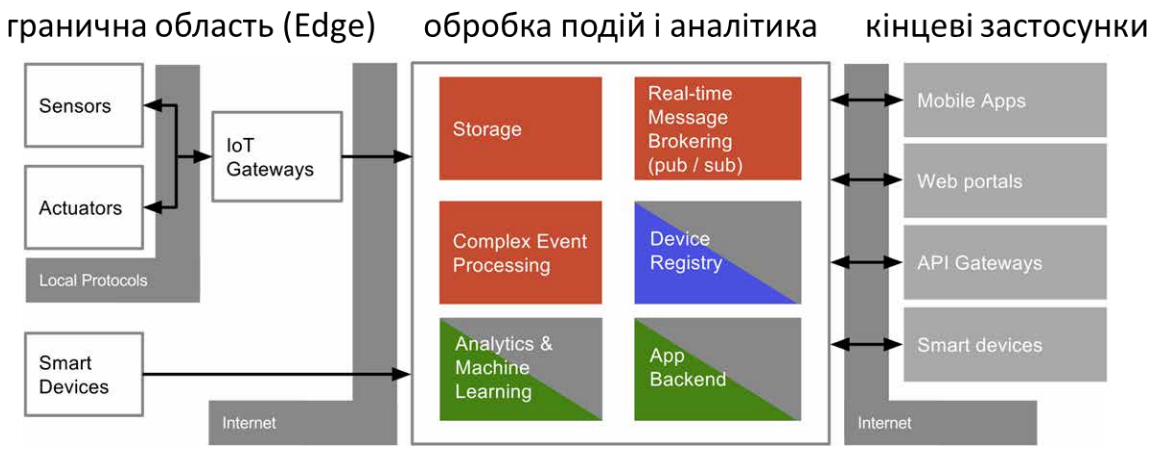


Рис. 1.2. Приклад архітектури IoT

Дані що поступають з граничної області зберігаються і обробляються відповідно до задачі. На цьому рівні дані зберігають (storage), опрацьовують (Event Processing), надсилають потрібним додаткам (Real-Time Message Brokering, Stream Processing). Додатково на цьому рівні здійснюють адміністрування та керування пристроями з граничної області (Device Registry, Edge Device Management). На основі подій (даних) проводять машинне навчання (Machine Learning), що дозволяє зробити певні висновки про об’єкт з використанням аналітичних сервісів (Analytics). Цей рівень, як правило реалізовано, з використанням хмарних (Cloud) або туманних (Fog) обчислень. Якщо провести аналогію с АСУТП, то це рівень контролерів та SCADA (за виключенням функцій HMI). Отримання результатів, контроль, віддалене керування та адміністрування системи здійснюють через кінцеві застосунки з використанням мережі Internet.

На рис. 1.3 показано архітектуру, схожу до наведеної вище, однак у вигляді сервісів. Тут у вигляді датчиків представлено область Edge, також до неї віднесено Device Hub/Gateway (збір та маршрутизація даних) та Device Management (керування пристроями), котрі частково виконуються як хмарні обчислення так і на граничних пристроях. Усі функції збереження

та первинного опрацювання подій (даних) зведено до Data Management. Усі інші функції обробки, в тому числі аналітичні, показано як додатки PaaS, котрі взаємодіють з сервісами керування даних через API (Application Program Interface).

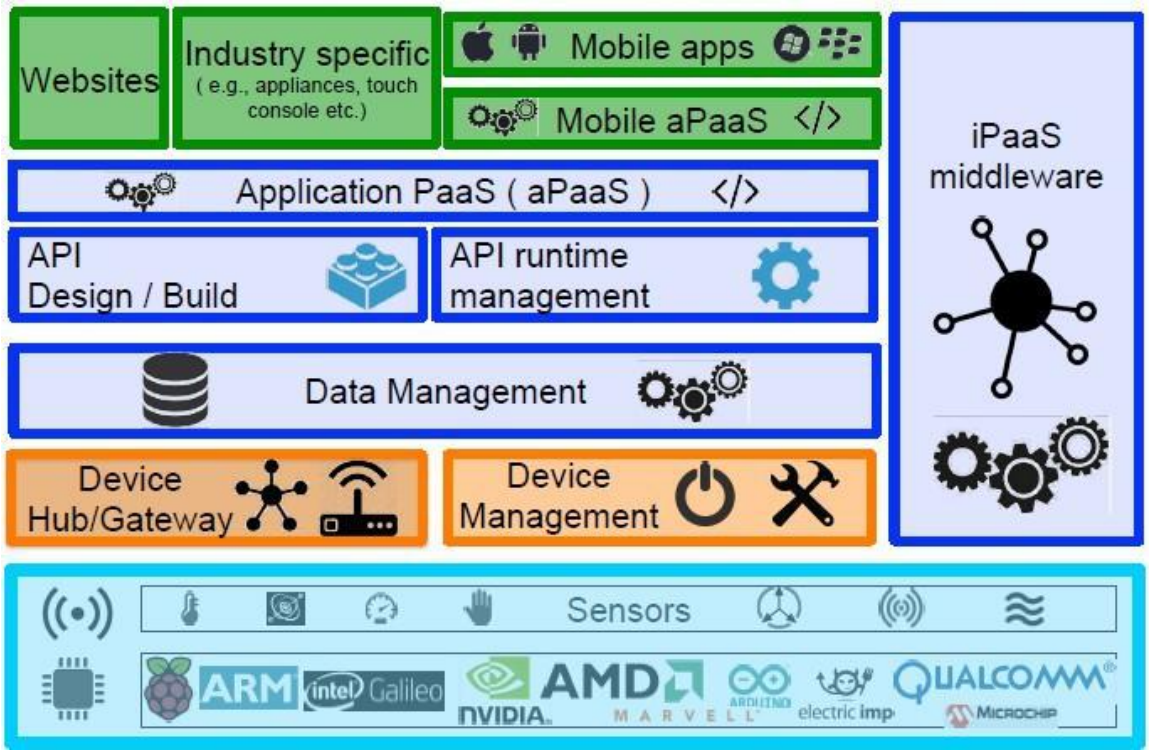


Рис. 1.3. Архітектура інтернету речей представлена у вигляді сервісів

Ще один приклад архітектури інтернету речей, вже на рівні модулів, показано на рис. 1.4. Як видно з цього рисунка, усі наведені архітектури мають спільні риси: наявність трьох рівнів, подібні функції, наявність хмарних обчислень, використання Інтернету як інтеграційного рівня.

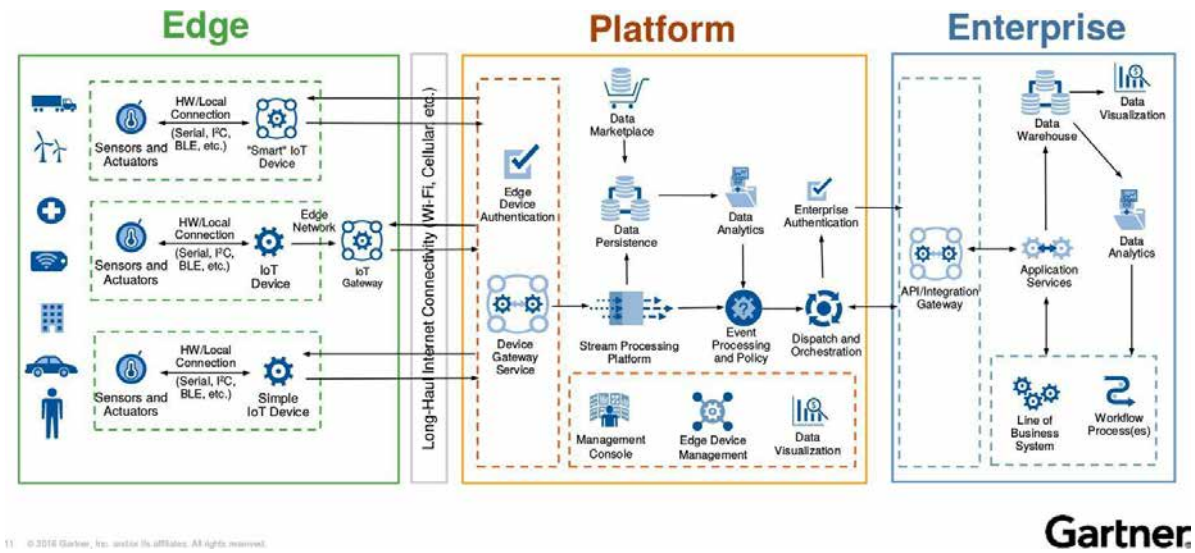


Рис. 1.4. Архітектура інтернету речей на рівні модулів

1.3 **Огляд компонентів архітектури інтернету речей**

1.3.1. Датчики та живлення. Інтернет речей починається або закінчується однією подією: простий рух, зміна температури або, можливо, важіль замикає замок. На відміну від багатьох існуючих IT-пристроїв, Інтернет речей здебільшого пов'язаний з фізичною дією або подією. Він формує реакцію на якийсь чинник реального світу. Іноді при цьому один-єдиний датчик може згенерувати величезний обсяг даних, наприклад, акустичний датчик для профілактичного огляду обладнання. В інших випадках, всього одного біта даних достатньо, щоб передати життєво важливі відомості про стан здоров'я пацієнта (рис. 1.5.). Якою б не була ситуація, системи датчиків еволюціонували і, відповідно до закону Мура, зменшилися до субнанометрових розмірів і стали значно дешевшими. Саме до цього апелюють ті, хто прогнозує, що до Інтернету речей будуть підключені мільярди пристроїв, і саме тому ці прогнози виправдаються.

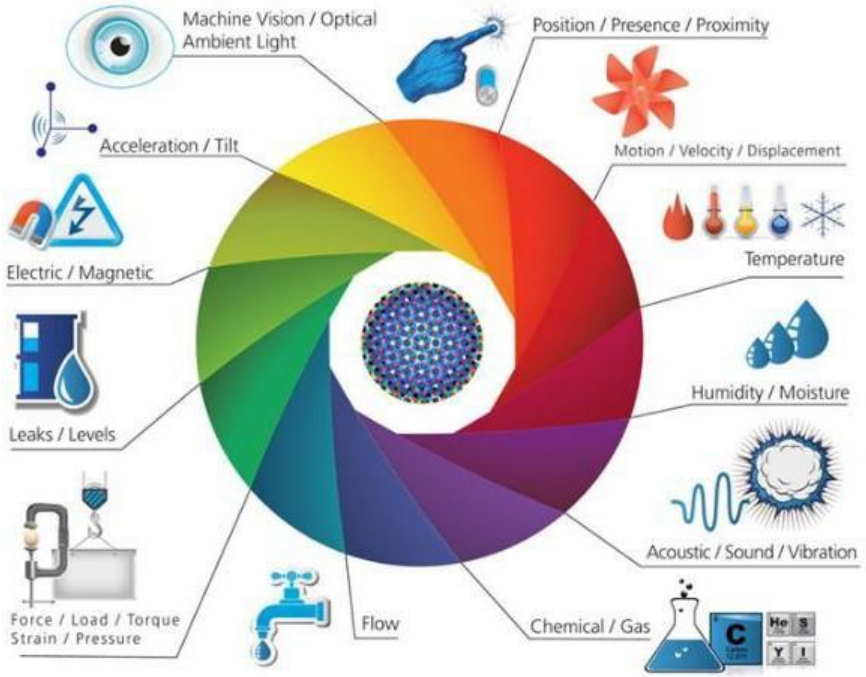


Рис. 1.5. Схематичне зображення сил та явищ, для обробки яких може використовуватись інтернет речей

Тому, розглядаючи інтернет речей, необхідно розглядати мікроелектромеханічні системи, датчики і інші типи недорогих граничних пристроїв та їх електрофізичних властивостей. Також це стосується силових і енергетичних систем, необхідних для живлення цих граничних пристроїв. Не можна вважати, що граничні пристрої забезпечуються енергією за замовчуванням. Мільярди маленьких датчиків все одно потребують великої кількості енергії. З питанням живлення також пов’язані питання організації хмарних сервісів IoT[1].

1.3.2. Передача даних. Велику увагу при розробці IoT приділяють встановленню з'єднання і роботі мереж. Інтернету речей не існувало б без надійних технологій передачі даних з найвіддаленіших і несприятливих областей в найбільші центри збору даних компаній Google, Amazon, Microsoft та IBM. Словосполучення «Інтернет речей» містить слово «Інтернет», тому необхідно вивчати питання, що стосуються мережних

технологій, обміну даними та навіть теорії сигналів. Базова опора Інтернету речей - не датчики і не програми, а можливість встановити з'єднання.

Передача даних і встановлення мережевого з'єднання базуються на системах зв'язку ближньої дії - персональних мереж (PAN), зазвичай побудованих без дотримання правил IP-протоколу. Це можуть бути як дротові, так і бездротові мережі. До бездротових IoT-мереж/протоколів, як правило, відносять протоколи Bluetooth, mesh-мережі, Zigbee, Z-Wave (рис. 1.6), для IIoT (Industrial Internet of Things) - також Wireless Hart та ISA100[2].

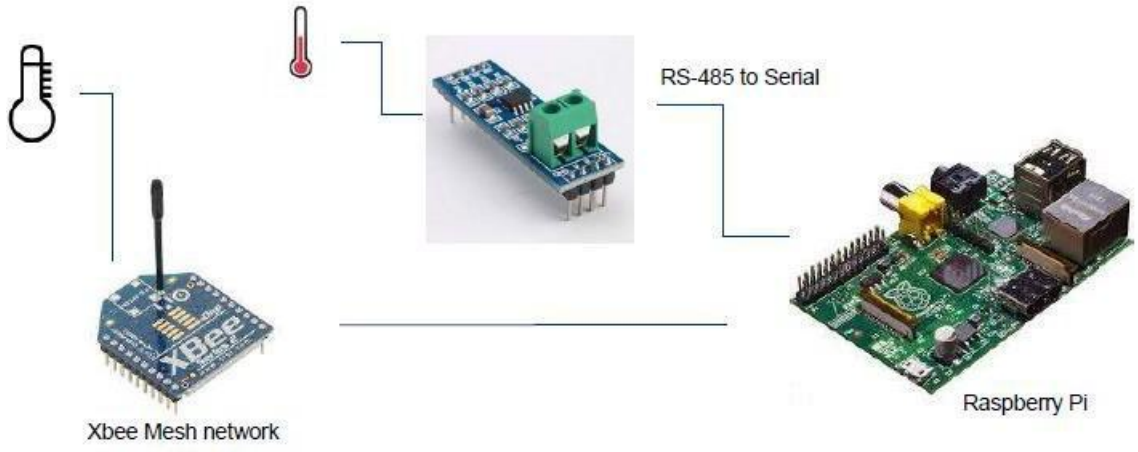


Рис. 1.6. Приклад пристроїв передачі даних в мережі PAN

Крім PAN, застосовують бездротові локальні мережі та системи зв'язку на основі IP-протоколу, включаючи широкий діапазон Wi-Fi-мереж на основі стандартів IEEE 802.11, 6LoWPAN і технології Thread. Нерідко користуються телекомунікаціями на основі стільникових стандартів (3G, 4G LTE) і новими стандарти, котрі забезпечують роботу Інтернету речей і міжмашинну взаємодію, такими як Cat-1 і Cat-NB, а також пропрієтарними протоколами LoRaWAN і Sigfox, котрі застосовують саме для IoT (рис 1.7).

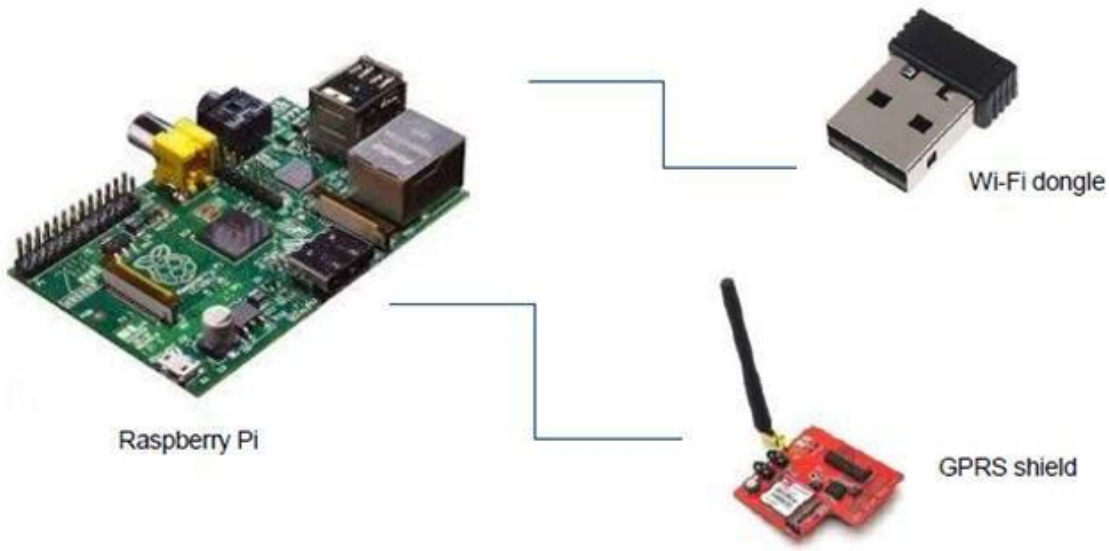


Рис. 1.7. Приклади пристрої для передачі даних в мережі LAN

1.3.3. Маршрутизація. Для передачі даних від датчиків в Інтернет-простір необхідно дві технології: маршрутизатор-шлюз і опорні інтернет-протоколи, що забезпечують ефективність обміну даними. Маршрутизатор особливо важливий в таких аспектах, як безпека, управління і напрям даних. Граничні маршрутизатори (Edge routers) керують і стежать за станом відповідних mesh-мереж, а також вирівнюють і підтримують якість даних. Також велике значення надають конфіденційності та безпеці даних. Маршрутизатор відіграє важливу роль у створенні віртуальних приватних мереж, віртуальних локальних мереж і програмно-визначених глобальних мереж. Вони можуть містити тисячі вузлів, котрі обслуговує єдиний граничний маршрутизатор, що служить розширенням для хмари (edge device)[2].

На цьому рівні застосовують низку протоколів, необхідних для обміну даними між вузлами, маршрутизаторами і хмарними сервісами у межах IoT-системи. Інтернет речей відкрив дорогу новим IoT-протоколам, які виходять на один рівень з традиційними протоколами HTTP і SNMP, які застосовують уже декілька десятків років. Для передачі IoT-даних потрібно

мати ефективні, енергозберігаючі протоколи з малою затримкою, здатні легко і безпечно відправляти дані у хмару і з неї. Зокрема, тут застосовують такі протоколи, як MQTT, AMPQ і CoAP.

1.3.4. Туманні і граничні обчислення, аналітика і машинне навчання. На цьому етапі необхідно вирішити, що робити з потоком даних, що надходять у хмарний сервіс з граничного вузла (Edge Device). Щоб навчитися правильно оцінювати, як система буде розвиватися і рости, необхідно розібратися у всіх нюансах архітектури хмарних систем, а також, як впливає запізнювання на IoT-систему. Крім того, не все треба відправляти у хмару. Пересилання всіх IoT-даних обходиться значно дорожче, ніж їх обробка на кордоні мережі (граничні обчислення, Edge Computing) або включення граничного маршрутизатора в зону, яку обслуговує хмарний сервіс (туманні обчислення, Fog computing). Туманні обчислення також стандартизують, зокрема, існує стандарт туманних обчислень, приміром, архітектура OpenFog.

Дані, котрі отримано шляхом перетворення аналогового фізичного впливу у цифровий сигнал, можуть мати велику вагу. Саме тут у гру вступають засоби аналітики і процесори правил IoT-системи. Ступінь складності введення в дію IoT-системи залежить від того, яке рішення проектують. У деяких ситуаціях все досить просто: наприклад, коли на граничний маршрутизатор, який контролює декілька датчиків, потрібно встановити простий процесор правил, що відслідковує аномальні скачки температури. Інша ситуація - величезну кількість структурованих і неструктурованих даних у режимі реального часу передають у хмарне озеро даних, що вимагає високої швидкості обробки (для прогнозної аналітики) і довгострокового прогнозування на базі високотехнологічних моделей машинного навчання, таких як рекурентна нейронна мережа у пакеті аналізу сигналів з кореляцією в часі. Тут є певні проблеми і складнощі аналітики, які вирішують різними підходами і методами, наприклад,

складними обробниками подій, байесівськими мережами і формування нейронних мереж.

1.3.5. Загрози і безпека в IoT. Багато IoT-систем не обмежуватимуться безпечним простором будинку або офісу. Вони розташовуватимуться в громадських місцях, у дуже віддалених областях, у рухомих транспортних засобах або навіть всередині людини. Інтернет речей - величезна єдина ціль для будь-яких видів хакерських атак. Вже виявлено величезну кількість направлених на IoT-пристрої навчальних атак, добре організованих зломів і навіть вразливостей у системі безпеки національного масштабу. Розробник IoT рішень повинен знати особливості таких вразливостей і способи їх усунення, стандартні заходи, спрямовані на захист Інтернету речей або будь-якого компонента мережі.

1.4 **Огляд технологій та протоколів передачі даних у мережі інтернету речей**

1.4.1 Технології та протоколи передачі на малі відстані. Оскільки парадигма інтернету речей передбачає, що елементи знаходитимуться на невеликій відстані один від одного, та можуть бути не підключені до постійного джерела енергії. Отже, для таких пристрої потрібно і специфічні технології, і протоколи передачі даних. Ось декілька популярних тепер технологій:

- Z-Wave - радіо технологія, що працює на частоті до 1ГГц та підходить для передавання простих команд з низькою затримкою;
- NFC - дозволяє передавати дані, застосовують на відстані до 20 см;
- RFID - технологія ідентифікації з допомогою радіосигналів;
- BLE - варіація технології Bluetooth з низьким енергоспоживанням;

- Wi-Fi HaLow - працює на частоті 900 МГц, має електроспоживання на рівні Bluetooth, та вищу, в порівнянні з Bluetooth, швидкість передачі даних.

Далі розглянемо технології детальніше, та визначимо, для чого їх потрібно.

1.4.1.1 Z-Wave. Бездротова радіо технологія з низьким енергоспоживанням. Z-Wave працює у діапазоні частот до 1 ГГц. ЇЇ оптимізовано для передавання простих команд управління з досить малими затримками (перемикання каналів, вимикання приладів, т. ін)[3].

Такі частоти вибрано не випадково, а для того, щоб зменшити кількість перешкод від інших технологій, якими вже користуються багато людей. Приміром, Wi-Fi працює на частотах 2,4 ГГц і 5 ГГц.

Хоча Z-Wave - складна система, нею дуже проста користуватися, а також вона ще є енергозберігаючою та економить наш дорогоцінний час. Система, що працює за допомогою дистанційного керування і користується радіохвилями малої потужності, є також і надійною майже на 100%, оскільки, ця мережа покриває всі області будинки, радіохвилі можуть проходити через стіни, поверхи та меблі,.

1.4.1.2 NFC. Near Field Communication, або система зв’язку на невеликих відстанях, призначена для обміну різною інформацією, наприклад, картинками, музичними файлами, номерами телефонів, або ключами цифрової авторизації між двома розташованими близько один до одного пристроями з підтримкою NFC. Це можуть бути смарт-картки, будь-які портативні пристрої, а також зчитувальні пристрої RFID. Такою технологією можна користуватися як ключем доступу до даних або служб[4].

Технологія також дозволяє передавати дані не тільки від активного пристрою до пасивного, але і між двома активними пристроями. NFC застосовують для взаємодії з пристроями радіочастотної ідентифікації RFID. Для забезпечення сумісності між картами RFID та мобільним

телефоном різних виробників перевіряють цифровий протокол і вимірюють усіх важливих властивості радіочастотного сигналу: тимчасові характеристики, чутливості та амплітуди приймача в активному режимі, частоти несучої амплітуди сигналу.

При передачі інформації від активного пристрою до пасивного пристрою застосовують амплітудну маніпуляцію ASK. При обміні обидва пристрої рівноправні. Кожен пристрій має власне джерело живлення, тому сигнал несучої відключають одразу після закінчення передачі (рис 1.8.).

Рис. 1.8. Принцип роботи NFC.

Внаслідок індуктивного зв'язку між опитуваним і прослуховуючим пристроями, пасивний пристрій впливає на активний. Зміна опору прослуховуючого пристрою викликає зміну амплітуди або фази напруги на антені опитуваного пристрою. Після цього, відбувається з'єднання двох пристроїв та передача даних. Як тільки ми роз'єднуємо два пристрої більш, ніж на 20 см, розривається електромагнітний зв'язок, і дані перестають передаватися автоматично.

Насправді, NFC можна вважати продовженням вже досить відомої технології радіочастотної ідентифікації RFID [18]. RFID знайдемо всюди у безконтактних картках і мітках. Однак, технологія NFC може не тільки зчитувати інформацію з будь-яких пасивних електронних міток, але, здатна забезпечувати двосторонній бездротовий зв'язок між пристроями.

1.4.1.3 RFID. Radio Frequency IDentification - метод автоматичної ідентифікації об’єктів, у якому за допомогою радіосигналів зчитують або записують дані, котрі зберігають у транспондерах, або RFID - мітках [19]. Будь-яка RFID-система складається із зчитувального пристрою (зчитувач, рідер) та транспондеру (він же RFID-мітка, іноді також називають RFID-тег)[5].

Більшість RFID-міток складається з двох частин. Перша - інтегральна схема для обробки та зберігання інформації, демодулювання та модулювання радіочастотного сигналу і деяких інших функцій. Друга - антена для прийому і передачі сигналу. А також, для роботи цих міток потрібно програмне забезпечення — програми, за допомогою яких збирають інформацію та аналізують, отриману із RFID-міток.

Мітки бувають двох видів - активні та пасивні:

- Активні мітки мають власне джерело живлення, тому вони можуть самі посилати сигнал і зчитуватися з досить великої відстані;
- Пасивні мітки не мають власного джерела енергії і активізуються тільки після того, коли надходить сигнал до пристрою зчитування, і тоді передають записану в них інформацію.

RFID-мітки (рис. 1.9) можуть керувати товарними запасами або відстежувати часу на спортивних змаганнях [5]. Магнітні мітки не замінюють штрих-коди, а доповнюють їх можливістю дистанційного зчитування. Мітками можуть маркувати велику рогату худобу для запису інформації про проходження ветеринарного огляду. Рішення для транспорту допомагають ідентифікувати автомобіль, навіть якщо він рухається з великою швидкістю. Деякі авіалінії користуються мітками для ефективного відстеження великих потоків багажу. Також RFID вбудовують у біометричні паспорти, кредитні картки для безпечного доступу у певні області.

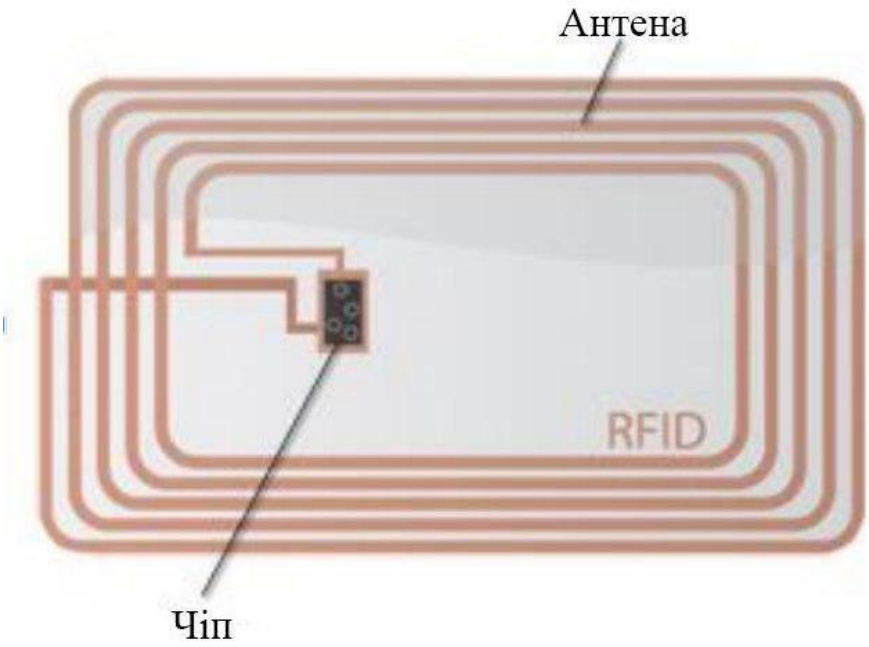


Рис. 1.9. Будова пасивної RFID мітки.

1.4.1.4 BLE, Bluetooth Low Energy. Бездротова технологія з низьким енергоспоживанням Bluetooth (BLE) – частина специфікації Bluetooth, котра починається з покоління Bluetooth 4.0 і тепер закінчується Bluetooth 5.0 [6].

Пристрої, котрі користуються BLE, споживають менше енергії, ніж інші версії Bluetooth-пристрої попередніх поколінь. У багатьох випадках пристрої зможуть працювати більше року на одній невеличкій батарейці типу таблетка без підзарядки. Завдяки цьому, можна користуватися датчиками невеликих розмірів, котрі постійно працюватимуть та взаємодіятимуть із іншими пристроями.

BLE призначено для тих пристроїв, котрі мають невеликі розміри, тобто для пристроїв, у яких важлива компактність і куди не можна встановити повноцінний акумулятор або батарею великого об'єму [6].

Bluetooth LE споживає в 10-20 разів менше енергії і здатний передавати дані в 50 і більше разів швидше та на відстані більше 100 метрів, ніж класичні Bluetooth рішення. Крім перерахованих вище переваг, BLE має високу безпеку, надійність, низьку затримку при підключенні та низьку споживчу потужність.

Ще одна важлива особливість стандарту полягає в адаптивності переналаштування частоти, тобто, відбувається захист від помилок при передачі сигналу, BLE швидко змінює свою робочу частоту, вибираючи найоптимальнішу для усунення перешкод, проблем переповнення і для зниження інтерференції.

Специфікацію Bluetooth 5.0 створено з орієнтацією на Інтернет речей. Це остаточно показало, що стандарт прагне "захопити" ринок пристроїв.

Порівняно із попередньою версією 4.0, підвищено швидкість передачі даних майже до швидкостей HSPA і LTE ранніх версій, тоді як енергоспоживання залишилося у колишніх показниках. Важливим показником для побудови мереж Інтернету речей є енергоефективність. В даний момент така специфікація є мало поширеною через те, що вона з'явилася нещодавно. Bluetooth 5, як і всі попередні версії, має зворотну сумісність. Цілком можливо, що через декілька років кожний мобільний пристрій підтримуватиме п'яту версію цього стандарту, що є найважливішою перевагою такої технології над іншими.

1.4.1.5 Wi-Fi HaLow. Протокол бездротової мережі, котрий опубліковано у 2017 році як доповнення до стандарту бездротової мережі IEEE_802.11 [7]. Протокол працює на частоті 900 МГц, котру не потрібно ліцензувати і забезпечує розширений діапазон Wi-Fi мереж, порівняно із звичайними мережами Wi-Fi, котрі працюють у діапазонах 2,4 ГГц і 5 ГГц. Його низьке енергоспоживання також є перевагою, котра дозволяє створювати великі групи станцій або датчиків, які взаємодіють і поширюють сигнали, підтримуючи концепцію інтернету-речей (Internet of

Things, IoT). Низьке енергоспоживання протоколу конкурує з Bluetooth і має додаткову перевагу - вищі швидкості передачі даних і більш широкий діапазон покриття (рис 1.10).

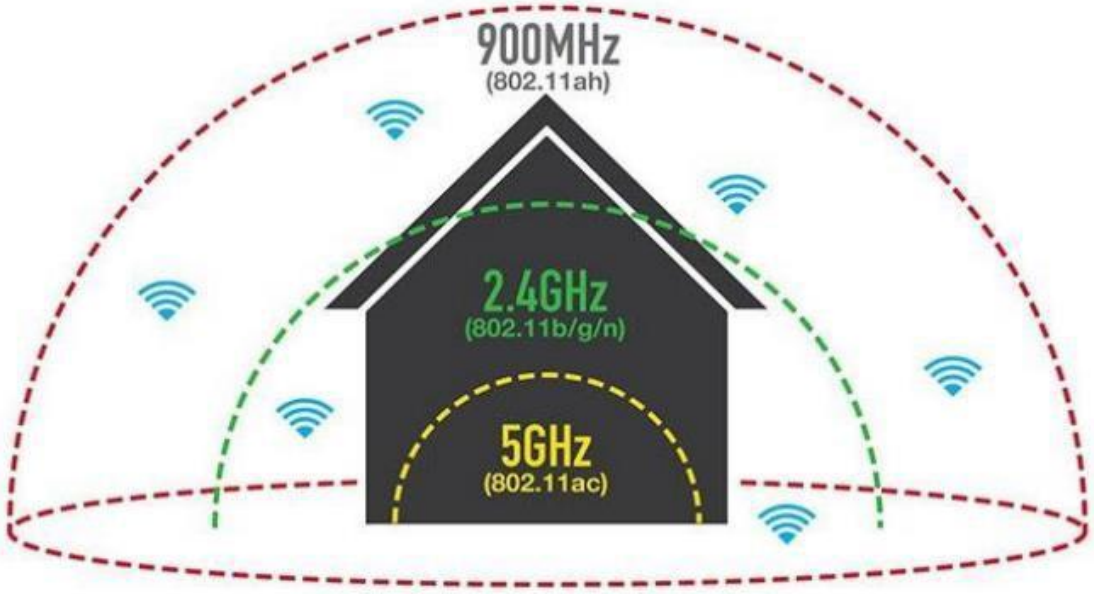


Рис. 1.10. Порівняння різних протоколів Wi-Fi за зоною покриття

Wi-Fi HaLow розширює Wi-Fi в діапазоні 900 МГц, надаючи можливість з'єднувати пристрої малої потужності, такі, як датчики та портативні комп'ютери. Wi-Fi HaLow успадковує позитивні якості попередніх протоколів, такі як надійний захист інформації, широку сумісність обладнання та простоту встановлення. Пристрої з підтримкою Wi-Fi HaLow також працюватимуть у діапазонах 2,4 та 5 ГГц, що надає можливість інтегрувати їх в екосистему, яка тепер налічує більше 7 млрд пристроїв. Також Wi-Fi HaLow матиме підтримку підключення по IP. Це дозволить працювати з хмарами, що дуже важливо для IoT. Також можна підключати до однієї точки доступу близько 1000 пристроїв.

1.4.1.6. Порівняльна характеристика технологій та протоколів передачі даних на малі відстані в IoT. Порівняльна характеристика мереж ближньої дії наведена у табл. 1.1.

Таблиця 1.1

Порівняння основних технічних характеристик мереж з низькою дальністю дії

Характеристики	RFID	NFC	BLE	Z-Wave	Wi-Fi HaLow
Смуга частот	6/13.5/433/863-870/902-928 МГц 2.4/5-27 ГГц	13.56 МГц	2.4 ГГц	868/915 МГц	Піддіапазон 1 ГГц
Швидкість передачі даних	500 кбіт/с	106/212/424/848 кбіт/с	1 Мбіт/с	9.6,40 та 100 кбіт/с	До 4 Мбіт/с
Радіус дії	0.1 - 5 м	0.1 м	70 м	100 м	100 – 1000 м
Пропускна здатність на канал	10 МГц для 6 МГц 14 МГц для 13.5 МГц 1.74 МГц для 433 МГц 7 МГц для 800 МГц 8 МГц для 2.4 ГГц 5 – 27 ГГц сегментований	Змінна	40 каналів з шириною в 2 МГц	300,400 кГц	1/2/4/8/16 МГц
Модуляція	-	ASK, BPSK	GFSK	FSK/GFSK	BPSK, QPSK, 16-/64-/256-QAM, OFDM
Топологія	Point to Point Point to Multipoint	Peer-to-peer	Single-hop	Mesh	Star
Безпека	Шифрування	Шифрування	AES-128	AES-128	WPA

1.5 **Протоколи для передачі повідомлень в мережі інтернету речей**

Обсяг інформації, що формується одним сенсорним вузлом, порівняно невеликий, однак більшість сервісів Інтернету речей побудовано на принципі обробки інформації від великої кількості клієнтів або давачів, що принципово відрізняється від архітектур, прийнятих в класичних мережах.

Таким чином, стикаємося з новою архітектурою: багато джерел - багато одержувачів, крім того, обсяг трафіку від сенсорного вузла може бути як дуже маленьким, так і дуже великим. У такому випадку звичними прикладними протоколами для передачі повідомлень користуватися не можна.

Базова топологія для передачі повідомлень у мережі інтернету речей має назву “видавець-підписник” (Publisher-Subscriber, або pub/sub) (див. рис. 1.11). У такій схемі введено поняття видавця – джерела інформації та передплатника – одержувача інформації. Термін передплата пов'язаний з певною операцією, котру виконують учасники, з метою отримати інформацію передплатником від конкретного видавця, а також упорядкувати збір інформації – параметри періодичності отримання та аналогічних (в залежності від реалізації) показників.

У даному випадку розглядають ситуацію, коли сенсорний вузол (Node) об'єднує інформацію від багатьох датчиків (наприклад, дані вологості повітря) і направляють її згідно з параметрами передплати або за запитом, або самостійно через певний інтервал часу. Зазвичай самі датчики досить примітивні, їх завдання зводять до постійної передачі інформації про контрольовані параметри. Тому з'являється необхідність об'єднувати датчики у вузли, оснащені мікроконтролерами, які відповідатимуть за зчитування вимірюваних даних і відправку їх за заздалегідь визначеними алгоритмами далі на сервер.

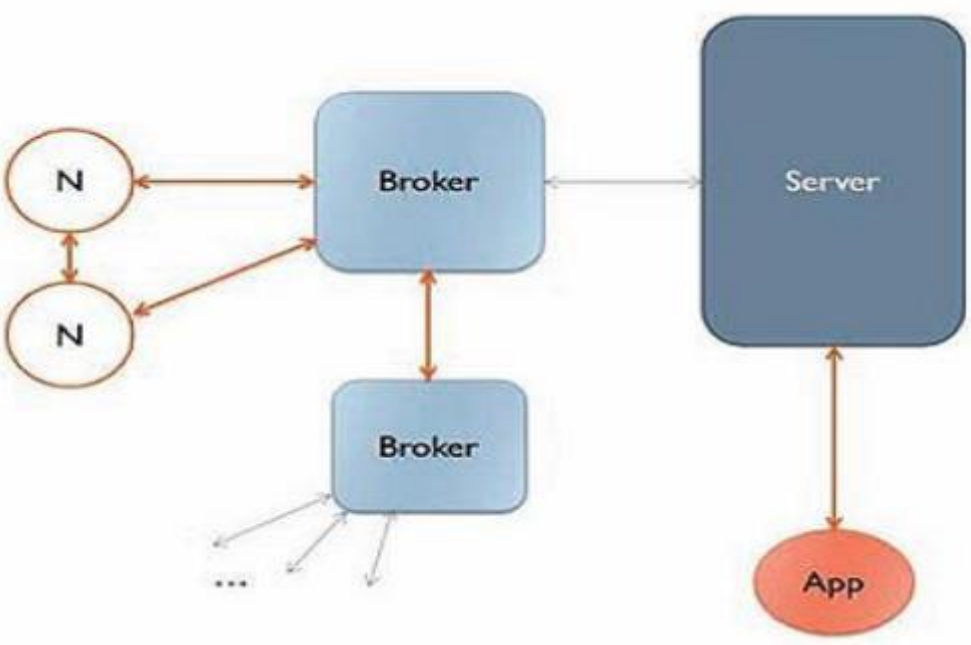


Рис. 1.11 Топологія системи, яка виконує передачу даних між різними вузлами мережі IoT

Також найчастіше для взаємодії клієнта з системою необхідно ще мати клієнтську програму (Application), встановлену на персональному пристрої, яка наочно представлятиме інформації, одержану від датчиків, або вже опрацьовану сервером управління системою. Така топологія також розрахована на включення брокера (Broker). Брокер – сервер, котрий приймає інформацію від видавців і передає її відповідним передплатникам. У складних системах брокер може виконувати також різні операції, пов'язані з аналізом та опрацюванням даних, котрі надійшли на сервер. Брокер може встановлювати пріоритети сполученням і формувати черги для передачі повідомлень. Таким чином, брокер організовує пересилання повідомлень, їх зберігання та фільтрацію. Черга повідомлень – контейнер, або блок, в якому зберігають повідомлення у процесі їх пересилання. При недостатньому ресурсі каналу зв'язку, або якщо одержувач недоступний під

час того, як надсилають повідомлення, черга зберігає повідомлення до того часу, поки його не доставлять до одержувача.

Протоколи, котрі застосовують у мережі інтернету речей:

- DDS – протокол прикладного рівня M2M для систем реального часу;
- XMPP – протокол для миттєвого обміну повідомленнями та інформацією про присутність у режимі, близькому до режиму реального часу;
- CoAP – створений для мереж і пристроїв з обмеженими ресурсами.
- MQTT – легкий, компактний і відкритий протокол обміну даними, створений для передачі даних на віддалених локаціях;
- STOMP – організовує зв'язок з брокером за методом "запит-відповідь";
- SOAP – протокол обміну структурованими та довільними повідомленнями формату XML у розподіленому обчислювальному середовищі.

Далі розглянемо найпопулярніші протоколи.

1.5.1 DDS (Data Distribution Service). Протокол прикладного рівня M2M для систем реального часу, котрий базується на моделі "видавець-передплатник". Основна функція протоколу полягає в тому, щоб здійснити з’єднання пристроїв з іншими пристроями за допомогою шини обміну повідомленнями (див рис. 1.12). Протокол DDS може ефективно та синхронно доставляти мільйони повідомлень в секунду.

Пристрої дають запит на дані інакше, ніж в IT мережах. По-перше, пристрої працюють швидко. Масштаб реального часу часто вимірюють у долях мікросекунд. Пристроям потрібно здійснювати зв'язок з іншими пристроями, використовуючи складні шляхи, тому прості і надійні двоточкові TCP потоки даних обмежують можливості для такої передачі.

Натомість DDS забезпечує деталізований контроль якості обслуговування (QoS), багатоадресну передачу, переналаштовану надійність і всеосяжну надмірність. Крім того, сильною стороною DDS є розгалуження даних. Протокол DDS забезпечує потужні способи фільтрації та відбору даних за адресами призначення, причому число синхронних одержувачів даних може обчислюватися тисячами. Деякі пристрої досить компактні, тому існують полегшені версії протоколу DDS, які працюють в умовах обмеженого обсягу.

Для того, щоб отримувати дані від пристроїв, зіркоподібна мережа зовсім не підходить. Замість цього, DDS реалізує напряму шинний зв'язок між пристроями на базі реляційної моделі даних. ЇЇ називають шиною даних (DataBus), оскільки це мережевий аналог бази даних (database).

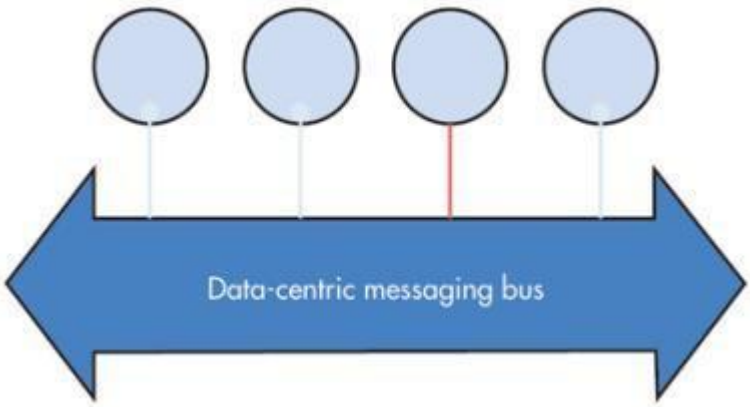


Рис. 1.12 Принцип з’єднання пристроїв за допомогою протоколу DDS

Протокол DDS застосовують у високопродуктивних пристроях. Це єдина технологія, яка забезпечує гнучкість, надійність та швидкість, необхідні для побудови складних додатків реального часу. До таких додатків належать військові системи, вітроелектростанції, інтегровані системи лікарень, системи діагностичної візуалізації, системи супроводження ресурсів, автомобільні системи випробувань і забезпечення безпеки.

1.5.2 MQTT (Message Queue Telemetry Transport) – легкий, компактний і відкритий протокол обміну даними, створений для передачі даних на віддалених локаціях, де потрібно невеликий розмір коду і пропускна здатність каналу обмежена. Перераховані вище вимоги дозволяють застосовувати його в системах M2M (машина-машина)[9-10].

Також існує версія протоколу MQTT-SN (MQTT для мереж датчиків), раніше відома як MQTT-S, призначена для вбудованих бездротових пристроїв без підтримки TCP/IP мереж.

Спрощений процес роботи протоколу MQTT:

Видавець передає повідомлення з певними даними (наприклад, інформація з датчиків вологості) на брокера, вказуючи при цьому тему (Topic), до якої ці дані відносяться (наприклад, "вологість") [].

Брокер аналізує, які із передплатників мають підписку на певні теми, в даному випадку – на тему "вологість".

Передплатникам, які підписані на тему "вологість", брокером буде відправлено повідомлення з інформацією від датчиків вологості.

Таким чином, велика кількість передплатників може підписатися на різноманітні теми і, залежно від таких передплат, отримувати необхідну їм інформацію, не спілкуючись з видавцем безпосередньо.

1.5.3 STOMP (Simple Text Oriented Message Protocol). Простий протокол обміну повідомленнями, що передбачає широку взаємодію з багатьма мовами, платформами та брокерами []. Такий протокол підходить під шаблон "видавець-передплатник" (див рис. 1.13) і за допомогою повідомлень SEND (відправити), SUBSCRIBE (підписатися), UNSUBSCRIBE (відписатися), BEGIN (почати), ABORT (переривати), ACK (підтвердити), NACK (не підтвердити), DISCONNECT (відключити) організовує зв'язок з брокером за методом "запит-відповідь".

Протокол схожий на HTTP, використовує транспорт TCP, є простим текстовим протоколом, що дозволяє клієнтам STOMP спілкуватися з будь

яким брокером повідомлень, котрі підтримують цей протокол. Таким чином, цей спосіб взаємодії, розроблений для обміну повідомленнями між платформою, написаною одною мовою програмування, і клієнтом, програмне забезпечення якого створене іншою мовою. Підтримує велику кількість сумісних клієнтських бібліотек.

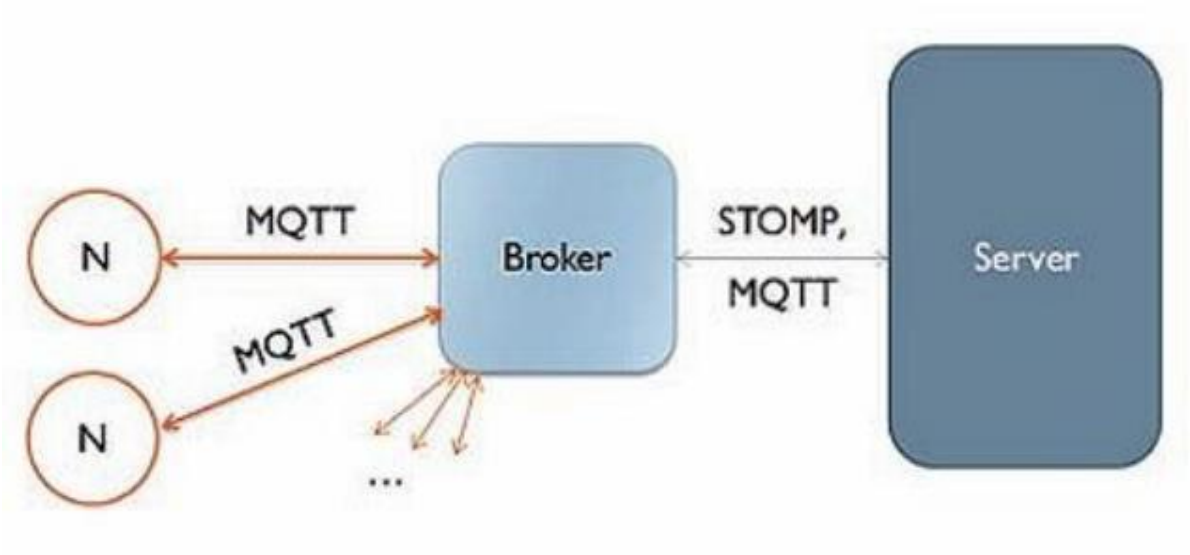


Рис. 1.13. Сегмент мережі, де застосовують протоколи MQTT та STOMP

1.5.4. Огляд операції, що можуть здійснюватись протоколами та їх призначення. З огляду згаданих вище пристроїв виходить, що кожен протокол має свою сферу застосувань. DDS призначено для комунікації між пристроями, де потрібно високу швидкість обміну великою кількістю даних, гнучкість, масштабованість. MQTT застосовують для дуже завантажених мереж, де є велика кількість пристроїв та брокерів, також якщо пропускна здатність каналу обмежена та кількість коду невелика. У свою чергу STOMP потрібно тоді, коли необхідно спростити системи обміну повідомленнями, та уніфікувати повідомлень від різних протоколів.

Далі у табл. 1.2 наведено призначення та операції, котрі можуть виконуватись даними протоколами.

Таблиця 1.2

Призначення та операції, котрі виконуються аналізованими протоколами

Протокол	Призначення	Операції
DDS	Для мереж, що потребують розподіленого навантаження	Процедури отримання та відправки даних
MQTT	Для завантажених мереж з великою кількістю пристроїв та брокером	Процедури обробки публікацій/передплати
STOMP	Для мереж, в яких є можливість застосовувати декілька комбінацій різних протоколів, що потребують простий протокол передачі повідомлень через брокера	Процедури публікацій/передплати. Операції з транзакціями

1.5.5 Висновки після огляду технологій та протоколів передачі даних. Для комунікації в мережі інтернету речей, користуються різними технологіями та протоколами передачі даних, залежно від обставин та середовища, в яких будуватимуть мережу.

Для мережі у межах одного будинку, найліпше підходять технологія передачі даних Wi-Fi HaLow та протокол MQTT. Для мережі PAN найліпше створювати на основі BLE та RFID, технологій, що дозволяють пристроям, котрі знаходяться на невеликій відстані, спілкуватися між собою, та протоколи MQTT та DDS для передачі команд та даних.

Протокол STOMP дозволяє підключатися до різних пристроїв та абстрагуватися від протоколів нижчого рівня, що дозволяє уніфікувати доступу до сервісів.

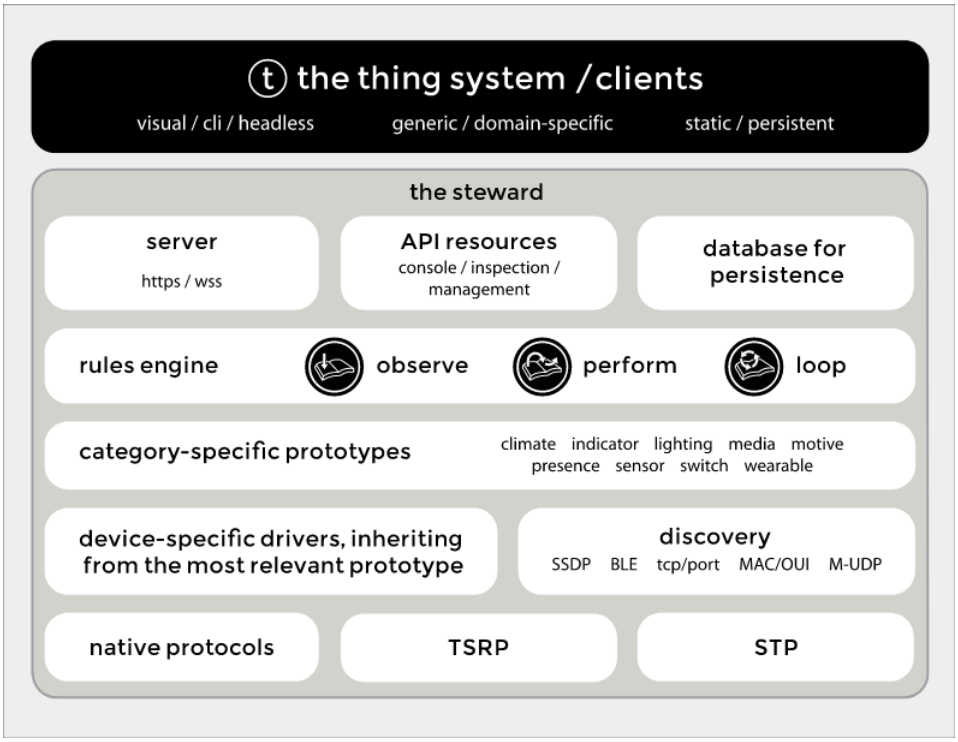
1.6 **Огляд існуючих засобів моніторингу та управління елементами інтернету речей**

1.6.1 The Thing System. Система моніторингу екосистеми інтернету речей, розроблена групою американських інженерів, яку створено разом із Денні Гудманом, автором книг із JavaScript і HTML.

Мета проекту - об’єднати гаджети/елементи інтернету речей, що знаходяться у системах розумного будинку з централізованим управлінням. Пристрої різних виробників часто не можуть взаємодіяти один з одним і працюють не синхронізовано. Щоб розв'язати таку задачу, автори створили ПЗ, яке може працювати з різними мережевими протоколами, гаджетами і клієнтським додатками.

Список пристроїв, котрі підтримує проект, можна знайти на сайті проекту.

Рис. 1.14. Діаграма модулів додатку Steward



Програмне забезпечення Steward, основне ПЗ the thing system, написано на node.js, тому воно портативне та легко розширюється. Воно працюватиме, як на ноутбуці, так на віддаленому сервері або на одноплатному комп'ютері Raspberry Pi.

Переваги такого ПЗ:

- Невеликі розміри;
- Портативність та мультиплатформеність;
- Підходить для різних пристроїв мережі інтернету речей;
- Легко налаштовувати та встановлювати.

Недоліки ПЗ:

- Систему розраховано на пристрої мережі розумного будинку і при розширенні на більш серйозні мережі виникатимуть проблеми з конфігурацією;
- Проблеми з продуктивності при розширенні системи, оскільки платформу написано на node.js;
- Система підтримує незначну кількість пристроїв.

Отже, система ідеально підходить для розумного будинку та простих локальних мереж. Однак при масштабуванні або передачі великої кількості даних можуть виникнути проблеми з продуктивністю.

1.6.2 Domoticz. Система автоматизації розумного будинку з відкритим кодом, котру створено для контролю різних пристроїв та опрацювання вхідних сигналів з різноманітних сенсорів. Система підтримує велику кількість пристроїв різних виробників. Також система має відкритий код, основна частина якого написана на C++, який поширюється під ліцензією "GNU GENERAL PUBLIC LICENSE". Користувацький

інтерфейс є HTML5 веб-інтерфейсом, що автоматично адаптується під різні пристрої[15].

Така система працюватиме на Windows, Apple, Cubieboard, Raspberry Pi та на всіх Unix-подібних операційних системах. Також, система має додатки для мобільних платформ, що дозволяють керувати нею віддалено.

Система дозволяє користувачеві програмувати обробку подій та повідомлень, що надходять з пристроїв та давачів. Програмувати потрібну користувачу логіку можна у візуальному додатку Blockly (див. рис. 1.15).

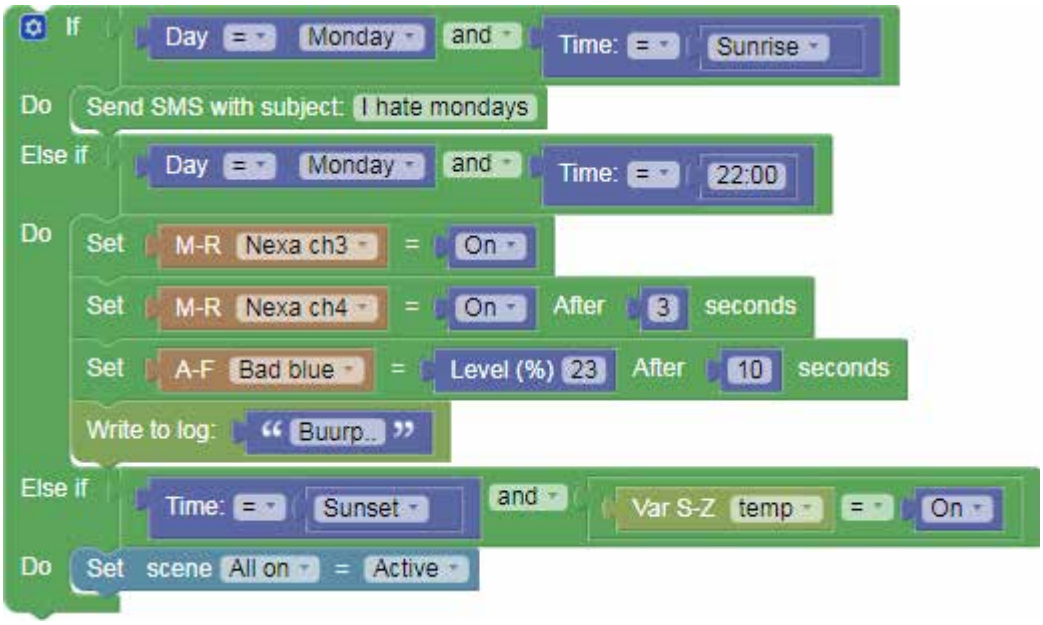


Рис. 1.15. Приклад візуального програмування з допомогою веб-додатку “Blockly”

Переваги системи:

- Мультиплатформність;
- Зручний веб-інтерфейс;
- Підтримка великої кількості вже готових пристроїв;
- Можливість писати драйвери\скрипти для своїх пристроїв;
- Легко встановлювати та налаштувати;
- Легка масштабувати системи для різних завдань;

- Можливість гнучко налаштовувати поведінку пристроїв завдяки програмування з допомогою логічних блоків;
- Відкритість коду системи, що дозволяє самостійно правити і додавати функціонал.

Недоліки системи:

- Оскільки інтерфейс написано на HTML5, то не підтримуються браузери старих версій;
- При встановленні на платформи з малим об’ємом оперативної пам’яті можуть виникнути проблеми з продуктивністю.

Особливістю системи є можливість керувати нею з допомогою асинхронних запитів, що у собі містять JSON. Система містить детальну документацію стосовно усіх запитах, якими можна керувати системою.

1.7 **Висновок до розділу 1**

У розділі розглянуто та проаналізовано архітектуру мережі Інтернету речей. Наведено основні протоколи, котрі застосовують для побудови таких мереж. На малій дистанції для комунікації між пристроями найкраще використовувати техногогії BLE та NFC, оскільки вони надають найкраще співвідношення швидкості передачі даних та енергоспоживання. Для передачі даних на середніх дистанціях найкращим вибором буде Wi-Fi HaLow оскільки при споживанні енергії на рівні Bluetooth має вищу швидкість передачі даних та більший радіус дії.

Також розглянуто існуючі засоби моніторингу та управління елементами інтернету речей. Серед яких можна виділити Domoticz, оскільки він має відкриту кодову базу, підтримує велику кількість пристроїв та має власний програмний інтерфейс, який дозволяє керувати системою віддалено.

РОЗДІЛ 2

ОПИС І АНАЛІЗ АПАРАТНОЇ ЧАСТИНИ ТА НАЛАШТУВАННЯ ЗАСОБІВ МОНІТОРИНГУ ТА УПРАВЛІННЯ ЕЛЕМЕНТАМИ ГРАНИЧНИХ ПРИСТРОЇВ ІОТ

2.1 **Опис вибраної платформи Raspberry Pi**

Raspberry Pi model B - одноплатний комп’ютер, що на початку свого існування призначався для вивчення у школах базових принципів комп’ютерних наук (рис. 2.1). З розвитком мобільних процесорів, став однією з найулюбленіших платформ для ентузіастів, що розробляють додатки для інтернету речей. Основні переваги платформи: ціна, універсальність та відкритість[11].

- Комп’ютер може працювати на базі Unix подібних систем;

- Може відображати Full HD відео;
- Об’єм ОЗУ від 512 МБ;
- Розміри плати 85,6 * 54 * 17 мм;
- Є підтримка композитного відеовиходу RCA і HDMI для підключення монітору, а також 3.5-міліметровий роз’єм для підключення пристроїв відтворення звуку.

Живлення плати може відбуватися декількома способами. Перший це з допомогою роз`єма micro-USB, який приймає 5В. Другий це через універсальні піни. Третій, через адаптер живлення на 5В і 2А. Перші два способи не гарантують стабільне живлення при підключенні великої кількості переферійних пристроїв та датчиків. Третій спосіб найстабільніший, оскільки надає хорошу енергоємність для пристроїв що будуть підключатись[12].

Плата підтримує велику кількість інтерфейсів(рис 2.2):

- UART
- Слот для карт SD, MMC, SDIO
- Роз’єм HDMI
- 3.5мм стерео
- Композитний відеовихід
- 2x USB 2.0
- 10/100Mb RJ45 Ethernet
- Wi-Fi 802.11n
- Bluetooth 4.1
- 16 портів input/output 3.3В
- Інтерфейси I2C і SPI
- Інтерфейс ARM JTAG
- Інтерфейс DSI
- Інтерфейс MIPI CSI-2

Інтерфейси Wi-Fi 802.11n і Bluetooth 4.1 забезпечуються окремою мікросхемою Broadcom BCM43438, яка доставляється на плату через інтерфейс Shield або окремими модулями, що підключаються як периферія.

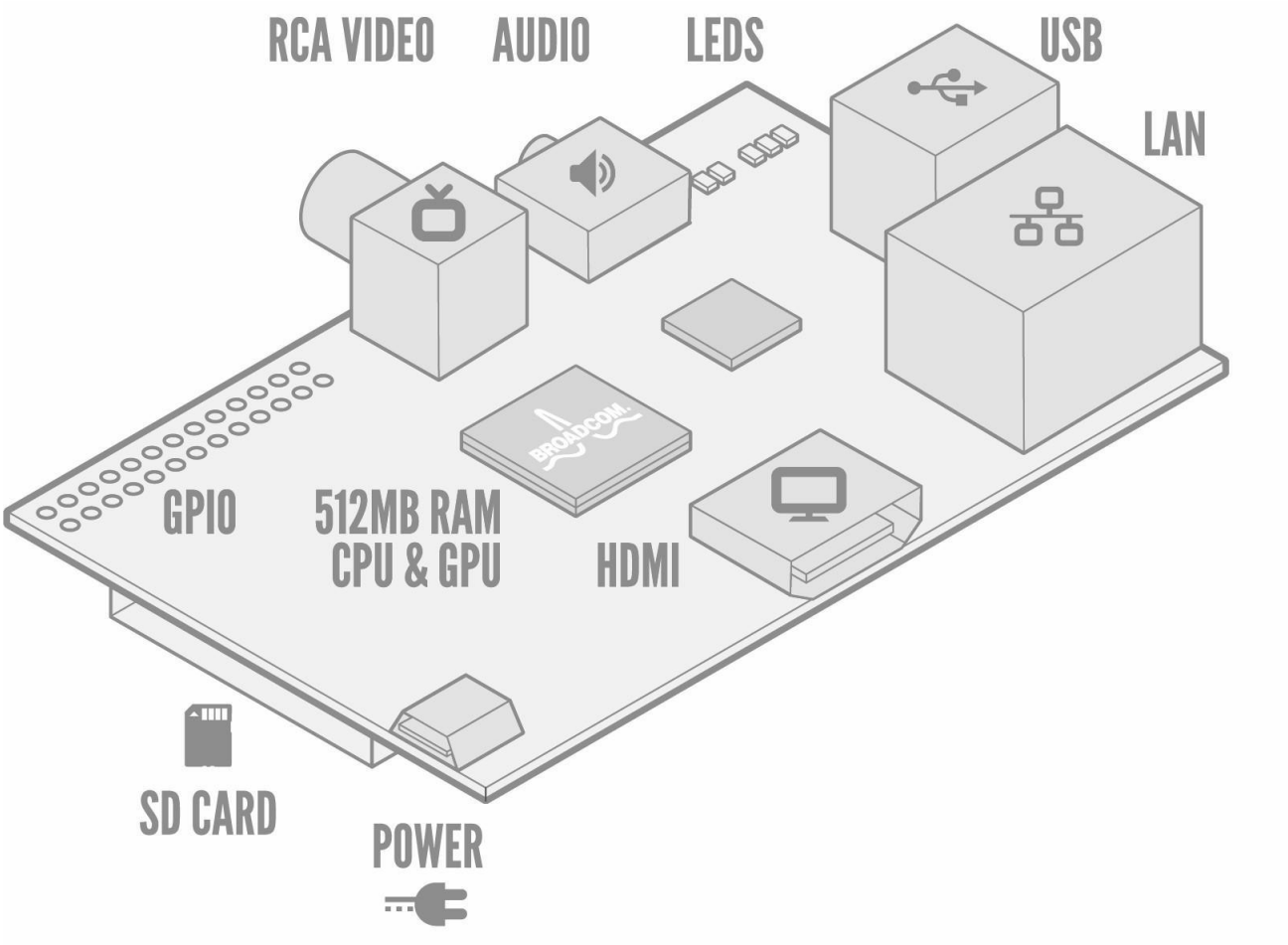


Рис. 2.2. Схема розміщення інтерфейсів на платі

Процесор. Raspberry Pi B використовує 32-розрядний процесор ARM Cortex-A7 900 МГц. Він має кеш першого рівня (L1) 16 Кб, а кеш другого рівня (L2) 128 Кб. Кеш-пам'ять другого рівня використовується насамперед графічним процесором. SoC розміщується під мікросхемою оперативної пам'яті, тому його не видно.

Продуктивність. Під час роботи на частоті 700 МГц, Raspberry Pi забезпечила реальну продуктивність, приблизно еквівалентну 0,041

GFLOPS. На рівні центрального процесора продуктивність аналогічна Pentium II 300 МГц 1997-99. GPU забезпечує 1 Gpixel/s або 1.5 Gtexel/s обробки графіки або 24 GFLOPS загальної обчислювальної продуктивності. Програмне забезпечення. Raspberry Pi працює в основному на операційних системах, заснованих на Linux ядрі. Також можлива установка Windows 10 IOT. Більш того, існують Raspberry з ліцензійною Windows 10 IOT. ARM11 заснований на 6 версії ARM, який завжди підтримує усі різновиди Linux. Для установки операційних систем існує інструмент NOOBS, або інші аналоги для запису образів ОС на носії, що дозволяють записувати на карти пам’яті операційні системи сумісні з Raspberry Pi.

2.2 **Встановлення та налаштування Raspberry Pi**

2.2.1 Підготовка карти пам’яті. Так як у Raspberry Pi немає вбудованої постійної пам'яті, для роботи комп'ютера попередньо необхідно підготувати карту пам'яті - розпакувати на неї образ бажаної операційної системи[12-13]. Для підготовки карти пам’яті потрібно:

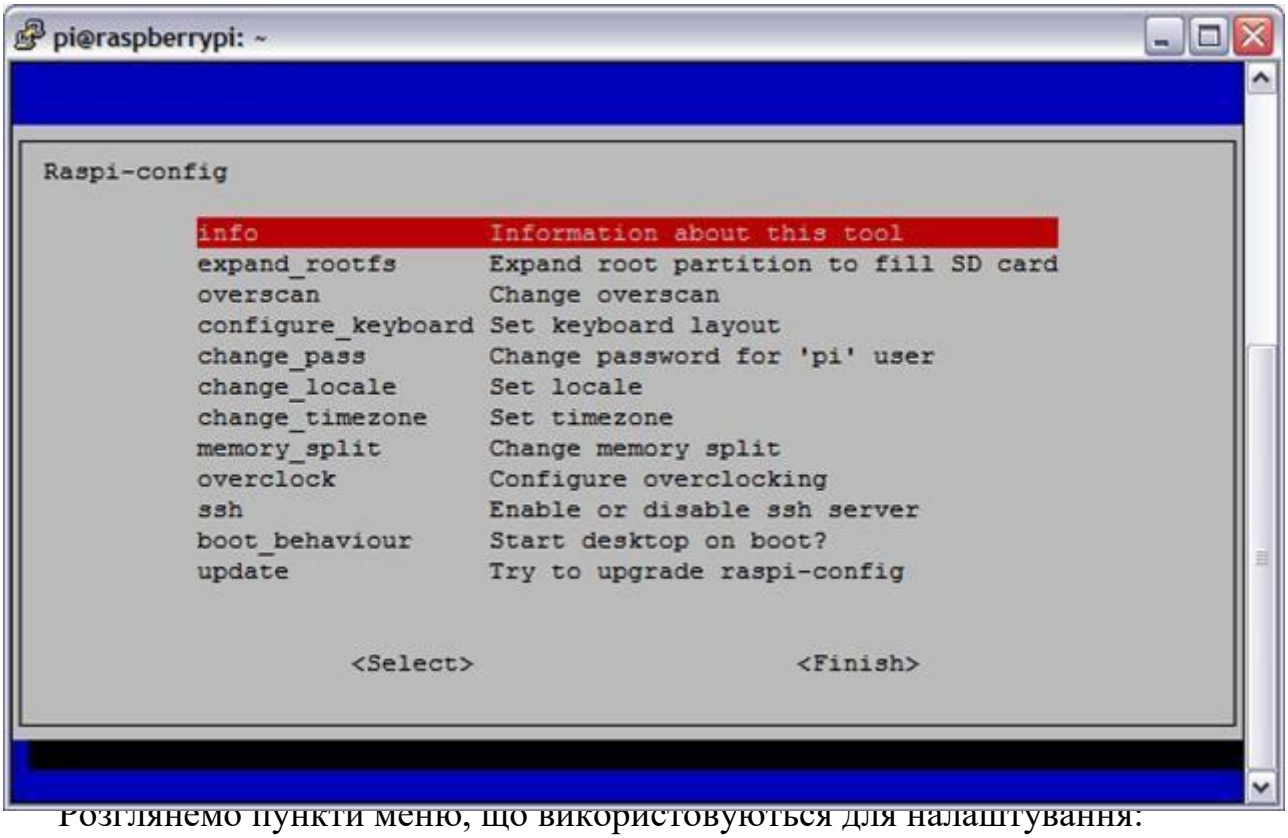
1. SD (MMC/SDIO) карта пам’яті об’ємом від 8 до 32Гб, з високим класом швидкості читання, не нижче 8;
2. Картридер для підключення карти до комп’ютера;
3. Програма Win32DiskImager;
4. Образ операційної системи, буде використовуватись Raspbian.

Далі карту потрібно підключити до комп’ютера з допомогою картрідера, та за допомогою Win32DiskImager записати на карту образ операційної системи.

2.2.2 Перше підключення Raspberry Pi, та первинне налаштування. Для першого запуску та налаштування окрім самої плати та карти пам’яті з операційною системою, знадобиться USB-клавіатура, монітор з HDMI

виходом або TV тюнер з RSA роз’ємом і кабель для підключення, та блок живлення 5В та мінімум 0.5А[13].

Після підключення всіх елементів та ввімкнення живлення, зразу почнеться загрузка операційної системи (рис 2.3).



Розглянемо пункти меню, що використовуються для налаштування:

- expand_rootfs, тут можна збільшити root розмір на весь розмір карти пам'яті;
- configure_keyboard, в цьому пункті ви можете вибрати драйвер клавіатури, якщо варіант за замовчуванням вас не влаштовує;
- change_pass, заміна паролю користувача “pi”, при вводі пароль не відображається, та ввести пароль потрібно двічі;
- change_locate, зміна мови системи;

- change_timezone, зміна часового поясу, плата немає свої часових поясів, тому підгружає їх з інтернету;
- memory_split, розподіл пам’яті;
- overclock, розгін процесора;
- ssh, ввімкнення або вимкнення SSH сервера, рекомендується включити, якщо в майбутньому буде використовуватись віддалене керування;
- boot_behaviour, завантаження візуальної оболонки при завантаженні системи.

Після закінчення налаштувань, потрібно перезавантажити систему, для цього потрібно нажати комбінацію клавіш Ctrl+F, та підтвердити вибір. Після чого плата перезавантажиться.

Далі після загрузки системи потрібно встановити пароль для користувача “root”. Для цього в командному рядку потрібно ввести комаду “sudo passwd root”, та ввести пароль двічі. Для перегляду процесів, що наразі запущені в системі потрібно набрати команду “top” (рис 2.4).

2.3 **Підключення зовнішнього диска та модуля Wi-Fi**

Для безпроводового доступу використовуватиметься Wi-Fi адаптер для Raspberry Pi Auspi Wireless Adaptor– 802.11 B/G/N. Його перевагою є робота у діапазонах b, g, n, сумісність з USB роз’ємом та те що він не потребує окремого живлення[13].

Зовнішній диск можу бути будь якого формату, основне щоб була можливість його підключення через USB.

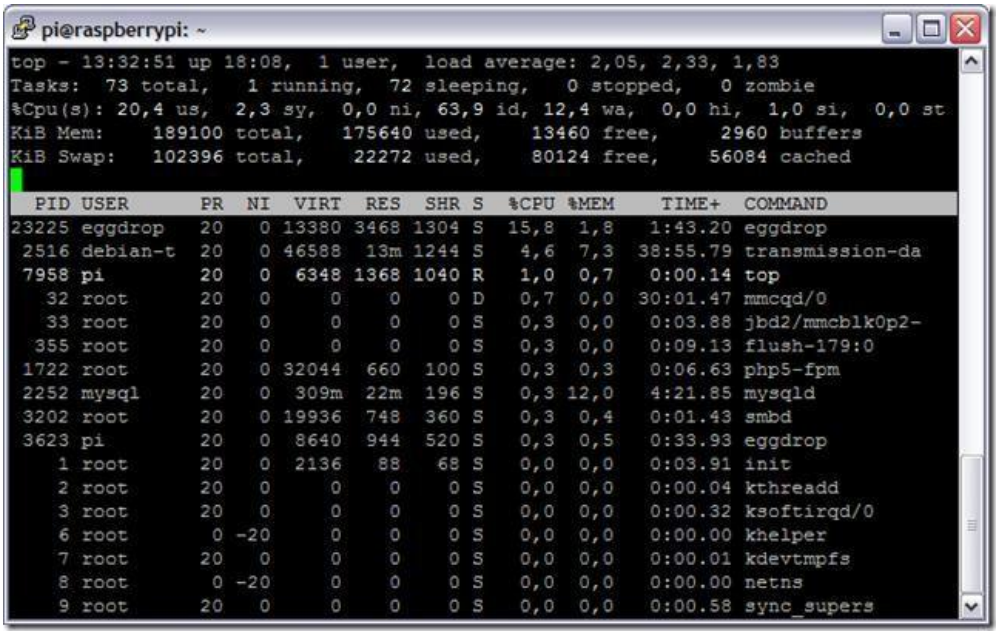


Рис. 2.4 Виконання команди top

Отже підключаємо та найперше потрібно перевірити чи вони є наявні в системі, для цього потрібно виконати команду “lsusb”, оскільки обидва пристрої підключені через USB. Якщо пристрої успішно підключені, потрібно внести конфігурацію мережі wlan0 у файл /etc/network/interfaces. для цього потрібно відкрити файл з допомогою команди “sudo nano /etc/network/interfaces”, та внести зміни у файл (рис 2.5).

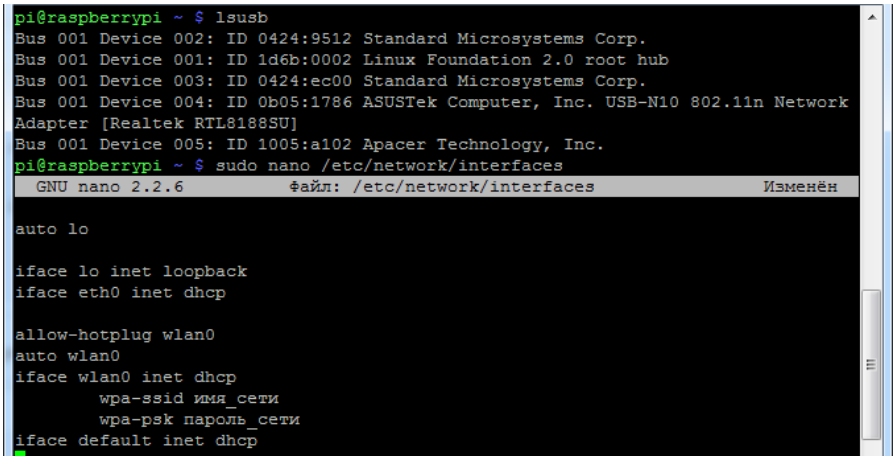


Рис 2.5 виконання команди lsusb та внесення змін у кофігурацію мережі.

Далі потрібно перезапустити мережеві служби, що можна зробити з допомогою команди “service networking restart”, та обновити систему виконавши по черзі “apt-get update” та “apt-get upgrade”.

Наступним підключемо зовнішній жорсткий диск. Для початку потрібно в каталозі “/media” створити папку “data”, куди будуть монтуватись розділи з файлами.

В “/etc/fstab” необхідно додати UUID розділу та шлях до дерективи, як показано на рис №. Далі зберігаємо файл конфігурації, монтуємо розділ “sudo mount /media/data”, для кожного наступного розділу потрібно повторити операцію заново. Перезагружаємо Raspberry та відключаємо клавіатуру з монітором, оскільки надалі будемо працювати з нею по ssh.

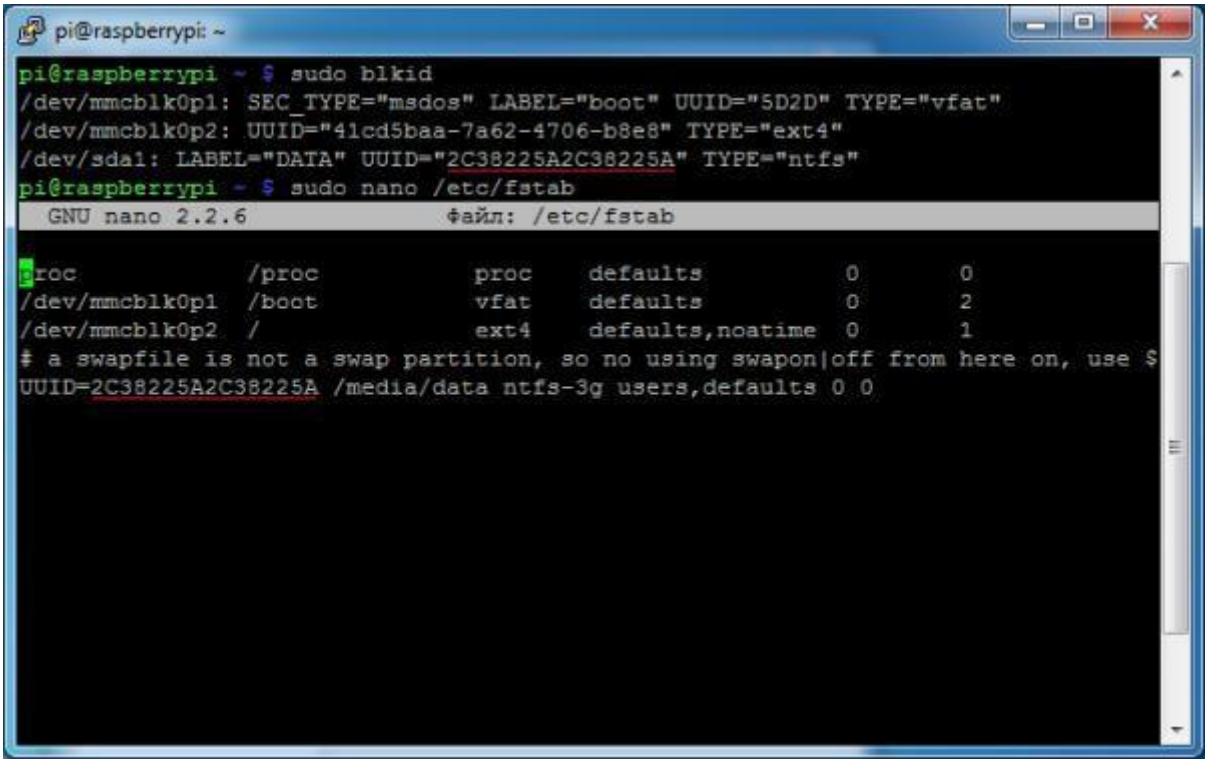


Рис. 2.6 Пошук зовнішнього диску та конфігурування файлу /etc/fstab

Далі в налаштуваннях роутера потрібно присвоїти Raspberry Pi статичну IP-адрессу і прописати параметри провайдера DDNS. На цьому

налаштування плати завершене, і її можна використовувати як сервер, та подальші дії проводити з допомогою ssh з’єднання.

2.4 **Підключення до Raspberry Pi з допомогою ssh. налаштування з’єднання та встановлення Domoticz**

2.4.1 Підключення до Raspberry з допомогою ssh з’єднання. Для створення ssh підключення з ОС Windows потрібна безплатна утилита PuTTY[14]. Для успішної роботи потрібно завантажити з сайту PuTTY такі файли:

- PuTTY.exe основний клієнт Secure Shell;
- PuTTYgen.exe генератор публічного та особистих ключів SSH;
- Pagent.exe для збереження ключів, агент аутентифікації;
- PSCP.exe захищене копіювання через консоль;
- PSFTP.exe захищене копіювання через протокол FTP.

Для коректної роботи у середовищі операційної системи Windows, потрібно встановити змінні середовища (рис 2.7):

Variable name - PATH

Variable value - C:\BIN; %PATH%

Вони потрібні для того, щоб утиліта знайшла всі системні файли, що потрібні їй для роботи.

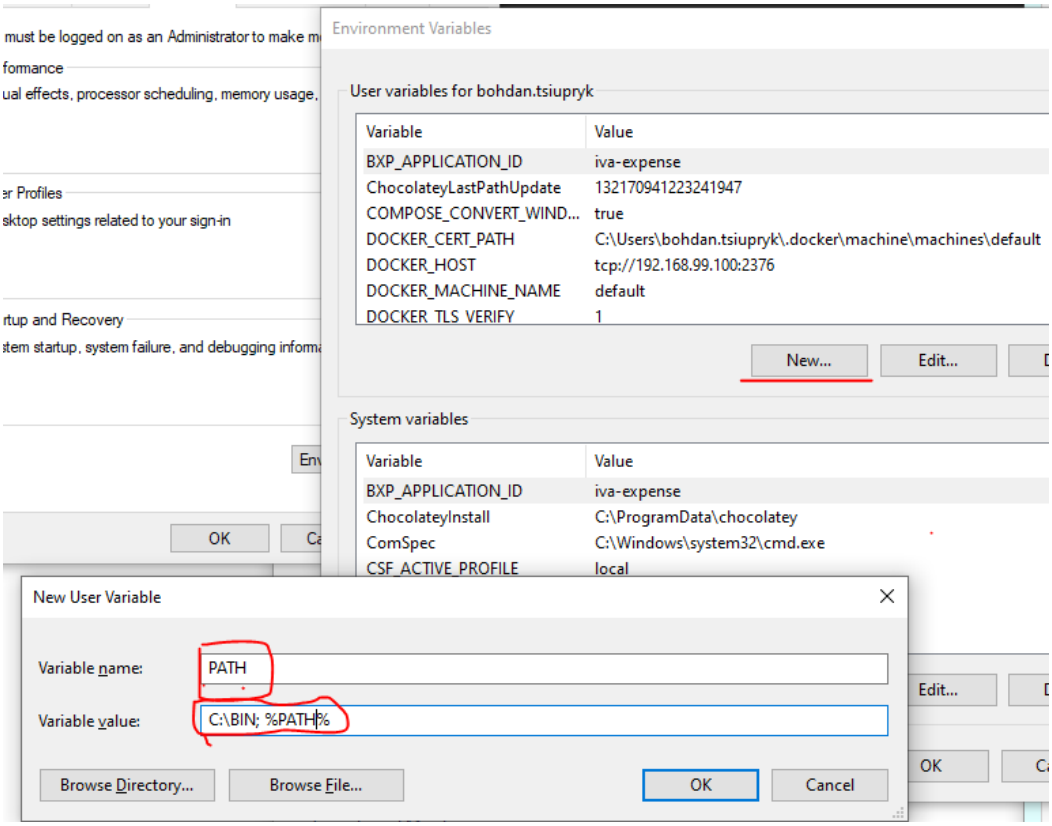


Рис. 2.7 Встановлення змінних середовища, для коректної роботи PuTTY

Далі розглянемо саму програму PuTTY, її налаштування, та процес підключення[14]. На рисунку 2.8 відображено налаштування сесії для підключення.

Розглянемо пункти налаштування підключення:

1. Host name (or IP address) - ім’я хоста до якого будемо підключатись, або його IP-адресса;
2. Port - для підключення по ssh використовується 22 порт;
3. Connection type - для використання з’єднання з допомогою security shell, потрібно обрати SSH;
4. Saved session - дозволяє зберігати сесії, щоб не потрібно було вводити дані кожного разу при запуску програми, для зберігання потрібно ввести назву сесії, та нажати кнопку Save.

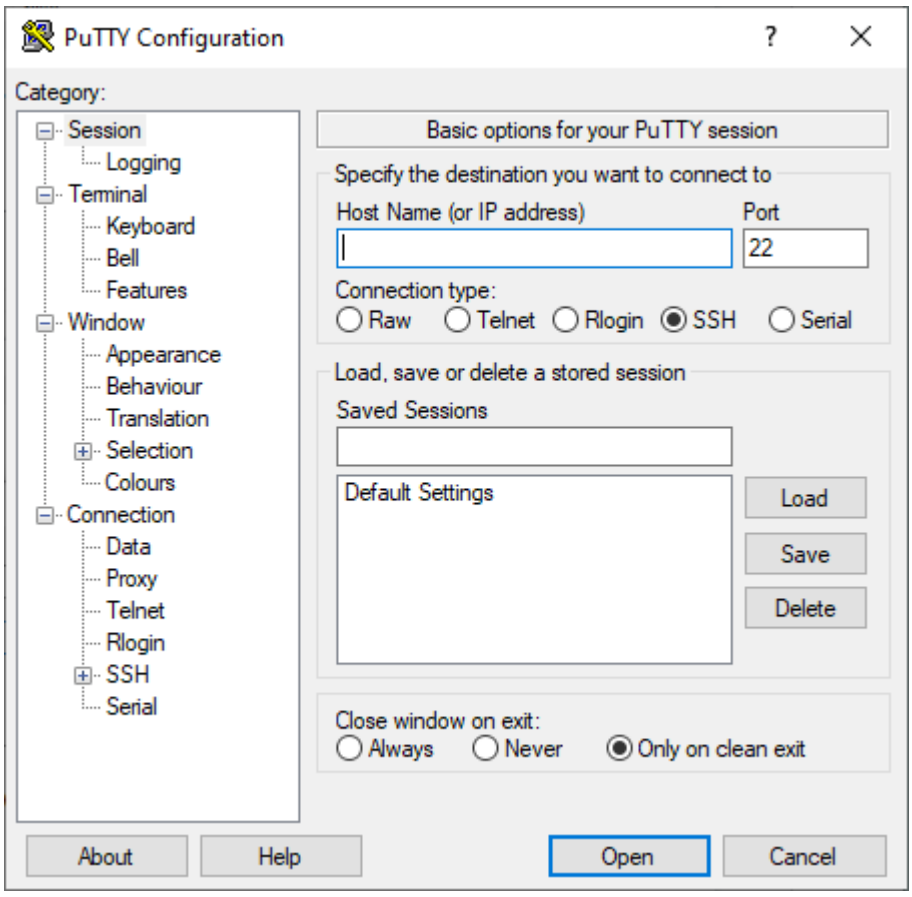


Рис. 2.8 Програма PuTTY. Вкладка session

Після введення всіх даних потрібно нажати кнопку Open, після чого відкриється термінал, де потрібно ввести логін та пароль, які ми задавали на початку конфігурації плати Raspberry Pi. Після введення даних, буде відкрите з’єднання з платою.

2.4.2 Встановлення та кофігурація Domoticz. Для встановлення потрібно лише перейти в домашню дирикторію з допомогою команди “cd /home/”, далі скачати та запустити з допомогою “curl -L https://install.domoticz.com | bash”. [15]Після закінчення встановлення, потрібно перейти у браузер та перейти по шляху “IP-адреса-Raspberry-Pi:8080”, відкриється початкова сторінка domoticz (рис. 2.9). Оскільки поки немає підключених пристроїв, вона буде пустою.

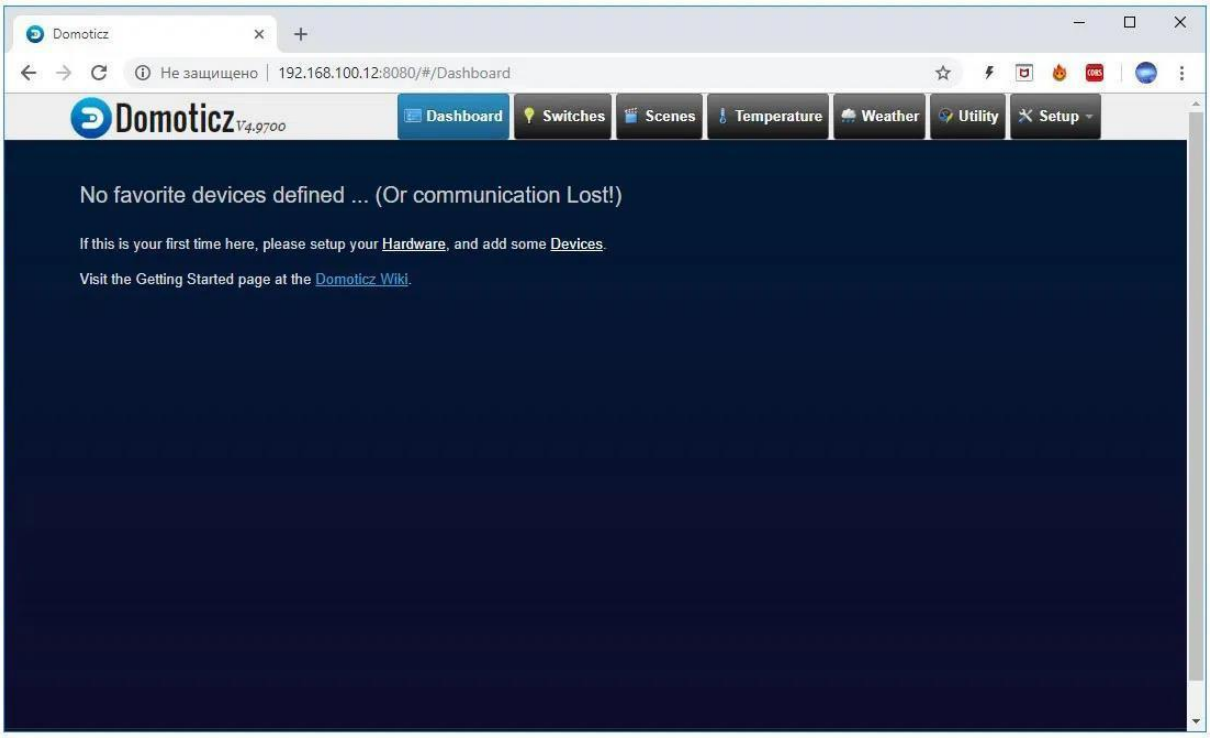


Рис. 2.9 Домашня сторінка програми Domoticz

Для налаштування потрібно перейти по шляху Setup - Setting (рис 2.10), та налаштувати такі параметри, як мова, тема оформлення, локація, захист сайту та інші. Для збереження налаштувань потрібно натиснути Apply Setting.

На цьому початкове налаштування завершене, і можна додати декілька датчиків для тестування.

Далі підключимо два датчика температури DS18B20 з допомогою інтерфейсу 1-ware, датчик температури та атмосферного тиску BPM180 та датчик освітленості TSL2561 з інтерфейсом I2C. Схема шини GPIO плати Raspberry Pi наведена у додатку Б.

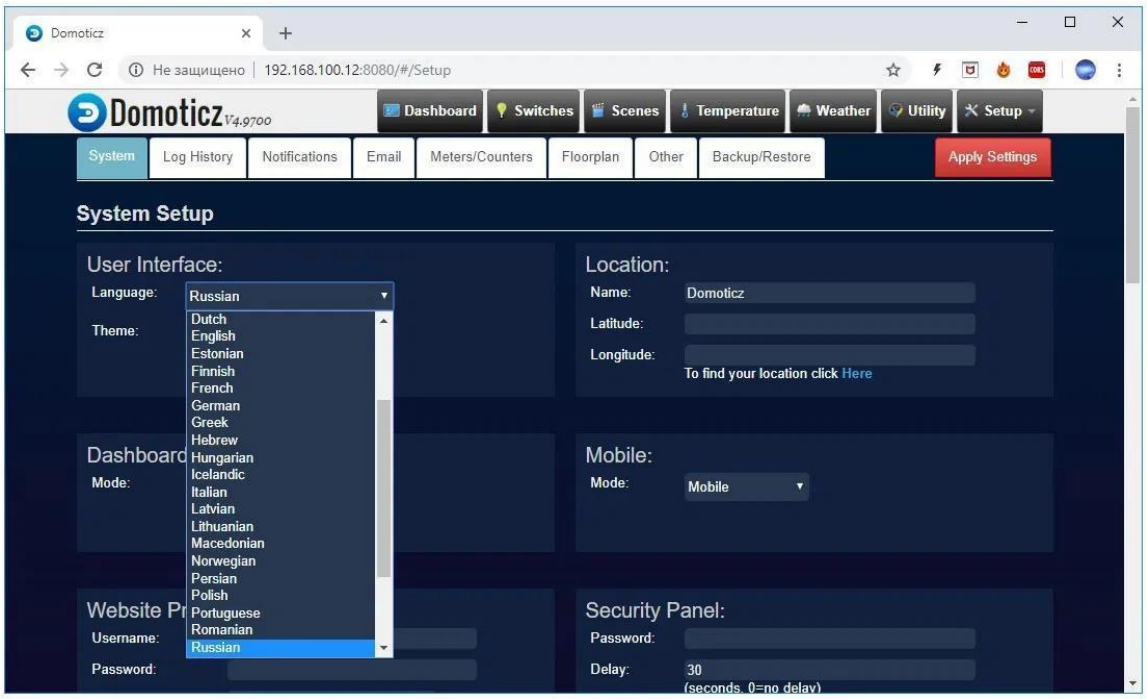


Рис. 2.10. панель налаштувань Domoticz

Підключення датчиків в програмі відбувається за допомогою веб-інтерфейсу. Для підключення потрібно перейти у вкладку Setup та обрати Hardware. у вікні що відкрилось можемо бачити список пристроїв, та вікно додавання нових (рис. 2.11). У вікні прописуємо ім’я пристрою (Name), наприклад “DS1820 #1”. Тип (Type) інтерфейсу встановлюємо “1-ware (System)”, шлях до патчу (OWFS Path) видаляємо.

інші налаштування залишаємо по-замовчуванням. Для збереження змін та додавання пристрою потрібно нажати кнопку Add. Аналогічно підключається і конфігурується другий датчик DS18B20[18].

Далі конфігурується датчик атмосферного тиску та температури BMP180. У пункті Type обираємо інтерфейс I2C sensors і у вкладці SubType - I2C sensor BMP085/180 Temp+Baro (рис. 2.12). За тим же принципом налаштовується датчик освітлення TSL2561. Тільки для даного датчика обираємо у пункті SubType - I2C sensor TSL2561 Illuminance[19].

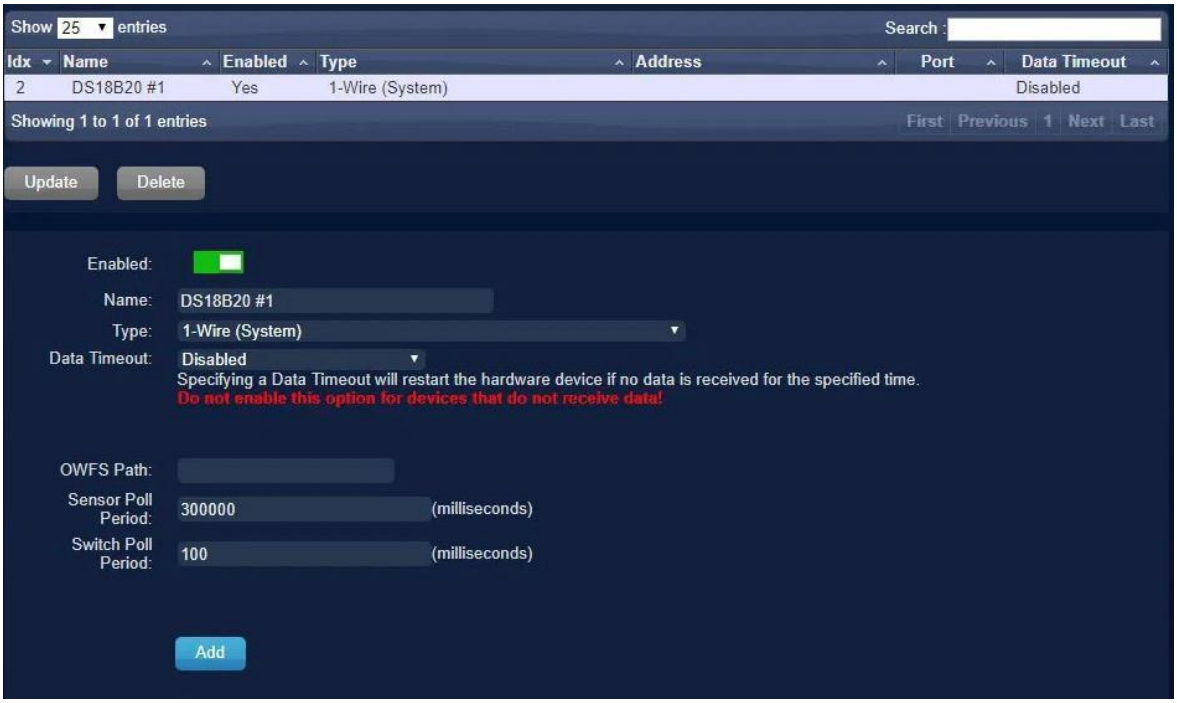


Рис. 2.11. Список підключених пристроїв та вікно додавання нового пристрою

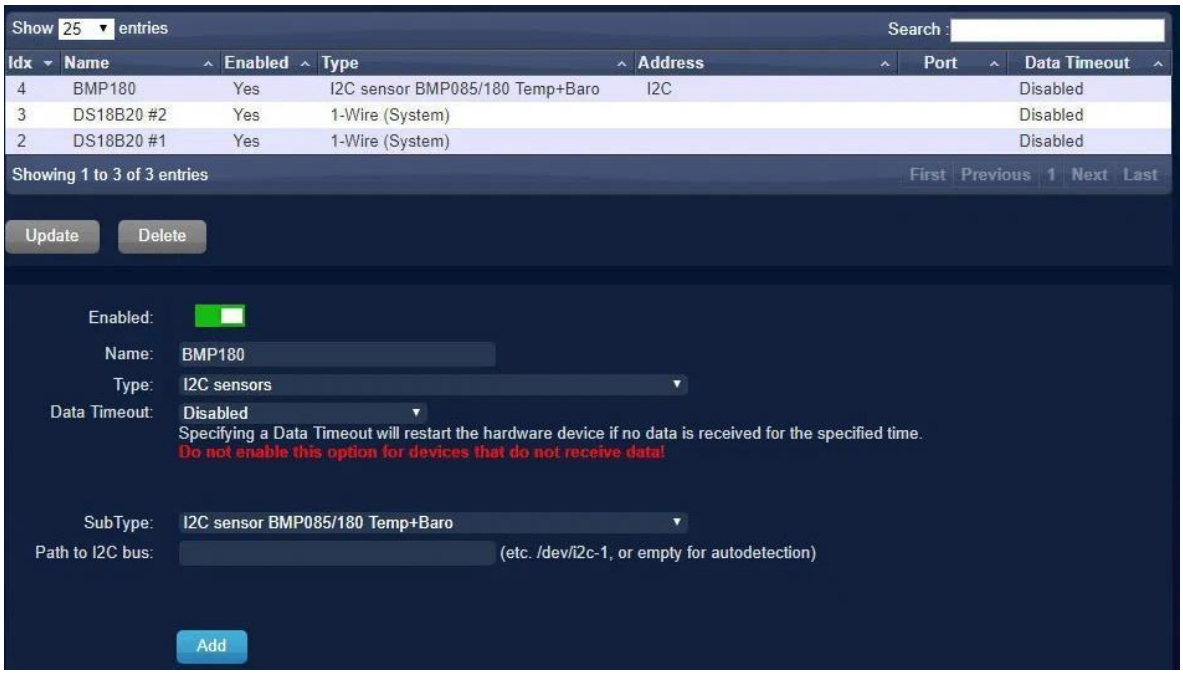


Рис. 2.12. Налаштування датчика атмосферного тиску та температури BMP180

Наразі датчики підключенні, але знаходяться у неактивному стані. Для їх активації потрібно перейти у вкладку Setup, обрати пункт Setting та активувати датчики. Для цього потрібно натиснути на зелене кільце напроти датчика, що потрібно активувати, та назвати його. Його назва буде відображатись в системі.

В системі датчики групуються по основній функціональності. Наприклад, датчики температури DS18B20 та BMP180 (температура) будуть відображатись у вкладці Tempetature. А датчик BMP180 (атмосферний тиск) у вкладці Weather. Датчик освітленості TSL2561, у свою чергу, буде у вкладці Utility.

Те що датчики гурупуються по функціональності, дуже хороша функція при великій кількості датчиків різної функціональності. Та у всіх будуть датчики які будуть використовуватись набагато частіше. Такі датчики можна додати на головну панель - Dashboard (рис 2.13.). Для того щоб додати пристрій на головну панель, потрібно просто натиснути на символ зірки на панелі пристрою. При цьому колір змінюється на жовтий, а пристрій з’являється на панелі Dashboard.

Рис. 2.13. Панель Dashboard з усіма обраними датчиками

Також в системі Domoticz, є система логування інформації. Система дозволяє відслідковувати динаміку зміни показань датчиків за різний час. Щоб побачити графік потрібного датчика потрібно просто вибрати його, та натиснути на його іконку. Як приклад на рис 2.14. показано графік добової температури датчика DS18B20.

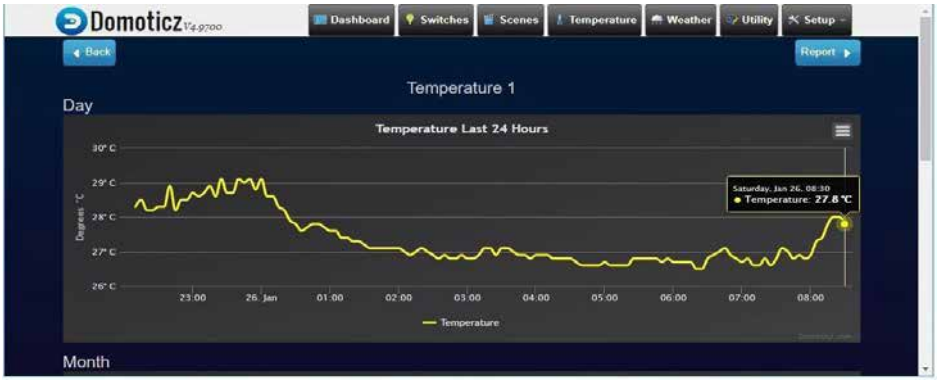


Рис. 2.14. Графік зміни температури у датчика DS18B20

2.5 **Аналіз Domoticz API, як способу управління**

Domoticz дозволяє всіляко взаємодіяти з усіма комутаторами та датчиками, що під'єднані до нього за допомогою JSON, або в інтерактивному режимі через браузер, або програмно з допомогою скриптів[20].

У даному розділі проаналізовано одноплатний комп'ютер Raspberry Pi, що найкраще підходить для розгортання на своїй базі засобу моніторингу та управління елементами інтернету речей.

Здійснено налаштування одноплатного комп'ютера Raspberry Pi, його першочергову конфігурацію. Здійснено установку системи Domoticz на одноплатний комп'ютер Raspberry Pi, та проаналізовано процес налаштування та досліджено можливості його користувацького програмованого інтерфейсу (API).

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗАСОБІВ МОНІТОРИНГУ ТА УПРАВЛІННЯ ЕЛЕМЕНТАМИ ГРАНИЧНИХ ПРИСТРОЇВ ІОТ

3.1 **Опис архітектури та інструментів розробки**

Для розробки було обрано архітектуру клієнт-сервер. Під клієнтом розуміється додаток, що встановлений на одній машині, або в локальній мережі з програмою domoticz. Сервер в свою чергу може знаходитись на віддаленій машині, або на хостингу.

Така архітектура дозволяє серверу не залежати від клієнтів і бути повністю автономним. А клієнтам у свою чергу надається простий спосіб підключення до сервера, та незалежність від інших клієнтів.

Додаток розроблений на мові програмування Java версії 8 та фреймворку Spring Boot.

За зберігання даних буде відповідальна СКБД (система керування базами даних) PostgreSQL. Також для розробки та роботи з даними використовуватиметься ORM фреймворк Hibernate. У нього є власна потужна мова запитів HQL, яка є спрощеною, об’єкто-орієнтованим аналогом мови SQL. Великою перевагою є те, що Hibernate дозволяє розробнику уникнути написання великої кількості SQL-запитів (оскільки вони вже реалізовані), йому залишається тільки викликати методи, які надає фреймворк.

Веб-інтерфес розроблений з допомогою шаблонізатору Freemarker.

Середовищем розробки було обрано Intellij IDEA від компанії JetBrains. У нього є безплатна версія з потужними плагінами та інструментами розробки, що полегшує розробку на мові Java.

3.1.1 Архітектура серверного додатку. Додаток розроблено за архітектурним шаблоном MVC (model, view, controller). Така аріхтектура дозволяє слабо зв’язувати між собою модулі, що дає можливість легкої підтримки та заміни модулів[23-25].

Детальніше про модулі:

1. Model це робота з базою даних, обробка та валідація даних;
2. View, частина яка приймає вхідні дані, та надсилає оброблені;
3. Controller, основна бізнес-логіка, обробка, та виконання операції з даними.

Model інкапсулює ядро даних і основний функціонал їхньої обробки і не залежить від процесу вводу чи виводу даних.

View може мати декілька взаємопов'язаних областей, наприклад різні таблиці і поля форм, в яких відображаються дані.

У функції Controller входить відстеження визначених подій, що виникають в результаті дій користувача. Контролер дозволяє структурувати код шляхом групування пов'язаних дій в окремий клас. Наприклад у типовому MVC-проекті може бути користувацький контролер, що містить групу методів, пов'язаних з управлінням обліковим записом користувача, таких як реєстрація, авторизація, редагування профілю та зміна пароля.

Зареєстровані події транслюються в різні запити, що спрямовуються компонентам моделі або об'єктам, відповідальним за відображення даних. Відокремлення моделі від вигляду даних дозволяє незалежно використовувати різні компоненти для відображення інформації. Таким чином, якщо користувач через контролер внесе зміни до моделі даних, то

інформація, подана одним або декількома візуальними компонентами, буде автоматично відкоригована відповідно до змін, що відбулися.

3.1.2 Опис архітектури клієнтського додатку. Клієнтський додаток розроблений за монолітною архітектурою. Причиною обрання даної архітектури є те, що у даного додатку відсутня потреба підключення бази даних, та веб інтерфейсу[25].

Основне завдання додатку це підтримка з'єднання з віддаленим сервером, та трансфер повідомлень. Тому використання мікросервісної або MVC архітектур є надмірним.

3.1.3 Опис мови програмування Java та фреймворку Spring Boot. Java є об'єктно-орієнтованою мовою програмування, що випущена у 1995 році. У мови С-подібний синтаксис. Та основною особливістю мови є Java virtual machine (JVM). Ця віртуальна машина дозволяє розробляти однаковий код для різних платформ, що значно збільшує швидкість розробки, та здешевлює її вартість. JVM використовує байт-код Java, який як правило, але не завжди генерується з вихідних кодів мови програмування Java.

Основні причини обрання Java, як мови для розробки даного додатку:

1. Простий, об'єктно-орієнтований синтаксис;
2. Безпечна та безвідмовна робота з пам'яттю;
3. Незалежність від платформи та архітектури;
4. Висока продуктивність.

Spring Boot Framework - фреймворк який спрощує розробку додатків на мові Java, оскільки надає реалізацію багатьох функцій, які у "простій" Java потрібно реалізовувати самостійно. Він дозволяє легко створювати повноцінні, продуктивні Spring-додатки, про які можна сказати - що їх потрібно "просто запустити"[22]. У Spring-платформу включено і сторонні бібліотеки, що дає змогу запустити додаток з мінімальними зусиллями. Більшості Spring Boot додатків потребують невеликої конфігурації.

Можливості Spring Boot:

1. Створення повноцінних додатків, які не потребують сторонніх програм;
2. Встроєні контейнери сервлетів такі як Tomcat або Jetty;
3. Автоматична конфігурація, де це можливо;
4. Надає можливості моніторингу станів, розширену конфігурацію та метрики;
5. Конфігурація без генерації коду або написання XML файлів (на відміну від звичайної версії фраємворку Spring).

Також для розробки веб-додатку, для Spring Boot потрібно підключити залежність для роботи з веб-інтерфейсами. Завдяки тому, що фраємворк використовує засіб автоматизації роботи з проектами Maven, підключення цілої бібліотеки реалізується у декілька стрічок коду, що потрібно додати у файл з назвою pom.xml (рис. 3.1).

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
        <exclusion>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-logging</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Рис. 3.1 Підключення залежностей для web розробки для Spring Boot

3.1.4 PostgreSQL та Hibernate. PostgreSQL - це реаляційна, об’єктна система керування базами даних, вона поширюється, як система з відкритим кодом, та підтримується великою кількстю людей, оскільки не прив’язана до будь якої з компаній. База підтримує велику кількість типів даних, окрім того система надає можливість користувачам створювати свої типи, та використовувати їх. Основні можливості PostgreSQL:

1. Дотримання принципів ACID;

2. Підтримка запитів і підзапитів з JOIN, UNION і тд;
3. Послідовності;
4. Реплікація;
5. Контроль цілісності;
6. Рекурсивні запити й загальні табличні вирази.

Hibernate у свою чергу є засобом перетворення класів у Java на таблиці у СКБД, та надає зручні інтерфейси керування даними та базою. Метою Hibernate є скорочення часу розробки завдяки вже реалізованим запитам. Він сам контролює зв'язок класів і відповідних таблиць, та надає простий спосіб їх з'єднання.

Також у Hibernate є своя мова запитів HQL, що є аналогом SQL, але сильно спрощена для роботи з об'єктами, а не з таблицями.

3.1.4 Шаблонізатор Freemarker. Простий шаблонізатор, що дозволяє без знання мов програмування для веб інтерфейсів, таких як JavaScript, або TypeScript, розробляти гнучкий та зручний інтерфейс користувача. Основною особливістю є макроси, які дозволяють зменшити кількість написання коду, завдяки перевикористання вже написаного раніше.

3.2 **Реалізація системи автоматизації збору даних**

3.2.1 Реалізація моделей та бази даних. У системі реалізовані такі моделі даних:

- Device, модель яка представляє собою пристрій, що підключений до системи;
- DeviceDTO, модель пристрою призначена для серіалізації та передачі, яка не містить зайвих даних;
- DeviceState, модель стану пристрою;
- User, модель користувача системи;
- UserRole, модель ролі користувача;

- Diagram, модель графіка даних;
- Client, модель клієнтської частини системи;
- Command, модель команди, що може передаватись, та зберігатись для історії пристрою.

Керування моделями відбувається за допомогою репозиторіїв “Repository”, інтерфейси фреймворку Hibernate, імплементація інтерфейсів продемонстрована на рис. 3.2.

Така імплементація дозволяє уникнути написання коду, що повторюється, адже у іншому випадку для кожної моделі потрібно реалізовувати методи доступу до БД[24].

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

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

Рис. 3.2. Приклад наслідування інтерфейсів, для уникнення великої кількості коду при доступі до БД

Інтерфейси JpaRepository та CrudRepository надають лише базові методи для доступу до БД, такі як, знайти всі об’єкти, знайти об’єкти по ідентифікатору, видалити об’єкти по ідентифікатору та оновити інформацію про певний об’єкт. Якщо потрібна ширша функціональність, у JPA (Java Persistence API) закладений механізм який дозволяє додавати методи для управління БД, з допомогою додавання методів з спеціальною назвою.

Наприклад, потрібно знайти пристрої які знаходяться у статусі OFF, базовий інтерфейс надає можливість пошуку лише по ідентифікатору. Для цього потрібно правильно сформулювати назву методу. Якщо це пошук, то

спочатку вказується, що повинно повернутись після запиту, оскільки кількість вимкнених пристроїв може бути більша ніж 1, то ми очікуємо масив або список пристроїв “List<Device>” далі потрібно вказати, що це пошук, з допомогою ключового слова “find”, потім вказується, що потрібні всі записи “All” та що ми шукаємо по параметру “ByStatus”, і в параметрах передаємо статус (рис. 3.3).

За цим ж принципом реалізуються інтерфейси для доступу для інших моделей даних.

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

import java.util.List;

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

Рис. 3.3. Приклад сформованого запиту на пошук пристрою по статусу

3.2.2 Реалізація бізнес-логіки додатку. Вся бізнес-логіка розміщена у пакеті service. Основна ідея додатку, це збір даних з декількох серверів, на яких встановлений Domoticz, їх систематизація в одному місці, збір статистики та можливий контроль.

У сситеми Domoticz є API, яке дозволяє через запити у форматі JSON, запитувати певні дані, або надсилати команди. Для відправки запитів буде використовуватись вбудований засіб Spring Boot під назвою RestTemplate. Він дозволяє без явних перетворень, надсилати дані. Клас має назву HttpUtil, та являє собою сервіс для відправки запитів. Клас наслідуєтсья від абстрактного класу ClosableHttpUtil (рис 3.4).

```java
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);
        }
```

Основні сервіси:

- HttpUtil, сервіс для відправки запитів;
- DeviceService, керування пристроями;
- ComandService, для створення, та обробки команд;
- UserService, керування користувачами, їх реєстрація, авторизація і тд;
- DiagramService, обрахування та перетворення діаграм у дані, які підходять для відображення;
- MailService, розсилка листів з статистикою та іншими даними про роботу програми;
- StatisticService, збір та обробка статистики про пристрої та роботу додатку;
- InitializationService, ініціалізує з’єднання з серверами, та проводить первинне налаштування системи

3.2.3 Реалізація інтерфейсу користувача та обробки інформації, що приходить від користувачів. Користувацький інтерфейс розроблений з допомогою шаблонізатора Freemarker. Великою перевагою якого є макроси. Макроси дозволяють використовувати повторно вже написаний код, змінюючи невеликі його частини. Наприклад, відображення діаграм. З їх допомогою, код відображення діаграм, написаний лише раз, а дані можна динамічно підставляти.

Для відображення діаграм було використано інструмент від компанії Google, Google chart. Це легковісний скрипт, який доволі легко використовувати. Все що потрібно, це підключити бібліотеку, вказати тип діаграми та дані для неї (рис 3.5).

Оформлення здійснено за допомогою бібліотеки Bootstrap 4. її перевагами є легкість в освоєні, та легковісність.

Обробку даних, що надсилаються з інтерфейсу користувача проводять контроллери. У них описується шлях, по якому вони будуть очікувати на дані, а після їх отримання обробляються та передають сервісу для подальших дій над даними.

```
<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]
    ]);
```

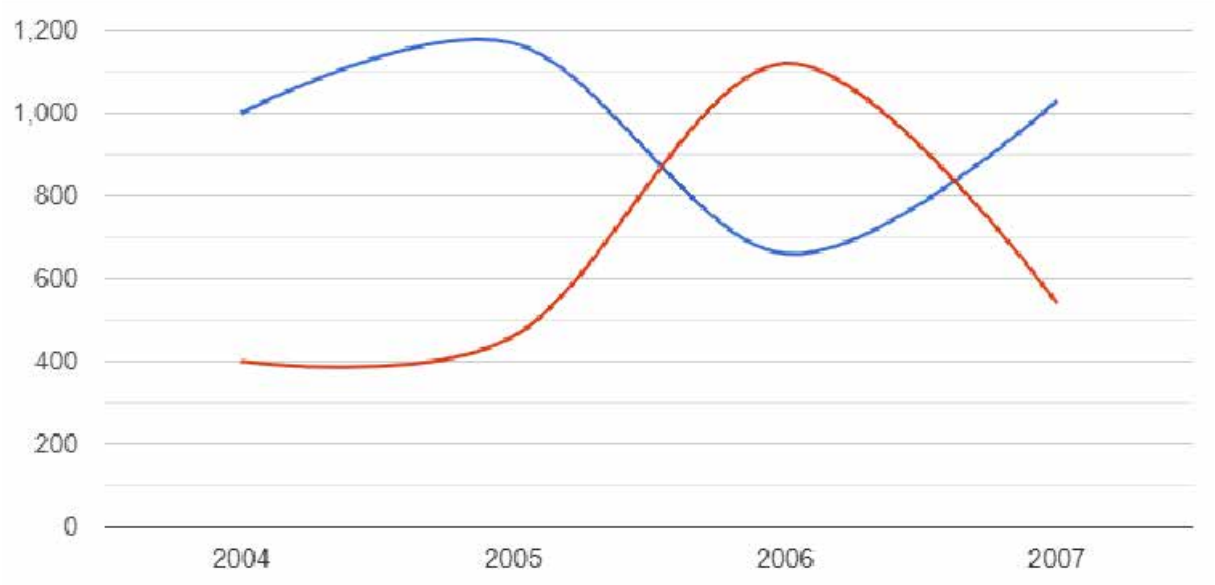


Рис. 3.5. Приклад коду діаграм, та їх вигляд після відображення

Оголошення контроллера відбувається за допомогою анотації @Controller або @RestController (рис 3.6). А шлях який буде обробляти контроллер задається з допомогою анотації @RequestMapping(path). Причому, методи контроллера, можуть мати свої шляхи, які будуть з'єднуватись з шляхом контроллера. Наприклад, якщо у контроллера вказаний шлях "/device", а в метода додана анотація @GetMapping("all"), то в результаті такий метод буде обробляти шлях "/device/all" з HTTP методом GET.

```
@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);
    }
```

Рис. 3.6. Приклад оголошення шляху, який обробляє HTTP метод GET

3.3 **Налаштування системи**

3.3.1 Конфігурація серверної частини системи. Конфігурація доступу до бази даних проводиться з допомогою конфігураційного файлу db.properties (рис 3.7). Після зміни конфігурації потрібно перезапустити додаток, оскільки конфігурація проводиться при запуску додатку.

Основні пункти налаштування:

1. datasource.url - шлях до бази даних;
2. datasource.username - ім'я користувача бази даних;

3. datasource.password - пароль для підключення до бази даних;

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

Рис. 3.7. Приклад конфігурації доступу до бази даних

3.3.2 Конфігурування клієнтської частини системи. Для налаштування потрібно у конфігураційний файл application.properties, встановити наступні налаштування:

1. remote-server.ip - IP-адреса віддаленого сервера куди будуть відправлятись дані;
2. remote-server.port - необов'язковий параметр, потрібно вказувати, лише якщо віддалений сервер використовує відмінний від 8080 порт;
3. remote-server.key - ключ безпеки, без якого запити будуть відхилятись;
4. remote-server.local-id - ідентифікатор клієнта;
5. domoticz.login - логін для підключення до domoticz;
6. domoticz.password - пароль для підключення до domoticz;
7. domoticz.port - необов'язковий параметр, потрібно вказувати, лише якщо додаток використовує відмінний від 8080 порт.

Після налаштування потрібно запустити клієнтську частину. Вона сама проведе підключення до domoticz, знайде всі підключенні пристрої, якщо попередні операції прошли успішно, підключиться до віддаленого сервера.

Якщо віддалений сервер недоступний, програма увійде у чекаючий режим, і буде періодично пробувати відновити підключення.

3.4 **Демонстрація роботи**

3.4.1 Встановлення та запуск. Для встановлення системи потрібно щоб на машині була встановлена JVM, не нижче 8-мої версії.

3.4.1.1 Клієнтський додаток. Потрібно скопіювати файл IoT-control-system-client.jar на машину з додатком Domoticz, та запустити. Запуск виконується за допомогою команди “java -jar IoT-control-system-client.jar”.

3.4.1.2 Серверний додаток. Для запуску додатка потрібна встановлена база даних з створеним користувачем. Після чого потрібно скопіювати файл IoT-control-system-server.jar, та запустити командою “java -jar IoT-control-system-server.jar”

3.4.2 Демонстрація роботи. користуватись системою можна, як з персональних комп’ютерів та ноутбуків, так і з мобільних пристроїв, основна вимога, підтримка браузера. Для початку користування потрібно перейти за адресою https://<ip-адресса-сервера>:8080. Після чого відкриється вікно входу в систему (рис 3.8).

Рис. 3.8. Форма входу в систему

Якщо це перший запуск, потрібно увійти під стандартними креденціалами “admin:admin”. Після чого змінити пароль, та додати інших користувачів за необхідності. Для цього в системі додана панель керування

користувачами (рис 3.9). На панелі можна редагувати інформацію та права користувачів, а також додавати нових.

Рис. 3.9. Панель адміністратора

Для підключення додатків-клієнтів до системи, потрібно спочатку його створити на панелі створення клієнтів (рис 3.10). Де потрібно ввести назву клієнта, що буде використовуватись в системі як ідентифікатор, та згенерувати ключ безпеки.

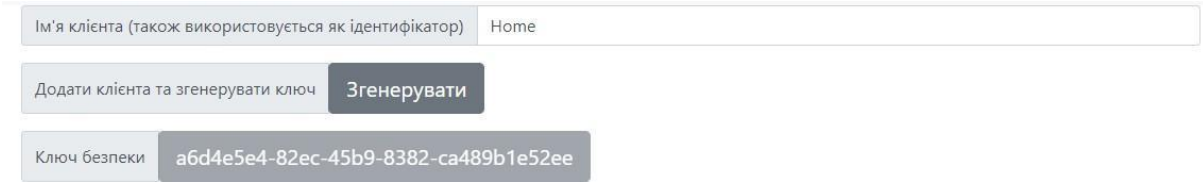


Рис 3.10. Приклад додавання нового клієнта

Після додавання клієнта, потрібно сконфігурувати його, та запустити. Під час конфігурації потрібно використовувати ідентифікатор та ключ отримані під час додавання.

Після успішного підключення клієнт з’явиться у списку з статусом “активний” (рис. 3.11.). Якщо підключення ще не було, клієнт буде у статусі “новий”. Якщо підключення вже відбулось, але з деяких обставин перервалось, або пристрій вимкнений, клієнт перейде у статус “не активний”.

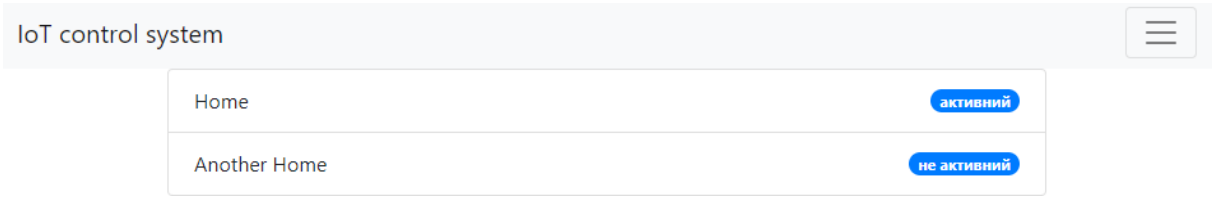


Рис. 3.11. Список клієнтів та їх статус

Для перегляду інформації про клієнта, потрібно натиснути на нього. Відкриється вікно з детальною інформацією про клієнта. Такою як, ім’я пристрою, його IP-адреса, час роботи, кількість підключених пристроїв, з можливістю переглянути інформацію про кожного з них окремо, та інша корисна інформація.

Для перегляду пристроїв, потрібно перейти на вкладку “Домашня”, тут відображені всі підключені на даний момент пристрої (рис 3.12). Їх можна сортувати за клієнтом, або типом пристрою (датчик температури, лампочка і тд).

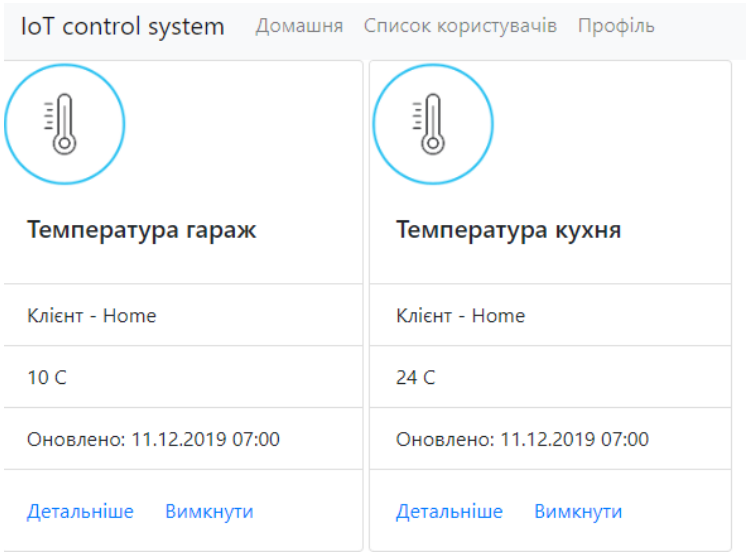


Рис. 3.12. Відображення на домашній сторінці датчиків температури

Також з домашньої сторінки можна перейти в детальніший огляд пристроїв. Та вимкнути пристрій, вимкнуті пристрої не відображаються на Домашній сторінці, і щоб їх увімкнути потрібно перейти в налаштування клієнта.

При переході за посиланням Детальніше на одному з пристроїв, відкривається вікно з детальною інформацією про пристрій. Його назва, показники, список команд які можна йому послати та якщо це наприклад датчик температури, то графік коливань температури за останню добу (рис. 3.12).

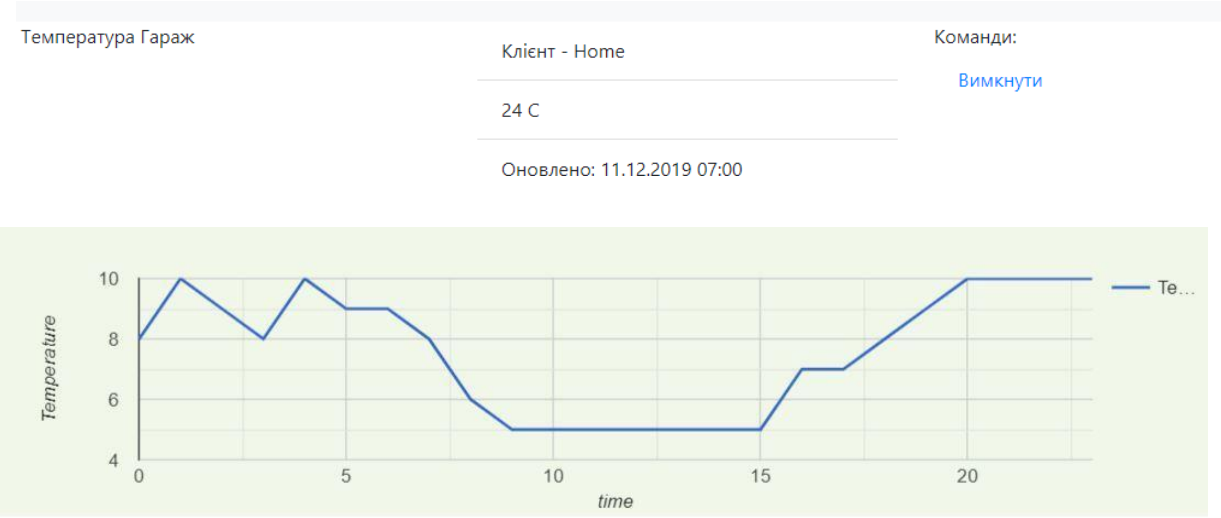


Рис. 3.12. Детальна інформація про пристрій та графік зміни температури

3.5 **Висновки до розділу 3**

У даному розділі описано інструменти, мову програмування Java та Spring Boot, з допомогою яких було розроблено систему яка працює поверх системи Domoticz та розширює її можливості. Використано незалежну архітектуру, що дозволяє швидко під’єднати ще декілька серверів-клієнтів до системи, без переписування коду системи.

Було продемонстровано принцип конфігурування клієнтів, можливість додавання нових клієнтів, їх конфігурацію та налаштування

ВИСНОВКИ

У бакалаврській роботі досліджено засоби моніторингу та управління елементами граничних пристроїв IoT, з метою полегшити роботу з мережею та елементами інтернету речей. Основні результати та висновки проведеної роботи:

- досліджено сучасні засоби передачі інформації у мережі інтернету речей;
- проаналізовано протоколи та методи, що використовуються для передавання інформації в мережі інтернету речей, що дало змогу виявити позитивні та негативні якості кожного з протоколів, та обрати оптимальні для створення мережі;
- досліджено систему Domoticz, та її програмний інтерфейс, що лягли в основу розроблюваної системи моніторингу та управління елементами інтернету речей;
- на основі системи Domoticz, з допомогою мови програмування Java та фреймворку Spring Boot, розроблено систему, що дозволяє з декількох віддалених один від одного місць віддалено спостерігати та керувати елементами інтернету речей;
- продемонстровано приклад роботи системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Z-Wave Technical Basics 2017 URL: https://www.domotiga.nl/attachments/download/1075/Z-Wave%20Technical%20Basics-small.pdf (дата звернення: 01.10.2019)
2. Технология NFC принципы работы и преимущества. 2016. URL: http://www.fotokomok.ru/texnologiya-nfc-principy-raboty-i-preimushhestva/ (дата звернення: 01.10.2019)
3. RFID-технології та магнітні мітки. 2015. URL: http://allta.com.ua/what-is-rfid (дата звернення: 01.10.2019)
4. The Basics of Bluetooth Low Energy (BLE). 2010. URL: https://www.novelbits.io/basics-bluetooth-low-energy/ (дата звернення: 01.10.2019)
5. LoRa Mesh Networking with Simple Arduino-Based Modules. 2018. URL: https://github.com/nootropicdesign/lora-mesh (дата звернення: 15.10.2019).
6. Bryan Boyd et al. Building Real-time Mobile Solutions with MQTT and IBM MessageSight. IBM Redbooks, 2014
7. IBM Podcast: Piper, Diaz, Nipper – MQTT. 2011. URL: http://www.ibm.com/podcasts/software/websphere/connectivity/piper_diaz_nipper_mq_tt_1118201 1.pdf.
8. Raspberry Pi Documentation. Official resource. URL: https://www.raspberrypi.org/documentation/(Last accessed: 14.10.2019).
9. THE OFFICIAL Raspberry Pi Beginner’s Guide. URL: https://www.raspberrypi.org/magpi-issues/Beginners_Guide_v1.pdf (Last accessed: 14.10.2019).
10. Questions regarding armhf port for Raspberry Pi. URL: https://lists.debian.org/debian-arm/2012/03/msg00002.html (Last accessed: 14.10.2019).

11. PuTTY Documentation Page. URL: https://www.chiark.greenend.org.uk/~sgtatham/putty/docs.html (Last accessed: 14.10.2019).

12. Domoticz. Частина Друга. webhomepi. URL: https://whp.home.blog/2019/02/02/domoticz-%d1%87%d0%b0%d1%81%d1%82%d1%8c-%d0%b2%d1%82%d0%be%d1%80%d0%b0%d1%8f/

13. Domoticz. Open Source Home Automation System. URL: https://www.domoticz.com/DomoticzManual.pdf

14. Domoticz. Official GitHub Page. URL: https://github.com/domoticz/domoticz

15. DS18B20. Programmable Resolution 1-Wire Digital Thermometer. URL: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

16. BMP180 Datasheet (HTML) - Texas Instruments. URL: https://www.alldatasheet.com/datasheet-pdf/pdf/514264/TI1/BMP180.html

17. Domoticz API/JSON URL's. Wiki page. URL: https://www.domoticz.com/wiki/Domoticz_API/JSON_URL%27s

18. The Java Language Environment. 1997 Sun Microsystems, Inc. URL: https://www.oracle.com/technetwork/java/intro-141325.html

19. Rob Harrop, Clarence Ho. Pro Spring 3. United Kingdom. 2012 944p

20. Joshua Bloch. Effective Java. Addison-Wesley Professional. 2018 412p

21. Андрій Будай. Дизайн-патерни — просто, як двері. 2012 90р

22. Alex Berson. Client/Server Architecture (McGraw-Hill Computer Communications Series). 1996 569p

23. Christian Bauer, Gary Gregory, Gavin King. Java Persistence with Hibernate Second Edition. 2015 608p

24. MQTT Essentials. URL: https://www.hivemq.com/mqtt-essentials/Robert C. Martin. Clean Code. 2008 464p

25. Brian Goetz, Tim Peierls, Joshua Bloch. Java Concurrency in Practice. 2006 432p

26. Eric Freeman, Elizabeth Robson. Head First Design Patterns. 2004 694 p.