МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ
ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ І.А. Жуков
(підпис)
«____»_____202__р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА СПЕЦІАЛЬНІСТЮ 123 «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ»

Тема: «Моніторинг та управління елементами *IoT*»

Виконавець: студент групи КС-231М Шевченко Олександр Олександрович
(студент, група, прізвище, ім'я, по-батькові)

Керівник: к.т.н., доцент кафедри, Дрововозов Володимир Іванович
(науковий ступінь, вчене звання, прізвище, ім'я, по-батькові)

Нормоконтролер: _____ Малярчук В.О.
(підпис) (ПІБ)

Засвідчую, що у дипломній роботі немає запозичень праць інших авторів без відпвідних посилань
Студент _____ Шевченко О.О.
(підпис) (ПІБ)

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних систем та мереж

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ Жуков І.А.
(підпис) (ПІБ)
«_____» _____2020 р.

ЗАВДАННЯ
на виконання дипломної роботи
Шевченка Олександра Олександровича
(прізвище, ім'я, по батькові)

1. Тема дипломної роботи
Моніторинг та управління елементами *IoT*

затверджена наказом ректора від "25" вересня 2020 р., № 1793/ст

2. Термін виконання проекту: з 05.10.2020 р. по 30.12.2020 р.

3. Вихідні дані до роботи:
   3.1 Засоби моніторингу та управління елементами *IoT*
   3.2 Протоколи передачі даних
   3.3 Засоби передачі даних

4. Зміст пояснювальної записки:
   4.1 Вступ
   4.2 Аналіз мережі інтернету речей
   4.3 Апаратне забезпечення
   4.4 Налаштування засобів моніторингу
   4.5 Програмне забезпечення

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:
   5.1 Презентаційні матеріали

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Проаналізувати джерела, по темі дипломної роботи	05.10 – 16.10.20	
2	Проаналізувати матеріали по мережі інтернету речей	17.10 – 25.10.20	
3	Проаналізувати задачі моніторингу у мережі *IoT*	26.10 – 30.10.20	
4	Визначити особливості архітектури мережі *IoT*	31.10 – 06.11.20	
5	Проаналізувати проблеми кібербезпеки у мережі *IoT*	07.11 – 12.11.20	
6	Визначити необхідне для роботи системи апаратне забезпечення	13.11 – 17.11.20	
7	Визначити та провести налаштування засобів моніторингу та управління	18.11 – 23.11.20	
8	Визначити необхідне для роботи системи програмне забезпечення	24.11 – 30.11.20	
9	Розробити структуру та зміст пояснювальної записки	01.12 – 05.12.20	
10	Розробити презентаційні матеріали до змісту дипломної роботи	06.12 – 20.12.20	
11	Захист дипломної роботи	22.12.20	

8. Дата видачі завдання: “05" 10 2020 р.

Керівник дипломної роботи _____ Дрововозов В.І.
(підпис керівника) (ПІБ.)

Завдання прийняв до виконання _____ Шевченко О.О.
(підпис випускника) (ПІБ.)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Моніторинг та управління елементами *IoT*» 91 сторінка, 33 рисунків, 6 таблиць, 34 джерел літератури.

ІНТЕРНЕТ РЕЧЕЙ, *INTERNET OF THINGS*, ДАТЧИКИ, МІНІ-ПК *RASPBERRY PI*, МОНІТОРИНГ, УПРАВЛІННЯ, *DOMOTICZ*, МЕРЕЖА *IOT*.

Мета і завдання дипломної роботи – дослідити систему керування засобами управління та моніторингу елементами *IoT*.

Об'єкт дослідження – віддалене управління та моніторинг елементами інтернету речей.

Предмет дослідження – апаратне забезпечення у синтезі з програмним забезпеченням, за допомогою якого відбувається управління та моніторинг елементами *IoT*.

Методи дослідження, технічні та програмні засоби – системний аналіз, синтез, аналіз, програмний додаток *Domoticz,* безпровідні та провідні технології для передачі інформації в мережі інтернету речей.

Основні конструктивні, технологічні показники об'єкту – система управління та моніторингу з використанням додатку *Domoticz*, мови програмування *Java*, використання міні-ПК *Raspberry Pi*.

Практична значимість дослідження – дослідження протоколів та засобів, концепції віддаленого керування та моніторингу елементами *IoT*, для виявлення необхідного та мінімально необхідного забезпечення, яке використовується у керуванні та моніторингу елементами інтернету речей.

Рекомендації щодо використання результатів дипломної роботи – об'єкт може бути використаний у промислових та побутових потребах, для керування та моніторингу власного виробництва малих та регіональних значень.

Прогнозні припущення про розвиток об'єкту дослідження – постійне удосконалення та розвиток системи захисту даних у мережі інтернету речей.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

API	–	*Application Programming Interface*
GPIO	–	*General Purpose Input / Outpu*
IoT	–	*Internet of Things*
RFID	–	*Radio Frequency IDentification*
SSH	–	*Secure Shell*

ВСТУП

Широкому поширенню Інтернету речей сприяє масова поява пристроїв, оснащених електронними компонентами, програмним забезпеченням і комунікаційними можливостями, будь то смартфон, камера на нафто буровій установці або оптичний датчик на сталепрокатному стані, які збирають і передають дані.

Можна припустити, що через кілька років практично на всіх виробничих підприємствах будуть широко застосовуватися технології та системи на базі Інтернету речей. Тема Інтернету речей є актуальною, тому що з'являється досить велика кількість пристроїв, здатних обмінюватися даними без участі людини, як у побутові плані, так і у промисловому.

Під впливом сучасних інформаційних технологій відбувається докорінна зміна всіх процесів в системі публічного управління, вони автоматизують багато управлінських процесів, сприяють ефективному виробленню управлінських рішень, роблять владу більш прозорою. Україна була першою країною, в якій ще у 80-х роках XX століття намагалися вирішити питання автоматизації управлінських рішень. Видатним українським вченим Глушковим В.М. була запропонована програма ЗДАС, тобто «Загальнодержавна автоматизована система збору та обробки інформації» для обліку, планування та управління народним господарством. В Україні було прийнято низку нормативно-правових актів, які впливають на використання та розвиток технологій Інтернет-речей.

Причиною обрання даної теми є стрімкий розвиток технологій у світі, багатомільярдні компанії об'єднуються в альянси задля проектування єдиних стандартів та протоколів роботи пристроїв інтернету речей, підприємства переходять від ручної праці до автоматизованої, у побуті з'являються пристрої здатні покращувати та полегшувати життя людей, навіть без їх втручання. Уся індустрія йде шляхом полегшення життя людей, розробки автоматизованих систем, де люди прийматимуть мінімальну участь. Це в свою чергу дозволить мінімізувати ризики та зменшити витрати. Але звичайно мають існувати способи для контролю цих

пристроїв. Будь-яка електроніка може містити у собі програмні або апаратні помилки, котрі треба вирішувати у короткий термін. Також для зручності мають існувати інтерфейси керування та моніторингу пристроїв. Наприклад, у побуті для керування пристроями інтернету речей можна використовувати додаток на смартфон, але щоб це працювало, пристрої мають бути об’єднанні в одну мережу тощо.

Інтернет речей є наступним кроком на шляху до оцифрування сучасного суспільства, де предмети і люди пов'язані один з одним через комунікаційні мережі і з’являється можливість повідомляти про їх стан та стан навколишнього середовища. Розробники таких програм намагаються якомога більше удосконалювати свої роботи. Інтернет речей створює нові можливості та забезпечує конкурентні переваги для бізнесу як на поточних, так і на нових ринках.

У найближче десятиліття завдяки інтернету речей (*IoT*) очікується сплеск обсягу даних, вироблених підприємствами. Це надасть великий вплив на ефективність і конкурентоспроможність бізнесу. Додатки *IoT* відрізняються від звичайних одноадресних комунікацій, тому підприємствам необхідно відповісти на ряд важливих питань, щоб зберегти бізнес в епоху інформаційної революції.

Інтернет речей називають третьою хвилею інформаційної революції. Другий було поширення смартфонів і мобільних комунікацій. Четвертою, ймовірно, буде штучний інтелект (ШІ), він візьме на себе подальший розвиток інформаційного середовища, в яку будуть інтегруватися люді. *IoT* - це не тільки мільярди речей, підключених до мережі, але і новий спосіб управління бізнес-процесами. Уже зараз технології інтернету речей набули широкого поширення. Наприклад, виробники авіадвигунів, будівельної техніки можуть стежити за роботою виробів в режимі реального часу, а керуючі компанії отримувати інформацію за показниками лічильників жителів району. Інтернет речей генерує величезну кількість інформації і швидко росте. За прогнозами *IDC*, ринок *IoT* До 2020 р досягне $ 7,1 трлн.

Десятки мільярдів пристроїв і датчиків в різних куточках планети будуть генерувати інформаційні пакети спорадично. Це реалії Інтернету речей, які мають на увазі відмову від застарілого принципу відстеження мережевого трафіку з інтервалами в п'ять хвилин. Так, при аналізі мережевого трафіку з інтервалами 1,5

секунди кількість сплесків трафіку може бути на 250% вище, ніж при інтервалі в одну хвилину. Потрібно використовувати платформу моніторингу, яка в змозі виробляти збір даних з будь-якої необхідної частотою. Для світу *IoT* це є обов'язковим.

Інтернет речей змінює спосіб розвитку промислового та споживчого ринку. Робото-технічні пристрої, дрони, автономні транспортні засоби, блокчейн, розширена та віртуальна реальність, цифрові помічники та машинне навчання (штучний інтелект) – це технології, які переводять на наступний етап розробки додатків IP.

Разом з перевагами від застосування технології Інтернет речей, зростає загроза їх неправомірного використання, збір персональних даних, втручання в приватне життя громадян, кібернетична загроза тощо.

Моніторинг та контроль приладів є одним із важливих заходів, за яким слід пильно стежити та використовувати його в режимі реального часу для безпеки, безпеки та комфорту людей. З розвитком Інтернет-технологій та бездротових сенсорних мереж реалізується нова тенденція в епоху повсюдного поширення. Величезний приріст користувачів Інтернету та модифікації робочих технологій Інтернету дозволяють створювати мережі повсякденних об'єктів. Споживачі покладають великі надії на системи з підтримкою Інтернету. Їх переваги добре відомі. Зниження експлуатаційних витрат та витрат на обслуговування завдяки віддаленому моніторингу, діагностиці, налагодженню та модернізації мікропрограми. Зручність та безпека, що поставляються з можливістю контролювати стан розумного будинку та контролювати Інтернет-прилади, коли споживачів немає вдома. Дистанційний моніторинг житлових та промислових об’єктів, повідомлення аварійних служб на випадок пожежі, крадіжки та витоку рідини чи газу. Подібні типи Інтернет-систем призначені для збору великої кількості даних перед їх обслуговуванням за запитом. У цих додатках дані компілюються на центральному сервері, а потім подаються клієнтам через Інтернет. У роботі пропонується система моніторингу та управління приладами на основі міні-ПК *Raspberry Pi*. Пропонується компактна мережу з можливістю Інтернету. Система може контролювати стан датчиків через Інтернет, коли оновлення інформації на веб-сервері зчитується за певним алгоритмом, що подається в *Raspberry pi*, а потім система відповідає на відповідні інструкції з високим

захистом. Користувач може безпосередньо входити в систему та взаємодіяти із вбудованим пристроєм у режимі реального часу. Система гнучка для розміщення широкого спектру вимірювальних приладів з відповідними інтерфейсами. Вона має різноманітні переваги, такі як енергоефективність, інтелект, низька вартість, портативність та висока продуктивність.

Актуальність теми є такою, що наразі на ринку існує безліч об’єктів, які можна автоматизувати та безліч фірм, які пропонують різні варіанти для задоволення потреб клієнта. Також у світі існує та ще довго існуватиме тренд на автоматизацію та віддалене керування і моніторинг власних об’єктів виробництва або навіть власного житла.

Не варто забувати, що такі потреби можуть виникати не тільки у підприємствах з багатотисячними бюджетами, а й у звичайних споживачів і їм необхідно пропонувати негірший продукт, але значно дешевший. Саме тому і було обрано дану тему, щоб дослідити, що являє собою мережа інтернету речей, які технології та протоколи там застосовуються, яке мінімальне апаратне та програмне забезпечення необхідне для функціонування системи керування та моніторингу елементами інтернету речей.

У дипломній роботі проводиться аналіз існуючих пристроїв мережі *IoT*, аналіз типів мереж, технологій та протоколів для визначення методів моніторингу та управління елементами *IoT*. Все це у своєму синтезі дає змогу визначити необхідне апаратне та програмне забезпечення для безперебійної роботи мережі пристроїв *IoT*.

Мета і завдання дипломної роботи. Метою даної дипломної роботи є дослідження існуючих засобів для розгортання *IoT* та визначення методів управління та моніторингу елементами *IoT*.

Для досягнення поставленої мети виконуються такі задачі:

– аналіз архітектури інтернету речей;

– аналіз існуючих типів мереж, протоколів для з’єднання пристроїв в одну мережу;

– аналіз апаратного забезпечення *IoT*;

– аналіз існуючого програмного забезпечення моніторингу та управління *IoT*;

– синтез проаналізованих рішень для вирішення поставленої мети.

Об’єкт дослідження – віддалене управління та моніторинг елементами інтернету речей.

Предмет дослідження – апаратне забезпечення у синтезі з програмним забезпеченням, за допомогою якого відбувається управління та моніторинг елементами *IoT*.

Методи дослідження. Для виконання задачі дипломної роботи використано наступні методи: системний аналіз, синтез, аналіз.

Наукова новизна отриманих результатів. У даній дипломній роботі проведено компаративний аналіз протоколів передачі даних, а також засобів моніторингу та управління елементами *IoT*, що дозволило виявити та обґрунтувати найбільш ефективні та оптимальні для їх подальшого використання у мережах інтернету речей.

Практичне значення отриманих результатів. Проведені дослідження дозволяють зрозуміти, які програмні та апаратні засоби необхідні для побудови системи управління та моніторингу елементами інтернету речей. А також, як це зробити з мінімальними матеріальними внесками.

Це дослідження є досить ефективним для використання, тому що в ньому наведені конкретні пристрої, програмні засоби та кроки, для синтезу програмних і апаратних забезпечень, що в свою чергу дає змогу використати ці результати у власних цілях та побудувати власну мережу інтернету речей.

В прогнозних припущення про подальший розвиток можна сказати, що предмет та об’єкт дослідження можна і надалі удосконалювати та досліджувати. Адже новітні технології розвиваються з кожним днем, вони не стоять на місці.

Компанії з кожним роком роблять великий внесок у розвиток новітніх технологій, а тому попередні продукти швидко старіють та дешевшають. Але вони ще деякий час можуть не втрачати свою актуальність. У світ приходять новітні стандарти зв’язку з іншими швидкостями передачі даних, тому однією з форм

удосконалення є оновлення апаратного забезпечення, яке відповідатиме сучасним вимогам.

Через деякий час, буде актуально провести нове дослідження для виявлення оптимальних технологій та засобів для побудови системи управління та моніторингу елементами інтернету речей. Але це дослідження вже буде відповідати новим стандартам та більш новим технологіям.

Також іншою формою розвитку дослідження є сторона захисту інформації. Захист інформації та конфіденційність є однією з пріоритетних складових під час вибору споживачем системи. Без належної впевненості в безпеці і приватності даних користувача ця система багатьма не буде прийнята.

РОЗДІЛ 1
АНАЛІЗ МЕРЕЖІ ІНТЕРНЕТУ РЕЧЕЙ

Інтернет Речей (*IoT*) - це нова концепція, в якій Інтернет еволюціонує від об'єднання комп'ютерів і людей до об'єднання (розумних) об'єктів або речей. З безперервним просуванням технологій Інтернет Речей потенційні інновації «обрушуються» на нас, розростаючись до глобальної обчислювальної мережі, де все і все будуть з'єднані за допомогою Інтернет (рис. 1.1). *IoT* постійно розвивається і є гарячою темою для досліджень в даний час. Звична форма Інтернету переходить в його модифіковану й інтегровану версію. Кількість пристроїв, що використовують Інтернет-послуги, зростає з кожним днем і з'єднання їх усіх за допомогою проводів або бездротових технологій дасть нам потужне джерело інформації на кінчиках наших пальців. Концепцію розширюють можливості взаємодій між розумними машинами та ультрасучасною технологією. Але технології, які становлять Інтернет Речей, не є чимось новим.

Моніторинг та контроль приладів є одним із важливих заходів, за яким слід пильно стежити та використовувати його в режимі реального часу для безпеки, безпеки та комфорту людей. З розвитком Інтернет-технологій та бездротових сенсорних мереж реалізується нова тенденція в епоху повсюдного поширення. Величезний приріст користувачів Інтернету та модифікації робочих технологій Інтернету дозволяють створювати мережі повсякденних об'єктів. Споживачі покладають великі надії на системи з підтримкою Інтернету. Їх переваги добре відомі. Зниження експлуатаційних витрат та витрат на обслуговування завдяки віддаленому моніторингу, діагностиці, налагодженню та модернізації мікропрограми.

Зручність та безпека, що поставляються з можливістю контролювати стан розумного будинку та контролювати Інтернет-прилади, коли споживачів немає вдома. Дистанційний моніторинг житлових та промислових об’єктів, повідомлення аварійних служб на випадок пожежі, крадіжки та витоку рідини чи газу [1].

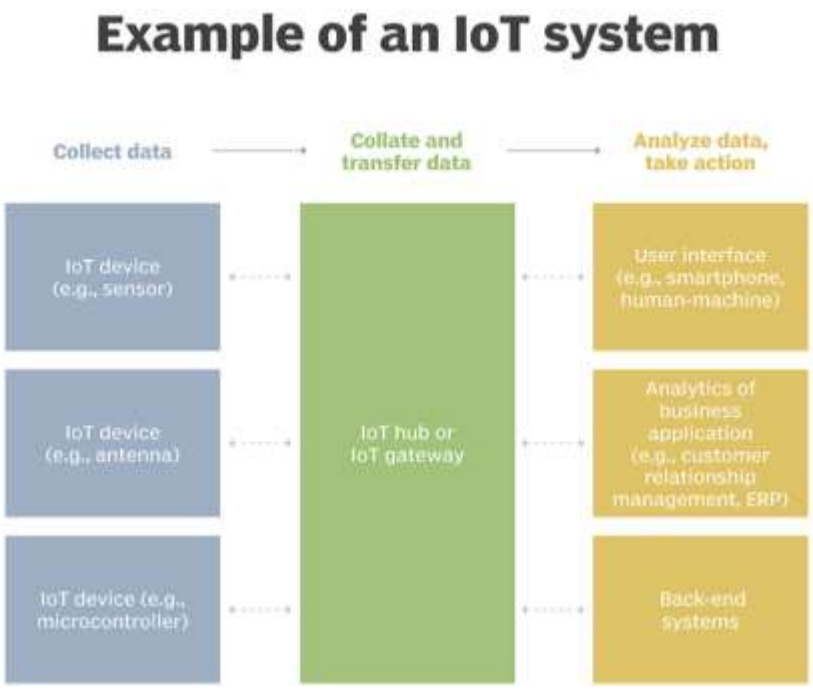

Рис. 1.1. Приклад мережі *IoT*

IoT є підходом з'єднання інформації, отриманої від різних джерел, на будь-якій віртуальній платформі або наявній інтернет-інфраструктурі. Концепція Інтернету Речей з'явилася в 1982 році, коли модифікований автомат з газованою водою був підключений до Інтернету і був здатний повідомляти про наявність в ньому напоїв і їх температурі. Пізніше, в 1991 році Марком Вайзером була вперше дана сучасна оцінка Інтернету Речей. Так чи інакше, в 1999 році Білл Джой підказав про зв'язок між пристроями у своїй таксономії Інтернету. У тому ж році Кевін Ештон запропонував термін «Інтернет Речей» для пов'язаних пристроїв. Базовою ідеєю *IoT* є надання можливості автономного обміну корисною інформацією між унікально ідентифікованими пристроями реального світу. Ці пристрої оснащені новітніми технологіями, такими, як радіочастотна ідентифікація (*RFID*) і бездротові мережі датчиків (*WSNs*), і надалі отримують можливість приймати самостійні рішення в залежності від того, яке автоматизоване дію виконується.

У 2005 році Міжнародний союз по електрозв'язку (*International Telecommunications Union, ITU*) оголосив еру всепроникаючих мереж, головною ознакою яких є зв'язок мереж між собою. Головна концепція Інтернету Речей - це середовище, в якому речі мають здатність слухатися управління, а дані про речі

можуть бути оброблені для виконання бажаної задачі за допомогою навчання пристроїв. Практична реалізація *IoT* добре продемонстрована в *Twine*, компактному і малопотужному апаратному забезпеченні, які працюють разом з мережевим програмним забезпеченням в реальному часі й дозволяє зробити цю концепцію реальністю. Проте, у різних людей і організацій є свої відмінні концепції Інтернету Речей [2-5].

1.1. Аналіз технологій, які використовуються в *IoT*

«Речами» в інтернеті речей є, головним чином, глибоко вбудовані пристрої з такими особливостями, як вузька смуга пропускання, збір даних з низкою повторюваністю і малий обсяг використовуваних даних. Ці пристрої обмінюються даними один з одним і надають дані через інтерфейси. Деякі вбудовані пристрої *IoT*, такі як охоронні відеокамери високого дозволу, відеотелефони *VoIP* і деякі інші, вимагають для роботи широкосмугового стрімінгу. Але незліченна кількість інших продуктів вимагає передачі пакетів даних всього лише час від часу [6].

1.1.1. Типи мереж *IoT*

Отже, для аналізу мережі слід почати з того, які типи мереж використовуються у побудові мережі *IoT*.

Мережі з низьким енергоспоживанням і малим діапазоном. Ці мережі добре підходять для будинку, офісу та інших невеликих середовищ. Для їх застосування досить невеликих акумуляторів, а іноді їх можна налаштувати й без використання акумулятора. Як правило, вони досить економічні в експлуатації. Нижче наведені найпоширеніші приклади:

– *Bluetooth* – забезпечує високошвидкісну передачу даних і відправляє сигнали голосу і даних на відстань до 10 метрів.

– *Wi-Fi / 802.11* – завдяки низькій вартості експлуатації *Wi-Fi* - стандартний варіант для дому та роботи. Але цей варіант підходить не для всіх сценаріїв через обмежений діапазон дії і постійне споживання енергії.

– *Z-Wave* – мережа в мережі для домашніх пристроїв. Забезпечує обмін даними за допомогою низькоенергетичних радіохвиль. *Z-Wave* забезпечує взаємодію між домашніми системами автоматизації на рівні додатків.

– *Zigbee* – це популярний варіант для домашніх систем автоматизації та медичних пристроїв. *Zigbee* найкраще підходить для особистих мереж з невеликими пристроями, які споживають мало енергії, мають низьку пропускну здатність і використовуються в замкнутому діапазоні.

Мережі з низьким енергоспоживанням і широкою зоною охоплення (*LPWAN*). Забезпечують зв'язок на відстані від 500 метрів, відрізняються мінімальним енергоспоживанням і використовуються для більшості пристроїв Інтернету речей. Наприклад, мережі з великим діапазоном і широкою зоною охоплення (*LoRaWAN*) забезпечують зв'язок між мобільними захищеними двонапрямними пристроями, що працюють від акумулятора. Нижче наведені найпоширеніші приклади.

– Інтернет речей *4G LTE* – такі мережі забезпечують високу потужність і малу затримку. Це відмінний варіант для сценаріїв Інтернету речей, в яких потрібно отримувати відомості чи оновлення в реальному часі.

– Інтернет речей *5G* – мережі Інтернету речей *5G* ще не реалізовані. Планується, що в майбутньому вони будуть підтримувати подальші інновації в Інтернеті речей, а також забезпечувати більш високу швидкість завантаження і підключення до набагато більшій кількості пристроїв в певній зоні.

– *Cat-0* – ці мережі на основі *LTE* - самі економічні. Вони забезпечують основу для технології *Cat-M*, яка замінить *2G*.

– *Cat-1* – цей стандарт для мобільного Інтернету речей в кінцевому підсумку замінить *3G*. Мережі *Cat-1* легко налаштовувати. Це відмінний варіант для додатків, яким потрібен голосовий або браузерний інтерфейс.

– *LTE Cat-M1* – ці мережі повністю сумісні з мережами *LTE*. Вони дозволяють оптимізувати вартість й потужність другого покоління мікросхем *LTE*, розроблених спеціально для додатків Інтернету речей.

– Вузькосмугове підключення. Цей стандарт радіочастотних технологій працює на основі підмножини стандарту *LTE*. Він орієнтований на покриття в приміщенні, а також забезпечує низькі витрати і тривалу роботу від акумулятора.

– *NB-IoT* / *Cat-M2* – Використовує модуляцію розширеного спектра з послідовним перебором (*DSSS*) для посилання даних безпосередньо на сервер. Це усуває необхідність у використанні шлюзу. Налаштування мереж *NB-IoT* коштує дорожче, але завдяки відсутності шлюзу їх робота супроводжується меншими витратами.

– *Sigfox* – Цей провідний глобальний постачальник мережевих послуг Інтернету речей пропонує бездротові мережі для підключення об'єктів з низьким енергоспоживанням, які безперервно створюють дані [7, 8].

1.1.2. Аналіз протоколів мережі *IoT*

Далі розглянемо протоколи. які використовуються в мережі *IoT*.

Пристрої Інтернету речей обмінюються даними за допомогою протоколів Інтернету речей. Протокол *IP* - це набір правил, що визначають спосіб посилання даних в Інтернет. Протоколи Інтернету речей гарантують, що інформація з одного пристрою або датчика буде зчитуватися і правильно інтерпретуватися іншим. Оскільки існує безліч пристроїв Інтернету речей, важливо використовувати потрібний протокол в потрібному контексті.

Тип використовуваного протоколу Інтернету речей залежить від рівня архітектури системи, на якому повинні передаватися дані. Модель *OSI* надає карту різних рівнів, на яких виконується відправка й отримання даних. Кожен протокол в архітектурі системи Інтернету речей забезпечує взаємодію між пристроями, між пристроєм і шлюзом, між шлюзом і центром обробки даних або між шлюзом і хмарою, а також обмін даними між центрами обробки даних.

Рівень додатків служить інтерфейсом для обміну даними між користувачем і пристроєм.

– Розширений протокол управління чергою повідомлень (*AMQP*) – рівень обладнання, на якому встановлюється взаємодія між ПО проміжного шару для обміну повідомленнями. Це забезпечує спільну роботу різноманітних систем і додатків. Так формується стандартизована система обміну повідомленнями в промисловому масштабі.

– Обмежений протокол додатків (*CoAP*) – мережевий протокол з обмеженням пропускної спроможності і доступу до мережі, призначений для пристроїв з обмеженою потужністю. Протокол забезпечує обмін даними між комп'ютерами. Крім того, *CoAP* - це протокол передачі документів, який працює за допомогою протоколу *UDP*.

– Служба розподілу даних (*DDS*) – універсальний спеціальний робочий протокол зв'язку, який дозволяє виконувати будь-які дії - від запуску невеликих пристроїв до підключення високопродуктивних мереж. *DDS* спрощує розгортання, підвищує надійність і зменшує складність.

– *MQTT* – протокол обміну повідомленнями, призначений для неінтенсивного обміну даними між комп'ютерами. В основному використовується для підключень з низькою пропускною спроможністю до віддалених розташуванням. *MQTT* використовує підписаний видавцем шаблон. Це ідеальний варіант для невеликих пристроїв, що вимагають ефективної пропускної спроможності і використання акумулятора.

Рівень транспортування забезпечує і захищає обмін даними між рівнями.

– Протокол *TCP* – головний протокол для більшості типів підключення до Інтернету. Забезпечує обмін даними між вузлами. Для цього великі набори даних розбиваються на окремі пакети. При необхідності пакети повторно відправляються і перекомпонуйте.

– Протокол *UDP* – Протокол зв'язку для взаємодії між процесами. Працює на основі протоколу *IP*. Протокол *UDP* підвищує швидкість передачі даних по протоколу *TCP*. Це кращий варіант для додатків, що вимагають передачі даних без втрат.

Рівень мережевий допомагає окремим пристроям взаємодіяти з маршрутизатором.

– *6LoWPAN* – версія *IPv*6 з меншою потужністю, яка скорочує час передачі.

– *IPv6* Це останнє оновлення протоколу *IP* для маршрутизації трафіку через Інтернет, а також визначення та виявлення пристроїв в мережі.

На канальному рівні дані передаються в межах архітектури системи. При цьому визначаються і виправляються помилки, виявлені на фізичному рівні.

– *IEEE 802.15.4* – Стандарт мобільних пристроїв для бездротового підключення з низьким енергоспоживанням. Він використовується з *Zigbee, 6LoWPAN* і іншими стандартами для створення бездротових мереж.

– *LPWAN* – Мережа такого типу забезпечує зв'язок на відстані від 500 метрів. *LoRaWAN* - це приклад мережі *LPWAN*, оптимізованої для енергозбереження [7, 8].

На таблиці 1.1 наведена порівняльна характеристика основних мережевих технологій.

Таблиця 1.1

Порівняльна характеристика основних мережевих технологій

Технологія	*ZigBee* (*IEEE* 802.1.5.4)	*Wi-Fi* (*IEEE* 802.11*b*)	*Bluetooth* (*BLE*) (*IEEE* 802.15.1)
Частотний діапазон, ГГц	2,4—2,483	2,4—2,483	2,4—2,483
Пропускна здатність, Кбіт/сек	250	11000	723,1
Розмір стеку протоколу, Кбайт	32-64	Більше 1000	Більше 250
Час неперервної автономної роботи від акумулятора, дні	100 – 1000	0,5 – 5	1 – 10

Продовження таблиці 1.1

Технологія	*ZigBee* (*IEEE* 802.1.5.4)	*Wi-Fi* (*IEEE* 802.11*b*)	*Bluetooth* (*BLE*) (*IEEE* 802.15.1)
Максимальна кількість вузлів у мережі	65536	10	7
Діапазон дії, м (середнє значення)	10 – 100	20 – 300	10 – 100

На фізичному рівні встановлюється комунікаційний канал, який здійснює підключення пристроїв у зазначеному середовищі [9].

– *Bluetooth* з низьким енергоспоживанням (*BLE*) – дозволяє значно скоротити енергоспоживання і витрати. Підтримує такий же діапазон підключення, як і класична технологія *Bluetooth. BLE* працює безпосередньо в мобільних операційних системах. Ця технологія швидко набуває популярності у сфері побутової електроніки, бо вона економічна і забезпечує тривалу роботу від акумулятора.

– *Ethernet* – це економічне дротове з'єднання, яке забезпечує швидке підключення до даних, а також меншу затримку, відносно безпровідних протоколів. Це дає змогу передавати дані, наприклад відеопотік, у режимі реального часу, а також дає змогу системі швидко реагувати на зовнішні, по відношенню до системи, зміни і повідомляти про це користувача.

– "Довготривалий розвиток" (*LTE*) – стандарт бездротового широкосмугового зв'язку для мобільних пристроїв і терміналів даних. *LTE* збільшує потужність і швидкість бездротових мереж, а також підтримує багатоадресні і широкомовні потоки.

– Зв'язок ближньої дії (*NFC*) – набір протоколів зв'язку, що стосуються використання електромагнітних полів. Дозволяє двом пристроям обмінюватися даними на відстані чотирьох сантиметрів один від одного. Пристрої з підтримкою

NFC працюють як карти ключів посвідчень і зазвичай використовуються для смарт-карт, безконтактних мобільних платежів і бронювання квитків.

– Радіочастотна ідентифікація (*RFID*) – використовує електромагнітні поля для відстеження електронних тегів, відомості про яких не можна отримати інакше. Сумісне обладнання забезпечує електроживлення та взаємодіє з цими тегами, зчитуючи інформацію для ідентифікації та перевірки автентичності.

– *Wi-Fi / 802.11* – стандартний варіант для дому та роботи. Цей варіант економічний. Але він підходить не для всіх сценаріїв через обмежений діапазон дії й постійного споживання енергії [7, 8].

1.2. Аналіз пристроїв мережі *IoT*

Проаналізуємо, які пристрої використовуються для побудови мережі інтернету речей:

– вбудовані системи – тут використовується як обладнання, так і програмне забезпечення. Такі системи управляють певними функціями, пов'язаними з більшою системою. Вбудовані системи створюються на основі мікропроцесорів або мікроконтролерів;

– інтелектуальні системи – ці пристрої можуть виконувати обчислення і часто оснащені мікро контролером;

– мікроконтролер (*MCU*) – ці невеликі комп'ютери впроваджуються в мікросхеми і оснащені ЦП, ОЗУ і ПЗУ. Вони містять елементи, необхідні для виконання простих завдань. Але мікроконтролери мають більш обмежену потужність, ніж мікропроцесори;

– мікропроцесор (*MPU*) – функції ЦП можна розміщувати на одній або декількох інтегральних мікросхемах. Хоча для виконання завдань мікропроцесорах потрібні периферійні пристрої, витрати на обробку значно скорочуються, так як в них входить тільки вартість ЦП;

– пристрої, які не призначені для обчислень – ці пристрої тільки встановлюють підключення і передають дані. У них немає можливості виконувати обчислення.

– перетворювачі – фізичні пристрої, які перетворюють один вид енергії в інший. В контексті пристроїв Інтернету речей до них відносяться внутрішні датчики і виконавчі пристрої, що передають дані в міру того, як об'єкти взаємодіють з їх середовищем [8, 10].

1.3. Аналіз архітектури *IoT*

Для подальшого вибору необхідного обладнання та розуміння роботи системи *IoT* проведемо аналіз архітектури *IoT*.

Говорячи про Інтернет речей, велика увага приділяється його потенціалу. Новини про те, що *IoT* буде в змозі зробити і полегшити наше життя продовжуються, але для багатьох може здатися, що ці піднесені бачення не втілити в реальність так швидко, як би вони могли. А проте, велика зміна трапляється, але все ж таки не в гігантських стрибках, а в дрипах і сидіннях. Причина цього досить проста, але вона, як правило, залишається поза увагою громадськості: саме властиве різноманіття систем *IoT* стримує прогрес і часто заважає зробити все, що пов’язано.

*Я*к одна з двох, мабуть, найбільших проблем, що стоять перед *IoT* (інша - безпека), фрагментація є основою Інтернету речей через різноманітний характер речей, які вона прагне з'єднати. Для запуску будь-якої системи *IoT* потрібно використовувати всі ресурси, обладнання, програмне забезпечення та системи, якими б різними вони не були, в єдину структуру, щоб сформувати інтегроване, надійне та економічно ефективне рішення. Говорячи простою мовою, кожному розгортанню *IoT* потрібна міцна архітектура *IoT*, щоб мати можливість виконувати задумане призначення; результуюча ефективність та придатність системи значною мірою залежить від якості розробленої інфраструктури [8, 10].

1.3.1. Основні складові архітектури *IoT*

Хоча кожна система *IoT* відрізняється, основа для кожної архітектури Інтернету речей, а також загального потоку процесів обробки даних приблизно однакова. Перш за все, вона складається з речей, які є об'єктами, підключеними до Інтернету, які за допомогою вбудованих датчиків і виконавчих механізмів здатні відчувати навколишнє середовище та збирати інформацію, яка потім передається шлюзам *IoT*. Наступний етап складається із систем збору даних *IoT* та шлюзів, які збирають величезну масу необроблених даних, перетворюють їх у цифрові потоки, фільтрують та попередньо обробляють, щоб вони були готові до аналізу. Третій шар представлений крайовими пристроями, відповідальними за подальшу обробку та посилений аналіз даних. На цьому рівні також можуть вступити технології візуалізації та машинного навчання. Після цього, дані передаються в центри обробки даних, які можуть бути або хмарними, або локально встановленими. Саме тут дані зберігаються, управляються та поглиблено аналізуються для отримання практичної інформації. Це чотири шари архітектури *IoT*, докладно описані на рис. 1.2 [11].

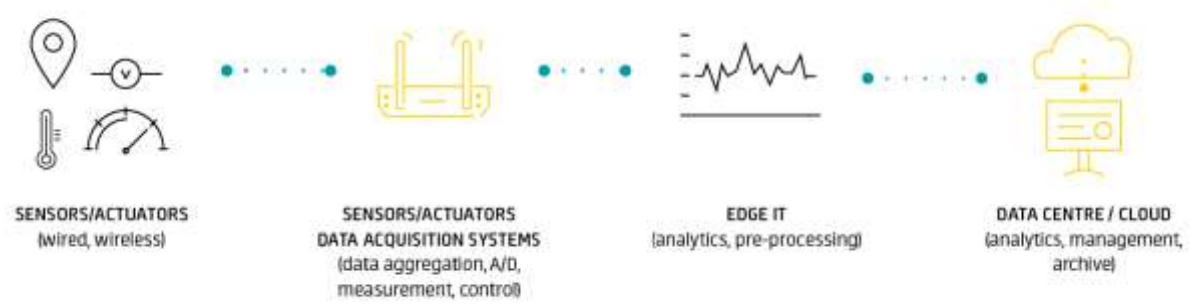

Рис. 1.2. Основні складові архітектури *IoT*

1.3.2. Сенсори та контролери

Як основа для кожної системи *IoT,* підключені пристрої відповідають за надання суті Інтернету речей, яка є даними. Для підбору фізичних параметрів у зовнішньому світі або всередині самого об’єкта їм потрібні датчики (рис. 1.3). Вони

можуть бути вбудовані в самі пристрої або реалізовані як самостійні об'єкти для вимірювання та збору телеметричних даних. Наприклад, подумайте про сільськогосподарські датчики, завдання яких полягає у вимірюванні таких параметрів, як температура та вологість повітря та ґрунту, рівень *pH* ґрунту або вплив сонячного світла на культури.

Ще одним незамінним елементом цього шару є виконавчі механізми. Тісно співпрацюючи з датчиками, вони можуть перетворювати дані, генеровані розумними об’єктами, у фізичні дії. Уявім розумну систему поливу з усіма необхідними датчиками на місці. На основі вхідного сигналу, даного датчиками, система аналізує ситуацію в режимі реального часу і наказує приводам відкрити вибрані водяні клапани, розташовані в місцях, де вологість ґрунту нижче встановленого значення. Клапани залишаються відкритими до тих пір, поки датчики не повідомлять про відновлення значень за замовчуванням. Очевидно, що все це відбувається без єдиного втручання людини.

Важливим є також те, що підключені об'єкти повинні мати можливість не тільки двосторонньо взаємодіяти з відповідними шлюзами або системами збору даних, але також мати можливість розпізнавати та розмовляти між собою для збору та обміну інформацією та співпраці в реальному часі, щоб використовувати значення всього розгортання. Особливо у випадку обмежених ресурсів та пристроїв, що працюють від акумуляторів, досягнення цього не є простим завданням, оскільки такий зв’язок вимагає великої обчислювальної потужності та споживає дорогоцінну енергію та пропускну здатність. Отже, надійна архітектура може забезпечити ефективне управління пристроєм лише тоді, коли вона використовує спеціальні, безпечні та легкі протоколи зв'язку, такі як *Lightweight M2M* який став провідним стандартним протоколом для управління малопотужними легкими пристроями, що характерно для багатьох випадків використання *IoT* [11].

Рис. 1.3. Апаратні складові *IoT*

1.3.3. Шлюзи та збір даних

Попри те, що цей рівень все ще функціонує в безпосередній близькості від датчиків та виконавчих механізмів на певних пристроях, важливо описати його як окремий етап архітектури *IoT*, оскільки він має вирішальне значення для процесів збору даних, фільтрації та передачі до крайової інфраструктури та хмарних платформ. З огляду на величезний обсяг введення та виведення даних, який може створити розгортання мільйонних пристроїв, можливості збору, відбору та транспортування даних повинні бути в центрі уваги. Як посередники між пов'язаними речами та хмарою та аналітикою, шлюзи та системи збору даних забезпечують необхідну точку зв'язку, яка пов'язує решту шарів (рис. 1.4).

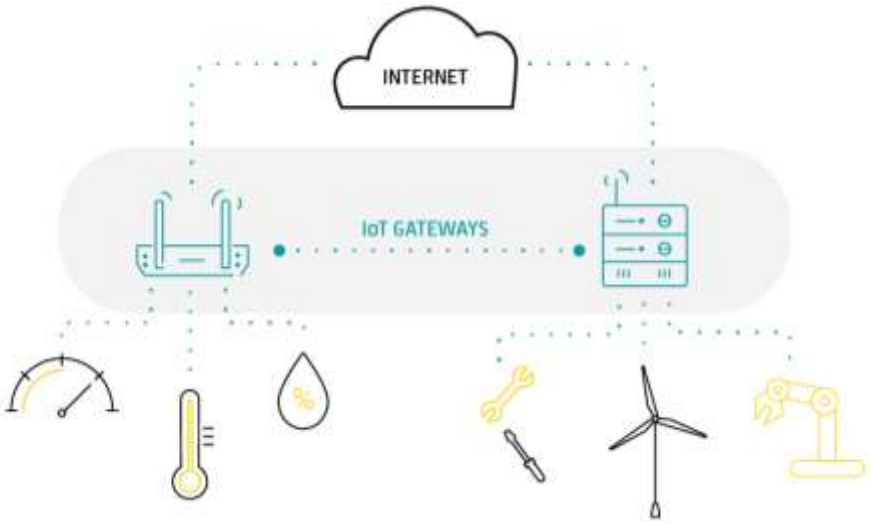

Рис. 1.4. Шлюзи та системи збору даних

Розташовані на межі світів *OT* та *IT,* шлюзи полегшують зв'язок між датчиками та рештою системи, перетворюючи дані датчиків у формати, які легко передаються та використовуються для інших компонентів системи. Більше того, вони здатні контролювати, фільтрувати та відбирати дані, щоб мінімізувати обсяг інформації, яку потрібно пересилати в хмару, що позитивно впливає на витрати на передачу мережі та час відгуку. Таким чином, шлюзи забезпечують місце для локальної попередньої обробки даних датчиків, які стискаються в корисні пакети, готові до подальшої обробки.

Ще одним аспектом, який підтримують шлюзи, є безпека. Оскільки шлюзи відповідають за управління інформаційним потоком в обох напрямках, за допомогою належних засобів шифрування та захисту вони можуть запобігти витоку хмарних даних *IoT,* а також зменшити ризик зловмисних зовнішніх атак на пристрої *IoT* [11].

1.3.4. Центр обробки даних

*Я*кщо датчики є нейронами, а шлюз є основою *IoT,* то хмара – це мозок у тілі Інтернету речей. На відміну від крайових рішень, центр обробки даних або хмарна система призначена для зберігання, обробки та аналізу великих обсягів даних для глибшого розуміння за допомогою потужних механізмів аналізу даних та механізмів машинного навчання, які крайові системи ніколи не зможуть підтримати (рис. 1.5).

Побачивши посилення впровадження (особливо в промисловій архітектурі *IoT*) протягом останніх кількох років, хмарні обчислення сприяють вищим показникам виробництва, скороченню незапланованих простоїв та споживання енергії та багатьом іншим вигодам для бізнесу.

*Я*кщо хмара забезпечена належними рішеннями для користувацьких програм, хмара може надавати бізнес-аналітику та варіанти презентацій, які допомагають людям взаємодіяти з системою, контролювати та контролювати її та приймати обґрунтовані рішення на основі звітів, інформаційних панелей та даних, що переглядаються в режимі реального часу [11].

Рис. 1.5. Центр обробки даних

1.4. Аналіз задач моніторингу у мережі *IoT*

Моніторинг починається зі збору даних шляхом контролю системи, з якою працює людина. У деяких сценаріях *IoT* системою, яка контролюється, можуть бути не самі пристрої, а зовнішнє середовище та процес, що стосуються пристрою. В інших сценаріях може зацікавити моніторинг працездатності самих пристроїв окремо.

Розглянемо завдання моніторингу людини-велосипедиста, яка їде дорогою. Існує багато різних частин загальної системи, за якими можна стежити. Деякі можуть бути внутрішніми в системі, наприклад, частота серцевих скорочень або пітливість велосипедиста. Інші можуть бути зовнішніми для велосипедиста, наприклад, нахил дороги або зовнішня температура та вологість. Ці внутрішні та зовнішні цілі моніторингу можуть співіснувати. Методології та інструменти можуть збігатися, але людина може розпізнати ці різні домени – лікар може піклуватися про інші виміри, ніж велосипедна механіка. Інструменти моніторингу можна використовувати для створення власних подань моніторингу.

Показники апаратного забезпечення пристрою – це вимірювання самого обладнання або фізичного пристрою, як правило, з якимсь вбудованим датчиком. Ці вимірювання часто важливі для розуміння продуктивності апаратного забезпечення пристрою для задоволення його конструкторської функції, але можуть бути менш актуальними для тих, хто використовує пристрій. Ці показники можуть включати такі речі, як напруга акумулятора або потужність радіосигналу.

Програмне забезпечення, що працює на пристроях, включає прикладне програмне забезпечення, а також саме системне програмне забезпечення, яке може бути операційною системою або шарами мережевого стека або драйверами пристроїв. Хоча термін прошивка часто використовується для опису всього програмного забезпечення на пристрої, він використовується тут лише для опису нижчих рівнів програмного забезпечення, які є більш специфічними для пристрою чи платформи, ніж для програми. Приклади показників прошивки включають частоту перезавантаження та тиск пам'яті.

Вимірювання навколишнього середовища за допомогою датчиків часто є тим, про що люди думають стосовно пристроїв *IoT*. Очевидні показники тут – це ті, які можна безпосередньо виміряти за допомогою приєднаного датчика. Але датчики можуть також охоплювати локальний бездротовий зв'язок. Прикладами є звичні типи датчиків, такі як датчики для вимірювання температури або світла.

Рішення *IoT* - це складна система, що включає програмні компоненти, що працюють як на пристрої, так і в хмарі. Розуміння того, як взаємодіють ці дві системи, вимагає розуміння того, до якої інформації має доступ кожна сторона та як поєднати два середовища виконання програмного забезпечення. Наприклад, таке питання, як «скільки часу потрібно повідомлення, щоб дістатися до хмари?». Це не те, що може сказати одна хмарна служба, оскільки хмарна служба знає лише про прибуття повідомлення. Щоб отримати потрібну інформацію, можна надіслати час публікації пристрою, а потім обчислити метрику на основі порівняння її з часом прибуття. Прикладом можуть бути тайм-аути підключення, передані байти та помилки автентифікації [12].

Повне рішення моніторингу вимагає контролю як основних, так і допоміжних компонентів. Моніторинг коду програми на пристрої є прикладом моніторингу білих скриньок, де видно, як функціонує програма. Ймовірно, може бути бажання включити певний моніторинг чорних ящиків. Наприклад, програмне забезпечення для моніторингу може перевіряти *API* та інші хмарні служби, від яких залежить рішення керівника. Коли він намагаєтеся розв’язати проблему, наявність цих зондів для чорного ящика може призвести до набагато швидшого вирішення. Приклади для цієї

категорії включають показники, які контролюють допоміжні послуги, такі як бази даних, або які відстежують результати завдань *ETL* або інших завдань з обробки даних.

Керівник стежить за системою, проводячи вимірювання та спостереження, а потім збираючи ці вимірювання у названі метрики, які зберігаються як серії. У *IoT* часто використовується датчик для спостереження за навколишнім середовищем. Бібліотеки контрольно-вимірювальних приладів для моніторингу часто використовують логічні програмно-визначені вимірювальні прилади, такі як датчики. Для моніторингу *IoT* фізичні датчики, такі як датчик температури, можуть бути безпосередньо зіставлені з логічною метрикою.

Метрики також можуть представляти більш абстрактний рівень функціональності, і вони передбачають певну попередню обробку. Наприклад, датчик акселерометра може постійно вимірювати силу сили *G*, але метрика, яка є цікавою, полягає в тому, коли відбувається подія "тремтіння". Локальна обробка може забезпечити необхідний алгоритм для цих "тремтячих" подій і підтримувати лише їх лічильник, не повідомляючи про всі вихідні дані акселерометра.

Визначившись, які системи контролюються, слід подумати, чому здійснюється моніторинг. Система, з якою ви працюєте, надає корисну функцію, а метою моніторингу є забезпечення того, щоб функція чи послуга працювали належним чином.

Коли користувач стежить за послугами програмного забезпечення, він шукає вимірювання навколо продуктивності цієї послуги, такі як час відгуку веб-запиту. Коли послуга є фізичним процесом, таким як обігрів приміщення, виробництво електрики або фільтрація води, він може використовувати пристрої для вимірювання цього фізичного процесу та проводити вимірювання таких речей, як години роботи двигуна або час циклу. Незалежно від того, використовує користувач пристрій як засіб виключно для контролю фізичного процесу, чи сам пристрій виконує послугу, він хоче провести ряд вимірювань щодо самого пристрою.

Вимірювання, зроблені в точці контрольно-вимірювальних приладів, призводять до того, що метрика надсилається і реєструється в централізованій системі

моніторингу. Метрики можуть бути низького рівня (прямі та необроблені) або високого рівня (абстрактні). Метрики вищого рівня можуть бути обчислені на основі метрик нижчого рівня. Спершу слід подумати про високоякісні показники, необхідні для того, щоб забезпечити надання послуг. Потім користувач може визначити, які метрики нижчого рівня потрібно зібрати, щоб підтримати задані цілі моніторингу. Не всі метрики корисні, і важливо не потрапити в пастку вимірювання речей лише тому, що користувач може або тому, що вони виглядають вражаюче (так звані "метрики марнославства").

Хороші показники мають такі характеристики:

– Вони можуть діяти. Вони інформують тих, хто працює або переглядає послугу, коли їм потрібно змінити її поведінку.

– Вони порівняльні. Вони порівнюють продуктивність чогось із часом або між групами пристроїв, члени яких знаходяться в іншому місці або мають різні версії мікропрограми чи обладнання.

– Вони зрозумілі та актуальні в оперативному контексті. Це означає, що на додаток до вихідних значень, таких як підсумки, вони можуть надавати таку інформацію, як коефіцієнти та показники.

– Вони надають інформацію з правильним дозволом. Користувач може вибрати, як часто вибиратиме дані, як часто буде складати звіти та як він буде складати середні показники, відставати та графікувати показники. Усі ці значення потрібно вибирати в контексті домену служби, яку він намагається надати. Наприклад, надання 1-секундної звітності про ємність *SD*-карти пристрою *IoT* генерує багато непотрібних деталей та обсягу. І дивлячись лише на середнє навантаження центрального процесора за годину, воно поглине та приховає короткі стрибки активності. Можливі періоди усунення несправностей, коли користувач набирає точність метрик для кращої діагностики. Але вихідна роздільна здатність повинна відповідати тому, що йому потрібно, щоб задовольнити його потреби у моніторингу.

– Вони висвітлюють різницю між симптомами, причинами та співвідношеннями між тим, що вимірює користувач. Деякі вимірювання є провідними

показниками проблеми, і користувач може побудувати попередження на них. Інші вимірювання є відстаючими показниками і допомагають зрозуміти, що сталося; ці виміри часто використовують для дослідницького аналізу.

Спільнота моніторингу розробила ряд методологій для програмного забезпечення та розподілених систем, як зведено у таблиці 1.2.

Таблиця 1.2

Методології для програмного забезпечення та розподілених систем

Метод	Сигнали
Чотири золоті сигнали (з книги *Google Engineering Reliability Engineering*)	Затримка Рух Помилки Насиченість
ЧЕРВОНИЙ метод , який фокусується на вимірюванні речей, про які турбуються кінцеві користувачі	Оцініть Помилки Тривалість
Метод *USE* , який орієнтований на продуктивність та вузькі місця у системі.	Використання Насиченість Помилки

Моніторинг призначений для надання сукупних поглядів або поглядів на те, як показник змінюється з часом. Але він не призначений для фіксації всіх критичних точок даних або подій, навіть для операцій. Індивідуальні вимірювання не вважаються критичними при розробці системи моніторингу [12].

Це контрастує із реєстрацією пристрою, яка більше стосується деталей конкретних подій з окремого пристрою. Обидві точки зору щодо системи важливі. Наприклад, моніторинг може дати користувачу сукупне уявлення про кількість помилок з часом, організовану за кодом помилки та версією мікропрограми. На

відміну від цього, реєстрація реєструє деталі про те, яким був контекст кожної помилки від конкретного пристрою.

Попри те, що користувач зазвичай збирає метрики, щоб допомогти в операціях, телеметрія, яка використовується для моніторингу, також може бути використана в інших зусиллях збору даних. Наприклад, користувач може використовувати значення з метрик моніторингу для більш тривалого аналізу або для відображення поточного стану користувачам у мобільному додатку як частину регулярного використання пристрою.

Можливо, іноді користувач захоче використовувати дані моніторингу для аналітики поза моніторингом. Але він повинен робити це уважно і переконатись, що цей варіант використання не заважає процесу моніторингу. Наприклад, йому може знадобитися визначити тенденції використання продукції за даними моніторингу. Але він не хоче створювати обчислювальну інформаційну панель, яка працює в тому ж інструменті, який є частиною критичної системи моніторингу. Роздільна здатність метрик також, як правило, достатньо висока, щоб тривале збереження даних, як правило, не було практичним, і дані метрик часто зберігаються лише на періоди порядку тижня. Користувач може вирішити обидві ці проблеми, скориставшись *Pub* / *Sub* для дублювання потоків метрик, що, у свою чергу, дозволяє вам повторно використовувати дані телеметрії для спостереження [12].

Висновки до розділу

У даному розділі проведений аналіз мережі інтернету речей з метою виявлення особливостей мережі інтернету речей. Проаналізовано, які протоколи, технології та типи мереж застосовуються для побудови мережі *IoT*, на якій архітектурі побудовані дані мережі, які пристрої беруть участь у роботі мережі. Проаналізовано, які задачі виконує моніторинг і навіщо його застосовувати, а також проаналізовано головні проблеми та задачі кібербезпеки мережі інтернету речей.

РОЗДІЛ 2
АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ

Для забезпечення функціонування системи управління та моніторингу потрібно вибрати таке апаратне забезпечення, яке забезпечуватиме виконання поставлених задач та цілей. Для цього пропонується наведене нижче обладнання.

Інтернет речей - це не тільки безліч різних приладів і датчиків, об'єднання між собою дротяними і бездротовими каналами зв'язку і підключати чинних до мережі Інтернет, а тісна інтеграція реального та віртуального світів, в середовищі якої спілкування здійснюється між людьми і пристроями.

Рішення на базі Інтернету речей стають зараз все більш затребуваними саме тому, що дають постачальникам «розумних» рішень можливість отримувати додатковий прибуток, - «розумна» поведінка може дати істотний приріст «корисності», споживчої вартості пристрою або системи. Так, вентилятор, який «сам» вимикається при досягненні потрібної температури, економить власнику електроенергію і тому може коштувати для нього дорожче. А вентилятор, який ще й «бачить», коли в приміщенні є люди, а коли ні, - цінний ще більше.

Але як техніка може стати «розумною»? По-перше, за рахунок, власне, своєї конструкції - ця конструкція може бути такою, що поведінка системи буде виглядати розумною.

По-друге, за рахунок «інтелектуалізації» - оснащення системи пристроями збору інформації, її обробки і прийняття рішень. Такий підхід дозволяє забезпечити досить складне і «розумне» поведінку набагато простішими способами, ніж за рахунок створення відповідної конструкції.

Нарешті, третій шлях - поведінка системи стає «розумною» внаслідок того, що вона взаємодіє з іншими системами. Так, для економії енергії системі опалення потрібно короткостроковий прогноз погоди. Цей прогноз можна отримати, встановивши відповідні датчики і систему обробки інформації з них, здатну прогнозувати погоду (міні-метеостанцію), а можна просто запросити погоду в

Інтернеті. І в тому, і в іншому випадку поведінка системи оточення буде виглядати «розумним».

Важливо, що в останньому прикладі з точки зору замовника система веде себе практично однаково - відповідно, замовник готовий заплатити за цю функціональність одну і ту ж ціну. Однак для постачальника такої системи організація підключення її до Інтернету буде коштувати значно дешевше, ніж розробка інтелектуальної метеостанції.

Завдяки інтелекту і конективності устаткування з'являється новий набір функцій. Їх можна розділити на чотири групи:

- моніторинг;
- управління;
- оптимізація;
- автономність.

Кожна функція, важлива і сама по собі, виявляється свого роду сходинкою для наступного рівня. Наприклад, функція моніторингу є основою для управління, оптимізації та автономності техніки. У результаті функція моніторингу дозволяє значно полегшити та оптимізувати користування системою, яка моніториться. Компанія може вибирати такий набір функцій, щоб її продукція була максимально корисною для споживача і таким чином, вона може зміцнювати свою конкурентну позицію.

Візьмемо, наприклад, автоматичну теплицю, яка самостійно здійснює полив, підтримання потрібної температури, рівня освітленості та ін. Така теплиця користуватиметься попитом тими, хто не хоче витрачати багато часу на догляд за рослинами, а також може не мати для цього можливості в періоди довгої відсутності персоналу: відрядження, відпустки і т.д.

Однією з проблем, яку вирішує функція моніторингу є усунення занепокоєння щодо того, чи все в порядку *с* рослинами під час його відсутності: чи є вода в системі, що не вимикалося електрику, чи може система вентиляції забезпечити потрібну температуру, якщо в приміщенні стало занадто жарко, і ін.

Клієнт напевно заплатить більше, якщо надати йому можливість в будь-який момент знати, які умови в його теплиці.

Таким чином, продажна вартість теплиці з функцією віддаленого моніторингу параметрів може зрости істотно, в той час, як її реалізація для виробниками буде досить простий. В результаті застосування технології інтернету речей дозволить виробнику отримати додатковий прибуток.

Ще вище споживча вартість буде у тій же теплиці, якщо додати функцію управління, щоб користувач міг дистанційно не тільки отримувати інформацію про умови в теплиці, а й змінювати їх на свій розсуд.

Напевно в теплиці підігрів включається автоматично, якщо температура падає нижче заданої межі, але, можливо, не варто його включати, якщо знати, що по прогнозом погоди зовсім скоро очікується підвищення температури? Таким чином, функція оптимізації за рахунок використання додаткової інформації дозволить заощадити гроші на утримання теплиці і отримати урожай з меншими витратами.

Нарешті, засобами Інтернету речей нескладно почати стежити за кількістю витрачених матеріалів - наприклад, добрив, - і автоматизувати їх замовлення, або контролювати стан елементів, що вимагають заміни або обслуговування: насосів, вентиляторів, що нагрівають, організувавши таким чином самодіагностику і самообслуговування теплиці аж до повної її автономності [13].

2.1. Мікрокомп'ютер *Raspberry Pi* та його альтернативи

З ціллю забезпечення моніторингу та досягнення поставлених цілей пропонується міні-ПК *Raspberry Pi*. Він здатний забезпечувати як і прості практичні завдання, наприклад вивчення комп'ютера і основ роботи з ним, інтернет-серфінгу, а також для програвання відео- та прослуховування аудіофайлів, так і більш складні задачі, наприклад бути частиною *IoT* системи.

Raspberry Pi - це мініатюрний, розміром з кредитну карту, комп'ютер вартістю близько 25 доларів за базову модель і 35 - за більш новітню, який має величезну популярність і розійшовся по світу в кількості більше 4,5 мільйонів штук.

Однією з основних переваг міні-ПК *Raspberry Pi* є співвідношення якості продуктів і його вартості. В першу чергу, міні-ПК *Raspberry Pi* відводиться роль комп'ютера, призначеного для вивчення з його допомогою базових інформаційних технологій в школі. Позиціонується міні-ПК *Raspberry Pi* як дешеве рішення для початківців розробників. З урахуванням задовільною потужності цього пристрою, низького енергоспоживання і малої собівартості його можна використовувати для створення особистого міні сервера.

Стосовно до статистичного управління процесами – методу моніторингу виробничих процесів з метою управління якістю продукції безпосередньо в процесі виробництва, міні-ПК *Raspberry Pi* є першим доступним технічним рішенням такого розміру, яке можна використовувати повсюдно для програмування на багатьох мовах і в якості мікроконтролерів для управління роботизованими пристроями.

Одна з головних і привабливих особливостей міні-ПК *Raspberry Pi* - наявність на платі апаратних портів введення / виводу *GPIO* (*General Purpose Input* / *Output*, інтерфейс вводу / виводу загального призначення), що відкриває перспективи використання його в робототехнічних проектах і пристроях «розумного будинку» [14].

Також на ринку існують альтернативні пристрої по відношенню до міні-ПК *Raspberry Pi.* Такими альтернативами є *Orange Pi Prime, Banana Pi M3, Rock64, ASUS Tinker board S, Libre Computer Renegade, Odroid H2, Arduino Mega 2560.*

Orange Pi Prime.

Відрізняється від *Raspberry Pi* 3 в кращу сторону наявністю 2 Гбайт ОЗУ і вбудованим в *SoC AllWinner H*5 відеоприскорювачем *Mali*-450 *GPU*, що дозволяє відтворювати 2*K* відео. Серед цікавих особливостей варто відзначити наявність ІК-приймача; платою можна управляти з пульта дистанційного керування або з деяких моделей стільникових телефонів з вбудованим ІЧ-світлодіодом. З нестандартного обладнання є також вбудований мікрофон і відеоінтерфейс *CSI*, що підтримує відеопотік до 1080*p* на швидкості 30 *fps*.

На платі розміром 98 × 60 мм знайшлося місце для роз'єму для карт пам'яті (до 32 Гбайт), *Wi-Fi* 802.11 *b* / *g* / *n*, *Bluetooth* 4.0, гігабітного *Ethernet*, чотирьох *USB* (три

USB 2.0 *Host* і один *USB* 2.0 *OTG*) і сорокаконтактной *GPIO*-гребінки. Є навіть окремо виведений *UART* з *TTL* рівнями, так що ви можете в терміналі спостерігати за деталями завантаження *Linux*. З аудіо обладнання, крім згаданого вище мікрофона, є ще лінійний вихід і аудіовихід в *HDMI*. Відеоприскорювач підтримує *OpenGL ES* 2.0 і *OpenVG* 1.1. Серед підтримуваних ОС присутні *Ubuntu, Debian* і *Android* 5.1.

Banana Pi M3.

Флагман *Banana Pi M*3 побудований на базі восьмиядерного *SoC Allwinner A*83*T* (процесори *ARM Cortex-A*7, графічний процесор *PowerVR SGX*544*MP*1), розганяється до 1.8 ГГц і працює в оточенні 2 Гбайт ОЗУ і 8 Гбайт флеш-пам'яті. Крім гігабітного *Ethernet*, двох *USB, Wi-Fi* 802.11 *b* / *g* / *n, Bluetooth* 4.0 і *HDMI*, на платі присутній *SATA*. Так само, як і у *Orange Pi Prime*, у *M*3 є ІК-приймач, відеоінтерфейс *CSI*, оцінний *UART*, мікрофон, лінійний вихід і аудіовихід в *HDMI*. На відміну від *Orange*, у *Banana* є інтерфейс дисплея *MIPI DSI*, об'єднаний з *I2C* для сенсорного екрану. Природно, є і сорокаконтактная *GPIO*-гребінка.

Rock64.

Одноплатний комп'ютер *Rock*64 комплектується вже 4 Гбайт ОЗУ, що обслуговують 64-х бітний *ARM Cortex A*53, відеопідсистема здатна впорається з потоком 4*K* на частоті 60 *fps*. Пристрій здатний харчуватися від *POE*. Графічна підсистема *ARM Mali* 450*MP*2 відповідає *OpenGL ES* 2.0, *OpenVG*1.1. На *Rock*64 перенести ОС *Debian, Cent OS, Fedora* і *Android* 8, взагалі, потрібно відзначити, що розробники і ентузіасти цього комп'ютера портувала на нього велику кількість ОС, що базуються на *Linux*. У *Rock*64 рясна, детальна документація і живе, активне ком'юніті, так що, з урахуванням непоганих апаратних специфікацій і щадний ціни, цей одноплатнік - непоганий претендент на заміну *Raspberry Pi* 3 в проектах з підвищеними вимогами до «заліза».

Розробники *Rock*64 додали 64 контакти *GPIO*, вивівши на них навіть сигнали *Ethernet*, так що, якщо плануєте робити на одноплатніке управління розгалуженою зовнішньої периферією, придивіться до цього комп'ютера пильніше. Крім того, є *USB*3.0.

ASUS Tinker board S.

Tinker побудований на базі *SoC Rockchip RK*3288 з счетверенний процесором *ARM Cortex-A*17 і працює під управлінням *TinkerOS* на базі *Debian*, можна встановити *Android*. Відеопроцесор *Mali-T*760 *MP*4 підтримує *OpenGL ES* 3.1, *OpenCL* 1.1, *Renderscript* і *Direct*3*D* 11.1.

Libre Computer Renegade.

Renegade, конструктивно розроблений настільки схожим на *Raspberry*, наскільки це тільки можливо; наприклад, є можливість розмістити *ROC-Rk*3328-*CC* прямо в корпусі *Raspberry Pi*.

SoC RK-3328 побудована на основі зчетвереної 64-х бітного процесора *ARM Cortex-A*53 з робочою частотою до 1.5 ГГц. *SoC* така ж, як і в *Rock*64, так що тут ви теж маєте той же *GPU Mali* 450*MP*2 з робочою частотою 500 МГц. При замовленні у вас є можливість варіювати обсяг ОЗУ, ви можете вибрати 1 Гбайт *DDR*4 (тоді комп'ютер обійдеться вам в $ 35), 2 Гбайт за $ 50 або 4 Гбайт за $ 80. З операційних систем на даний момент доступні *Ubuntu* 18.04, *Debian* 9, *OpenMediaVault* 4, *Station OS* і *Android* 7.1

Odroid H2.

*Odroid H*2, побудований на базі 64-х бітного 4-*х* ядерного *Intel Celeron Gemini Lake J*4105, цілком може претендувати на шматок ринкової ніші, яку займає одноплатними комп'ютерами на базі процесорів *ARM*. Якщо вам миліше *x*86-сумісні рішення, можливо, ця невелика плата розміром 110 × 110 мм, з пасивним охолодженням, *GPU Intel UHD Graphics* 600, шиною *PCI-E gen*2 і здвоєним *SATA* 6 Гбайт / с доведеться вам до вподоби.

За специфікації *Intel* максимальний розмір ОЗУ дорівнює 8 Гбайт (*SO-DIMM DDR*4 2400 МГц, в комплект не входить, купується окремо, як і для будь-якого «Пісюк»), але виробник *Odroid H*2 стверджує, що встромляв дві планки по 16 Гбайт (разом 32 Гбайт) і все прекрасно працювало. Зрозуміло, плата підтримує *Windows* 10 / *Linux x*64, *DirectX* 12, *OpenGL* 4.3, *OGL ES* 3.0, *OpenCL* 2.0.

У таблиці 2.1 наведемо порівняльну характеристику розглянутих пристроїв та міні-ПК *Raspberry Pi* [15].

Таблиця 2.1

Порівняльна характеристика *Raspberry Pi* та аналогів

Модель	*SoC*	Процессор	Графіка	Ядра	Частота	Розмір	Ціна
Raspberry Pi 3B+	*Broadcom BCM2837B0*	*ARM Cortex A53*	*Broadcom VideoCore IV*	4	1.4 ГГц	85.6 × 56.5 мм	$35
Banana Pi M3	*Allwinner A83T*	*ARM Cortex-A7*	*PowerVR 544MP1*	8	1.8 ГГц	92 × 60 мм	$68
Banana Pi M2 Zero	*Allwinner H2*	*ARM Cortex-A7*	*Mali400 MP2*	4	1.0 ГГц	60 × 30 мм	$18
Rock64	*Rockchip RK3328*	*ARM Cortex A53*	*Mali 450MP2*	4	1.5 ГГц	56 × 85 мм	$45
Asus Tinker board S	*Rockchip RK3288*	*ARM Cortex-A17*	*Mali T760 MP4*	4	1.8 ГГц	54 × 86 мм	$92
Libre Computer Renegade	*Rockchip RK-3328*	*ARM Cortex-A53*	*Mali 450MP2*	4	1.5 ГГц	85 × 56 мм	$80
Odroid H2	-	*Intel Celeron J4105*	*Intel UHD Graphics 600*	4	2.3 ГГц	110 × 110 мм	$111
Arduino Mega	-	*ATmega2560*	-	1	16 МГц	53 × 102 мм	$31

У таблиці 2.2 наведено порівняння підтримуваних інтерфейсів та оперативної пам’ті.

Таблиця 2.2

Порівняльна характеристика підтримуваних інтерфейсів *Raspberry Pi* та аналогів

Модель	ОЗУ	Флэш	*GPIO*	*USB*	*Ethernet*	*Wi-Fi*	*Bluetooth*
Raspberry Pi 3B+	1 Гбайт	Слот *MicroSDHC*	40	4	1000 Мбит/с	802.11 *b/g/n/ac* 2.4/5 ГГц	4.2 *LS BLE*

Продовження таблиці 2.2

Модель	ОЗУ	Флэш	*GPIO*	*USB*	*Ethernet*	*Wi-Fi*	*Bluetooth*
*Banana Pi M*3	2 Гбайт *LPDDR*3	8 Гбайт *eMMC*	40	3 (2 × 2.0, 1 × *OTG*)	1000 Мбит/с	802.11 *b*/*g*/*n*	4
*Banana Pi M*2 *Zero*	512 Мбайт *DDR*3	Слот *MicroSDHC*	40	1 × *USB* 2.0 *OTG*	-	802.11 *n*	4
*Rock*64	4 Гбайт *LPDDR*3	128 Мбайт	64	3 (3.0, 2.0, *OTG*)	1000 Мбит/с	802.11 *b*/*g*/*n*	4
Asus Tinker board S	2 Гбайт *LPDDR*3	16 Гбайт *eMMC*	40	4 × *USB* 2.0	1000 Мбит/с	802.11 *b*/*g*/*n*	4
Libre Computer Renegade	4 Гбайт *DDR*4	-	40	3 (1 × 3.0, 1 × 2.0)	1000 Мбит/с	-	-
*Odroid H*2	2 слота *DDR*4 *SO-DIMM*	128 Мбайт (*BIOS*), слот *eMMC*	-	4 (2 × 3.0, 2 × 2.0)	2 × 1000 Мбит/с	-	-
Arduino Mega	8 кбайт	256 кбайт	54	*USB-UART* преобразователь	-	-	-

Отже, згідно даного порівняння та у ході аналізу було визначено, що для даного дослідження оптимальним по ціні та своїм технічним показникам буде міні-ПК *Raspberry Pi* [15].

2.2. Підбір необхідного обладнання

Для початку потрібно скласти список необхідного обладнання для системи засобів моніторингу та керування елементами *IoT*.

Підбір необхідного обладнання є важливим кроком, оскільки знадобиться не тільки сам центральний пристрій, такий як міні-ПК *Raspberry Pi,* а й перефирійні

Для системи знадобиться сам центральний пристрій, в даному випадку міні-ПК *Raspberry Pi.* Для налаштування міні-ПК потрібно такі пристрої як клавіатура, маніпулятор типу миш, монітор, карта пам'яті. Повний список пристроїв необхідних для налаштування міні-ПК *Raspberry Pi* наведено нижче.

1. Одноплатний комп’ютер *Raspberry Pi.*
2. Монітор.
3. Клавіатура.
4. Маніпулятор типу «миша».
5. Адаптер живлення.
6. Карта пам’яті.

Також для роботи всієї системи небхідні датчики. Вони є невід’ємною частиною даної системи. Повний список необхідних датчиків наведено нижче.

1. Датчик температури *DS18B20.*
2. Датчик температури та атмосферного тиску *BPM180.*
3. Датчик освітленості *TSL2561.*

Так як міні-ПК *Raspberry Pi* не має у своєму *Wi-Fi* модулю, потрібно встановити *Wi-Fi* адаптер, який підключається шляхом підключення до порту *USB,* а саме *EDUP Nano EP N8531.* Адаптер *Wi-Fi* дозволяє підвищити мобільність даної системи і не бути прив’язаним до мережевого кабелю.

У таблиці 2.3 наведений повний перелік пристроїв, їх моделі та необхідна кількість.

Таблиця 2.3

Перелік додаткового обладнання

Тип пристрою	Модель	Кількість
Монітор	*Samsung S22R350FHI*	1
Клавіатура	*Logitech Keyboard K120*	1
Маніпулятор	*Logitech B110*	1
Wi-Fi адаптер	*EDUP Nano EP N8531*	1
Адаптер живлення	*Grand-X CH-65B*	1

Продовження таблиці 2.3

Тип пристрою	Модель	Кількість
Карта пам'яті	*Transcend Premium microSDHC* 16*GB Class* 10 *UHS-I*	1
Датчик температури	*DS18B20*	2
Датчик температури та атмосферного тиску	*BPM180*	1
Датчик освітленості	*TSL2561*	1

2.3. Технічні характеристики і можливості міні-ПК *Raspberry Pi*

Перші версії міні-ПК *Raspberry Pi*: моделі «A», «B» (рис. 2.1), «A+» і «B+» - оснащені процесором *Broadcom* BCM2835 архітектури *ARM 1* з тактовою частотою 700 МГц і модулем оперативної пам'яті на 256 (або 512) Мбайт, розміщеним за технологією *package-on-package* безпосередньо на процесорі. Модель «A» оснащена одним портом *USB* 2.0, модель «B» – двома такими портами, модель «B+» – чотирма. У моделей «B» і «B+» також є в наявності порт *Ethernet*. Крім основного ядра, процесор BCM2835 має у своєму складі графічне ядро з підтримкою *OpenGL ES* 2.0, апаратного прискорення та відео *Full* HD, а також ядро *DSP* (*Digital Signal Processor*, цифровий сигнальний процесор).

Комп'ютер працює через роз'єм *MicroUSB*, при цьому сила струму повинна становити мінімум 0,5-0,7A. При менших значеннях комп'ютер все ще може включитися, але буде йти в перезавантаження при запусці ресурсоємних задач. Отже, підключати плату краще не через хаб, а безпосередньо до *USB* порту комп'ютера або в розетку через спеціальний перехідник.

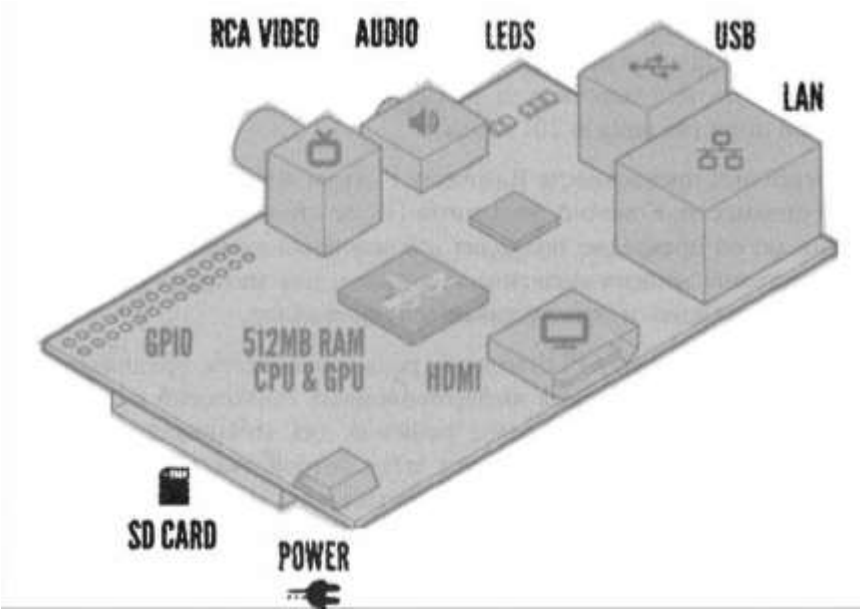

Рис. 2.1. Схема міні-ПК *Raspberry Pi*, Модель «B»

Ніяких кнопок ввімкнення / вимкнення на платі міні-ПК *Raspberry Pi* немає. Якщо необхідно пристрій запустити – потрібно підключити *US*B-живлення, для вимкнення - висмикнути шнур. Залишається сподіватися, що в майбутніх ревізіях міні-ПК *Raspberry Pi*, можливо, добавлять живлення по *Ethernet*, оскільки це один з найбільш частих запитів від користувачів.

Версії «A» і «B» пристрої оснащені слотом для карт пам'яті формату *SD*, версії «A +» і «B +» - слотом для карт формату *MicroSD*. Без карти пам'яті міні-ПК *Raspberry Pi* не включається, оскільки саме на ній повинна бути записана операційна система, - це все одно що намагатися запустити комп'ютер без жорсткого диска. Оскільки власної ОС у внутрішній пам'яті (як, наприклад, в телефонах) у міні-ПК *Raspberry Pi* немає, то з цього випливає один позитивний момент - пристрій практично неможливо перетворити в «цеглина»: після будь-якого невдалого експерименту досить перезаписати дистрибутив на карті пам'яті, і міні-ПК *Raspberry Pi* знову запрацює як новенький.

Для підключення дисплея на платі міні-ПК *Raspberry Pi* є відразу два інтерфейси: *RCA Video* (композитний) і H*DMI*. Застосовуючи відповідні перехідники, можна вийти і на більш традиційні: *VGA* і *DVI*. H*DMI* підтримує передачу як відео, так й звуку, але якщо буде потрібно окремий аудіо-канал, то і він присутній на платі

у вигляді стандартного міні-джека 3,5 мм. Підключення мікрофона також можливо, але для цього знадобиться знайти *US*B-пристрій, сумісний з міні-ПК *Raspberry Pi*.

Існуючі зараз моделі міні-ПК *Raspberry Pi* не мають модуля *Wi-Fi*, і для роботи з ними в Інтернеті знадобиться задіяти порт *Ethernet*. Оскільки фізично на платі він скомутований через *USB* 2.0, то забезпечує не гігабитну, а всього лише 100-мегабітну швидкість.

Чіпи процесора і графічного прискорювача не оснащені навіть найпростішими радіаторами, і після декількох годин роботи комп'ютера стає очевидним, чому його розробники зупинилися саме на цьому рішенні: плата нагрівається при роботі зовсім незначно - вона швидше тепла, ніж гаряча.

Частота процесора, як уже згадувалося, становить 700 МГц, і, в залежності від дистриб'ютора, його можна розігнати до 1000 МГц без втрати гарантії (можливий вибір і більш сприятливих режимів). Чіп пам'яті виробництва *Samsung* або *Hynix* напаяний прямо поверх основного чіпсета, так що збільшити *R*AM самостійно не вийде. При покупці варто звернути увагу на маркування цієї *SoC* (*System-on-a-Chip,* системи на кристалі): номер партії для «старих» версій моделі «B» з 256 Мбайт *R*AM починається з *K*4*P*2*G*, а у випуску з 512 Мбайт пам'яті - з K4P4*G*.

Відеоприскорювач *Broadcom VideoCore N* дозволяє навіть при такому слабкому процесорі декодувати відео 1080p *h*.264 з бітрейтом аж до 40 Мбіт / с. Для апаратного прискорення MPE*G*-2 і *VC*-1 ліцензії доведеться докуповувати окремо.

Плата оснащена індикаторами - п'ятьма світлодіодами: три з них демонструють активність і режим роботи *Ethernet*, а ще два сигналізують про наявність харчування і роботі з *SD*-картою.

А тепер перерахуємо набір низькорівневих інтерфейсів:

– по-перше, на платі є слоти *CSI*-2 для підключення камери і *DSI* – для підключення дисплею;

– по-друге. присутня колодка на 26 (40 для версії «B+») ліній вводу та виводу загального призначення *GPIO*. На них же реалізовані ін*г*ерфейси *UART*, консольний порт, шина *SPI* (*Serial Peripheral lnterface*, послідовний периферійний інтерфейс), PC (*lnter-Integrated Circuit*, послідовна шина даних для зв'язку

інтегральних схем) і 12*S* (*lntegrated lnter-chip Sound*, послідовна шина даних , що служить для з'єднання цифрових аудіопристроїв).

Втім, недоліків у міні-ПК *Raspberry Pi* теж вистачає. У ньому, наприклад, немає власних годин реального часу (*Real Time Clock, RTC*), тому єдиний спосіб отримання часу – це синхронізація з *N*тр-серверами. *SoC* містить в собі цифровий сигнальний процесор (*DSP*), але повного доступу до його *API* досі немає.

У новій версії – міні-ПК *Raspberry Pi* 2 *Model* B (рис. 2.2), замість процесора BCM2835 в ній встановлений чіп BCM2836. Відрізняється він від попередника наявністю чотирьох ядер A*R*M *Cortex-A*53 з набором інструкцій *AR*M*v*7 - проти *AR*M*v*6*k* у BCM2835. Тактова частота цього процесора складає 1400 МГц – не надто багато за мірками сучасних рішень *Qualcomm* і навіть *MediaTek*, але для переважної більшості *DI*У-проектів (*Do lt Your self*, «Зроби сам») її вистачить з лишком, тим більше що, нагадаємо, ядер на цей раз цілих чотири. Об'єм оперативної пам'яті нового пристрою збільшений з 512 Мбай*r* до 1 Гбайт. Відеоядро змін не зазнало, і це як і раніше *Broadcom VideoCore IV*. Розробники говорять про шестиразовий приріст продуктивності в багатопоточних тестах і про триразове в однопоточних [14].

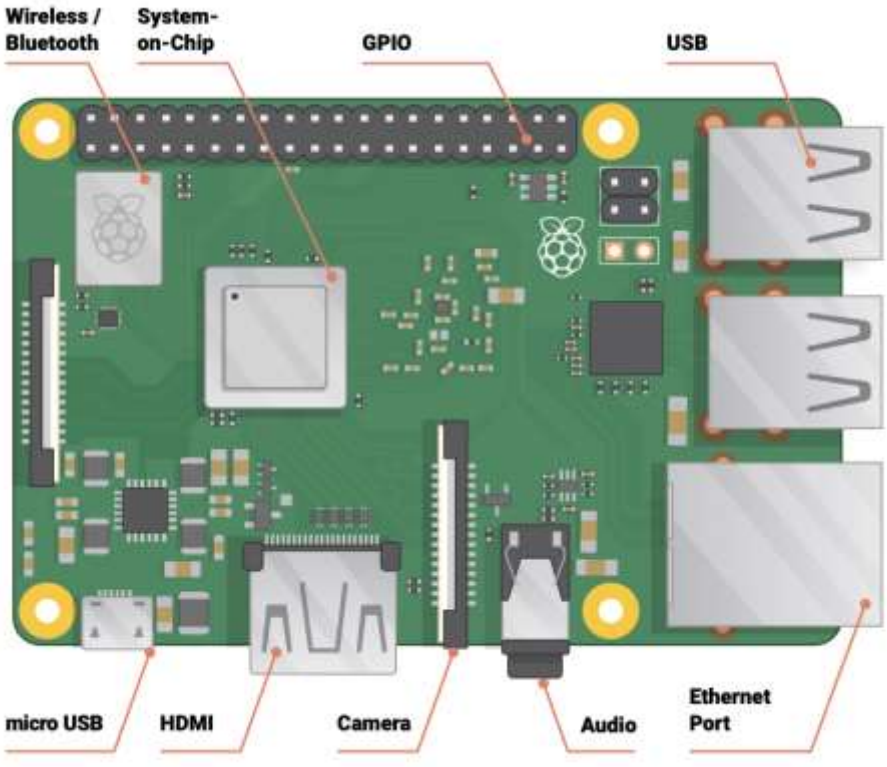

Рис. 2.2. Схема міні-ПК *Raspberry Pi* 2 *Model* B

2.4. Додаткове обладнання

Характеристики монітору:

- діагональ дисплею – 22 дюйм;
- частота оновлення – 60 гц;
- максимальна роздільна здатність – 1920 * 1080;
- тип матриці – *tn;*
- яркість дисплею – 250 кд/м²;
- інтерфейси – *VGA, HDMI;*
- контрастність – 1000:1;
- співвідношення сторін – 16:9;
- максимальна кількість кольорів – 16,7 млн. кольорів;
- споживна потужність:
  (a) напруга: 100 ~ 240в;
  (b) енергоспоживання:
    1. робочий режим: 23вт;
    2. Режим очікування: 0,3Вт.

Клавіатура:

- кількість клавіш – 104;
- тип клавіатури – мембранні;
- інтерфейс – *USB.*

Маніпулятор типу «миша» :

- джерело живлення – *USB* кабель;
- довжина кабелю – 1,8 м.;
- сумісність з ОС – ОС *Windows;*
- тип сенсору – оптичний.

Wi-Fi модуль:

- частота – *Wi-Fi*2412-2484 МГц;
- стандарт – 802.11 *b*/*g*/*n;*
- потужність – +20дБ;

– типи шифрування – *WEP, WPA, WPA*2;
– режими роботи – клієнт (*STA*), точка доступу (AP), клієнт + точка доступу (*STA+AP*);
– напруга живлення – 1,7-3,6 В;
– споживання току – 70 мА (пиковое значение 240 мА);
– температурний режим – від -40 до +70 °С.

Адаптер живлення:

– вхідна напруга – 220 В;
– вихідна напруга – 5 В;
– вихідний струм – 3.1 *A;*
– розмір – 88 х 45 х 25 мм.;
– вага – 73 г.;
– довжина кабелю – 114 см.

Карта пам’яті:

– розміри – 11 *mm x* 15 *mm x* 1 *mm;*
– форм-фактор – *microSD* карта пам'яті;
– тип *Flash* пам'яті:
  1. Пам'ять типу *MLC;*
  2. Пам'ять типу 3*D;*
– ємність – 8 ГБ / 16 ГБ / 32 ГБ / 64 ГБ / 128 ГБ;
– робоча напруга – 2.7В ~ 3.6В;
– тест на падіння – падіння з 1.5 метрів;
– робоча температура – стандартний діапазон-25 ° *C* (-13 ° *F*) ~ 85 ° *C* (185 ° *F*);
– швидкість читання – (макс.) До 90 Мб / с.

Датчик температури *DS18B20:*

– розрядність – 9-12 біт;
– час перетворення – 750*nS*(*max*);
– точність вимірювання ± 0.5% в області температур – -10 ….+85°*C*;
– напруга живлення для точності вимірювання ± 0.5% – 3,0-5,5*V*.

Датчик температури та атмосферного тиску *BPM180*:

– діапазон вимірювань: 300 - 1100 гПа (9000 ... -500 метрів над рівнем моря);
– діапазон виміру температур: -40 *to* + 85 ° *C* (точність ± 2 градуси);
– напруги живлення: 2 - 5В;
– споживання в режимі очікування: 0,1 мкА.

Датчик освітленості *TSL2561*:

– робоча напруга живлення - 3.3-5 В;
– чутливість – 65536 градацій;
– 2 вбудованих інтегруючих модуля АЦП;
– шина даних – *I2C*;
– освітленість – від 0.1 до 40000 люкс;
– точність – 0.1 люкс;
– розміри – 14 мм на 18 мм.

Wi-Fi адаптер *EDUP Nano EP N8531*:

– мережевий стандарт – *IEEE* 802.11*n*, *IEEE* 802.11*g* (*draft*), *IEEE* 802.11*b*;
– частотний діапазон – 2.4-2.4835*GHz*;
– робоча зона – до 100 м в приміщенні / до 200 м на відкритому просторі швидкість передачі даних до 150*Mbps*;
– число каналів – 13;
– інтерфейс – *USB*.

Розглянемо економічні показники обраного обладнання (табл. 2.4).

Таблиця 2.4

Економічні показники обладнання

Пристрій	Назва	Кількість	Ціна, грн.
Одноплатний комп’ютер	*Raspberry Pi*	1	1 600,00
Монітор	*Samsung S22R350FHI*	1	2 700,00
Клавіатура	*Logitech Keyboard K120*	1	450, 00
Маніпулятор	*Logitech B110*	1	340,00

Продовження таблиці 2.4

Пристрій	Назва	Кількість	Ціна, грн.
Wi-Fi адаптер	*EDUP Nano EP N*8531	1	157,00
Адаптер живлення	*Grand-X CH*-65*B*	1	165,00
Карта пам'яті	*Transcend Premium microSDHC* 16*GB Class* 10 *UHS-I*	1	130,00
Датчик температури	*DS18B20*	2	40,00
Датчик температури та атмосферного тиску	*BPM180*	1	60.00
Датчик освітленості	*TSL2561*	1	60,00

Отже, згідно таблиці 2.4, загальна ціна апаратної частини складає 5 400 гривень. Але, з точки зору практики, можливий такий варіант, що для налаштування міні-ПК *Raspberry Pi* вже є у наявності периферійні пристрої. Тому загальна вартість системи може значно скоротитись. Таким чином, якщо не брати до уваги периферійні пристрої, вартість системи становить 1910 гривень [16-20].

2.5. Загальний вигляд системи

Так як міні-ПК *Raspberry Pi* не має вбудованої внутрішньої пам’яті, потрібно попередньо підготувати її. Для цього було обрано карту пам’яті формату *MicroSD Transcend Premium microSDHC* 16*GB Class* 10 *UHS-I*. Вона має у своєму складі 16 гб постійної пам’яті, чого вистачить для встановлення системи та подальших маніпуляцій. Щоб встановити карту потрібно посунути її у слот *MicroSD* (рис. 2.3).

Рис. 2.3. Підключення карти пам’яті до міні-ПК *Raspberry Pi*

Далі потрібно підключити засоби введення та виведення. Для початку підключимо клавіатуру та маніпулятор типу «мишь». Для таких задач *Raspberry Pi* має у своєму складі *usb* порти, у які можна підключити периферію.

Наступним кроком є підключення монітору. Для цього потрібно з'єднати *Raspberry Pi* та монітор *HDMI* кабелем та підключити відповідно у *HDMI* порти.

Підключення міні-ПК *Raspberry Pi* до джерела живлення є останнім кроком у процесі апаратного налаштування, і це потрібно робити лише тоді, коли користувач готовий налаштувати програмне забезпечення: міні-ПК *Raspberry Pi* не має вимикача живлення і ввімкнеться як тільки він підключений до джерела живлення під напругою.

У загальному вигляді маємо пристрій, до якого підключені периферія по кабелю *USB* (білий колір кабелю), монітор по кабелю *HDMI* (червоний колір кабелю), а також адаптер живлення через *micro USB* (чорний колір кабелю) (рис.2.4).

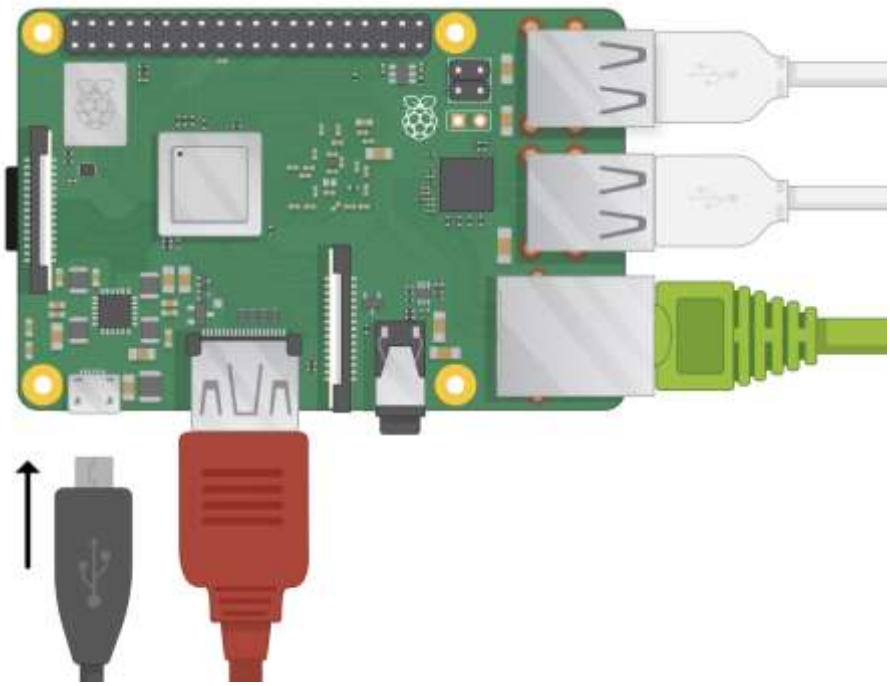

Рис. 2.4. Загальний вигляд міні-ПК *Raspberry Pi* з підключеними пристроями

У результаті повна схема системи на основі міні-ПК *Raspberry Pi* відображена на рисунку 2.5. до міні-ПК *Raspberry Pi* також мають здатність підключатися необхідні для роботи системи датчики, а також є можливість підключатися до мережі інтернет [21].

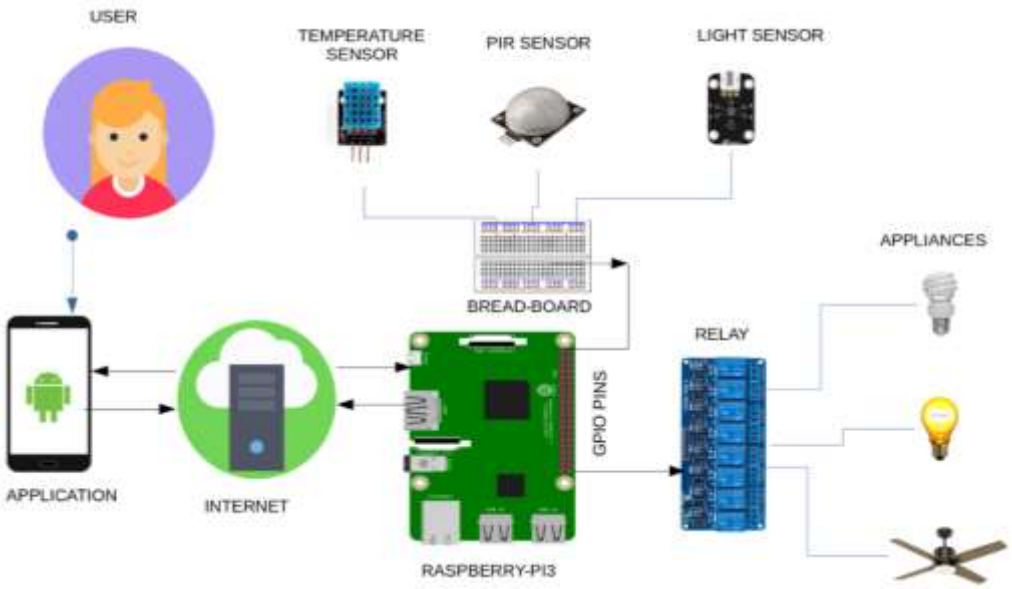

Рис. 2.5. Загальна схема системи *IoT* на основі міні-ПК *Raspberry Pi*

Висновок до розділу

У даному розділі було проаналізовано апаратне забезпечення. Для забезпечення функціонування системи управління та моніторингу потрібно вибрати таке апаратне забезпечення, яке забезпечуватиме виконання поставлених задач та цілей. Тому було запропоновано міні-ПК *Pasbperry Pi*, як серце системи. Він здатний забезпечувати як і прості практичні завдання, наприклад вивчення комп'ютера і основ роботи з ним, інтернет-серфінгу, а також для програвання відео- та прослуховування аудіофайлів, так і більш складні задачі, наприклад бути частиною *IoT* системи. Також варто зазначити, що міні-ПК *Pasbperry Pi* є економним варіантом для побудови системи управління та моніторингу елементами *IoT*. А саме такі цілі и ставились у даному дослідженні.

Були наведені технічні характеристики і можливості міні-ПК *Pasbperry Pi* та додаткового обладнання, необхідного для налаштування системи управління та моніторингу елементами *IoT*.

Також були підсумкові економічні показники обладнання.

РОЗДІЛ 3

НАЛАШТУВАННЯ ЗАСОБІВ МОНІТОРИНГУ ТА УПРАВЛІННЯ

3.1. Налаштування програмного забезпечення

Перш ніж розпочати використання міні-ПК *Raspberry Pi*, потрібно налаштувати його програмне забезпечення, зокрема його операційну систему, програмне забезпечення, яке контролює, що може робити *Pi*. *NOOBS*, нове готове програмне забезпечення, покликане максимально спростити це, дозволяючи вибрати один із декількох різних операційних систем і встановити їх автоматично. *NOOBS* можна скачати з сайту *Raspberry* та встановити на карту пам’яті.

Завантажений образ операційної системи необхідно записати на карту пам'яті *microSD*. Це можна зробити за допомогою утиліти *Win*32*DiskImager*.exe. Запускаємо *Win*32*DiskImager*, в поле *Device* потрібно вибрати карту пам'яті, на яку здійснюється запис образу, і клікнувши на піктограму з синьою папкою вказати шлях до образу ОС. Після цього натискаємо кнопку *Write* (рис. 3.1).

Рис. 3.1. Вікно запису програми *Win*32*DiskImager*

Коли міні-ПК *Raspberry Pi* вперше увімкнеться або завантажиться зі свіжою інсталяцією *NOOBS* на своїй картці *microSD*, з’явиться екран з логотипом міні-ПК

Raspberry Pi і невелике вікно прогресу у верхньому лівому куті. Після однохвилинної паузи з’явиться відповідне вікно на рис. 3.2.

Рис. 3.2. Меню системи *NOOBS*

Меню *NOOBS* – система, яка дозволяє вибрати операційну систему для запуску на міні-ПК *Raspberry Pi*. У стандартну комплектацію *NOOBS* входять дві операційні системи: *Raspbian*, версія операційної системи *Debian Linux*, спеціально розроблена для міні-ПК *Raspberry Pi* та *LibreELEC*, версія програмного забезпечення *Kodi Entertainment Center*. Якщо міні-ПК *Raspberry Pi* підключений до мережі - або за допомогою дротового з’єднання, або за допомогою опції „Мережі *Wi-Fi* (*w*)“ у верхній панелі піктограм - також можливо завантажити та встановити інші операційні системи. У нашому випадку буде обрано *Raspbian*.

Після вибору операційної системи з’явиться вікно прогресу інсталяції ОС (рис. 3.3).

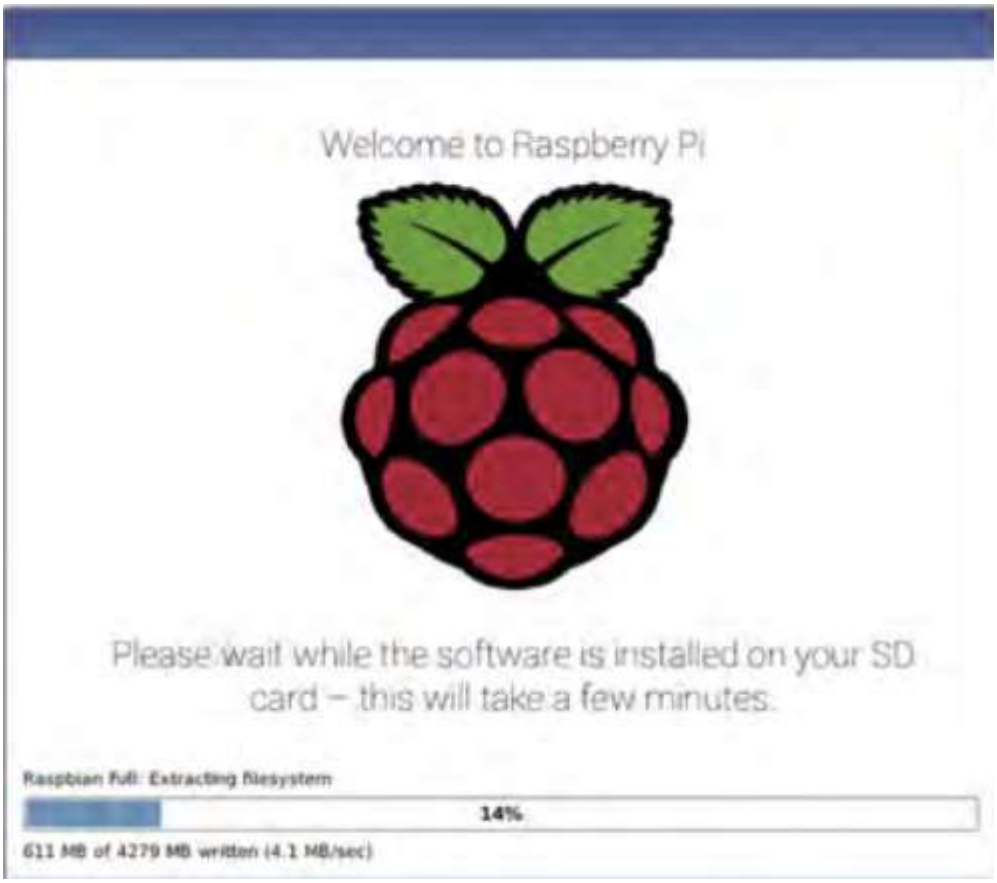

Рис. 3.3. Встановлення операційної системи *Raspbian*

Після встановлення потрібно перезавантажити міні-ПК *Raspberry Pi*. Далі систему буде встановлено.

Після включення користувач одразу побачить робочий стіл та майстер налаштувань операційної системи (рис. 3.4) [21, 22].

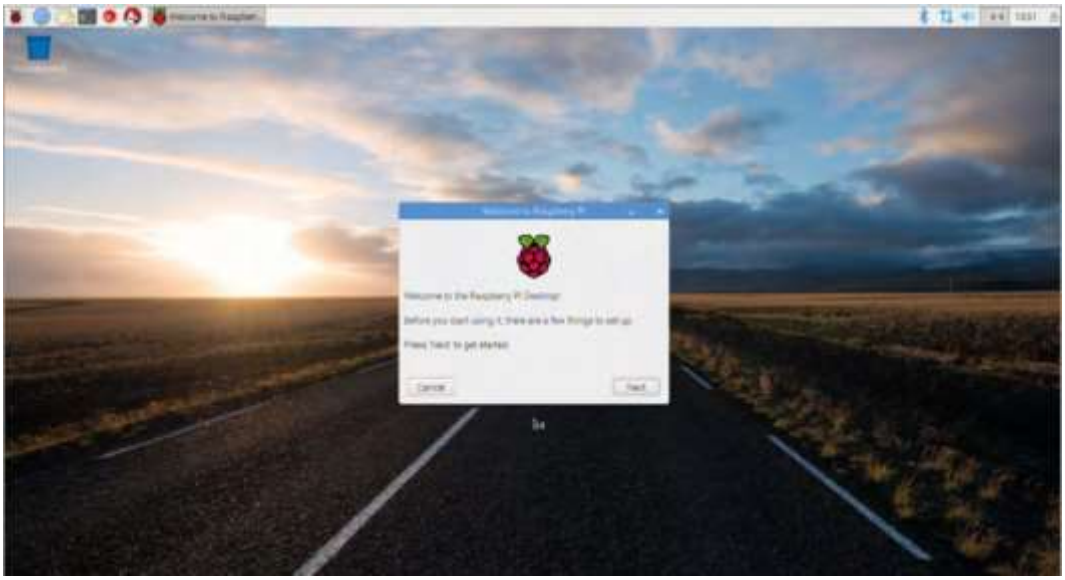

Рис. 3.4. Майстер налаштувань та робочий стіл

3.2. Налаштування мережевих параметрів

Швидше за все, міні-ПК *Raspberry Pi* не використовуватиметься для виведення інформації на монітор - вам напевно знадобиться віддалений доступ до нього по *ssh*, тому зараз слід налаштувати його параметри мережі.

Установку мережевого з'єднання для міні-ПК *Raspberry Pi* забезпечує файл конфігурації / *etc* / *network* / *interfaces*, і якщо *I*P-адреса міні-ПК *Raspberry Pi* отримує по *DHCP*, то нічого правити в цьому файлі вам не доведеться. У разі ж призначення міні-ПК *Raspberry Pi* статичного *I*P-адреси, виконуємо команду:

```
sudo nano /etc/network/interfaces
```

і дописуємо в файл конфігурації / *etc* / *network* / *interfaces* наступні рядки:

```
iface eth0 inet static
address 192.168.0.117
netmask 255.255.255.0
gateway 192.168.0.28
auto eth0
```

де:

*iface eth*0 *inet static* - вказує системі, що інтерфейс (*iface eth*0) знаходиться в діапазоні адрес *IPv*4 (*inet*) зі статичним *I*P-адресою (*static*);

– *address* 192.168.0.117- вказує, що *I*P-адреса (*address*) нашої мережевої карти 192.168.0.117;

– *netmask* 255.255.255.0 - вказує, що наша маска підмережі (*netmask*) має значення 255.255.255.0;

– *gateway* 192.168.0.28 - вказує, що адреса шлюзу (*gateway*) за замовчуванням 192.168.0.1;

– *auto eth*0 - вказує, що інтерфейс *eth*0 необхідно включати автоматично при завантаженні системи з зазначеними тут параметрами.

Якщо потрібно змінити адреси серверів *DNS*, виконуємо команду:

```
sudo gedit /etc/resolv.conf
```

і вписуємо в розпочатому файлі конфігурації *resolv.conf* наступні рядки:

```
nameserver 192.168.1.1
nameserver 8.8.8.8
```

де: 192.168.1.1 і 8.8.8.8 - адреси серверів *DNS*. Якщо потрібно додати більше адрес - кожну адресу слід починати з нового рядка і з слова *nameserver* [14].

3.3. Налаштування доступу по *Wi-Fi*

Далеко не завжди зручно вести до міні-ПК *Raspberry Pi* мережевий провід. Вирішити цю проблему можна за допомогою адаптера *Wi-Fi*, що підключається через *USB*. У попередньому розділі було зазначено, що цим адаптером є *EDUP Nano EP N*8531.

Далі підключаємо адаптер до *USB* – порту та перевіряємо, чи визначить його система. Для цього виконуємо команду *iwconfig* та бачимо, що адаптер знайдено (рис. 3.5).

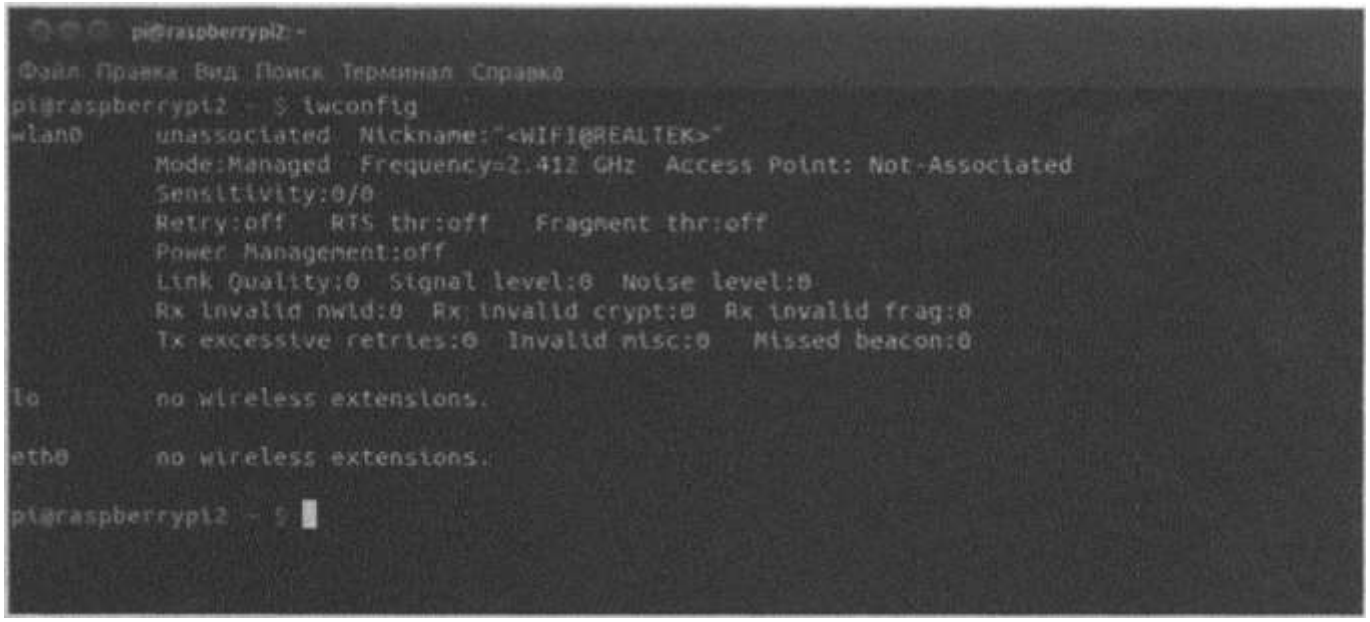

Рис. 3.4. Результат виконання команди *iwconfig*

Тепер скануємо простір на пошук доступних бездротових мереж командою:

```
sudo iwlist wlan0 scan
```

і бачимо, що знайдена точка доступу *AndroidAP* (рис. 3.5) [14].

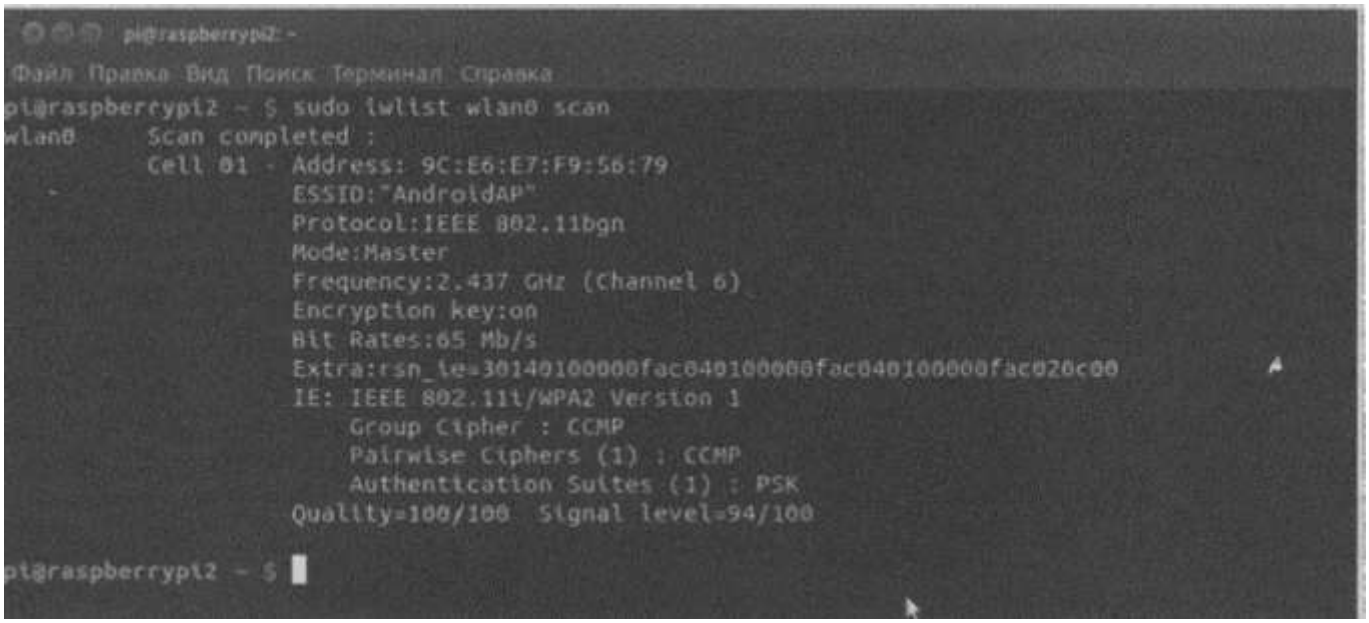

Рис. 3.5. Пошук доступних бездротових мереж

3.4. Налаштуванная *ssh* для підключення міні-ПК *Raspberry Pi*

SSH - це протокол, який забезпечує з'єднання по мережі (локальної або інтернет) між двома віддаленими пристроями. По ньому можна передавати майже будь-яку інформацію:

- файли;
- відео та аудіо в потоці;
- команди.

У нього є два значимих переваги. По-перше, він вміє на льоту стискати дані. По-друге, він створює зашифрований тунель.

Перше, що передбачає на міні-ПК *Raspberry Pi SSH* налаштування - запуск міні-ПК *Raspberry Pi* і відкриття терміналу. У консолі потрібно виконати команду *raspi-config*. Через мить після натиснення на *Enter* відобразиться інтерфейс стандартної утиліти конфігурації *Raspberry*. У ній необхідно знайти пункт *Interfacing Option* (рис. 3.6).

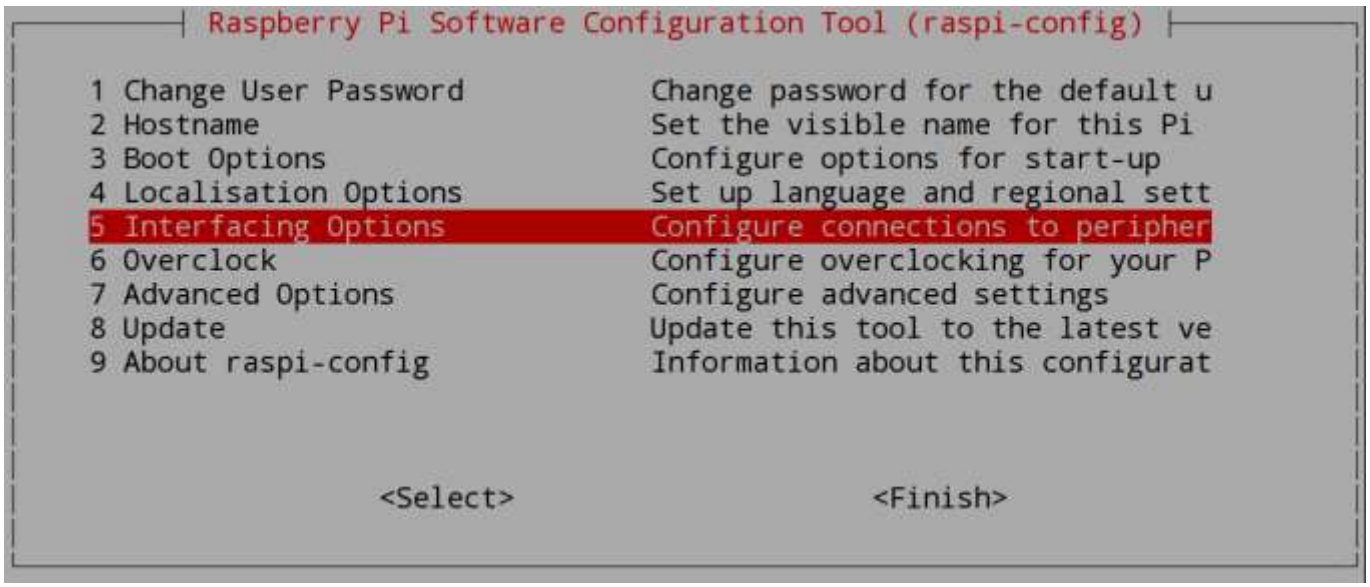

Рис. 3.6. Утиліта конфігурації *Raspberry*

Тепер потрібно зробити на міні-ПК *Raspberry Pi enable SSH*. Для цього слід виділити пункт *SSH*, натиснути *Enter* і вибрати *Enable*, а потім знову натиснути на "Введення". Потім залишиться повернутися на головний екран і вибрати *Finish*. Після цього всі внесені зміни будуть застосовані.

Тепер слід розглянути питання щодо того, як виконується підключення по *SSH* до *Raspberry Pi*. Розглянемо варіант з'єднання на платформі *Windows*.

Перше, що знадобиться - спеціальний додаток, так як в *Windows* немає своєї програми для роботи по протоколу *Secure Shell*. Найпопулярнішим з них є *PuTTY* - це повністю безкоштовний *SSH*-клієнт.

У вікні налаштування *PuTTY* (рис. 3.7) на вкладці "Сеанс" потрібно зробити наступне:

– вказати адресу *Raspberry* в поле "Ім'я хоста";

– ввести номер порту (для *SSH* найчастіше використовується 22-й, і він буде саме таким, якщо користувач сам нічого не змінив);

– в списку перемикачів "Тип з'єднання" повинен бути активний *SSH*.

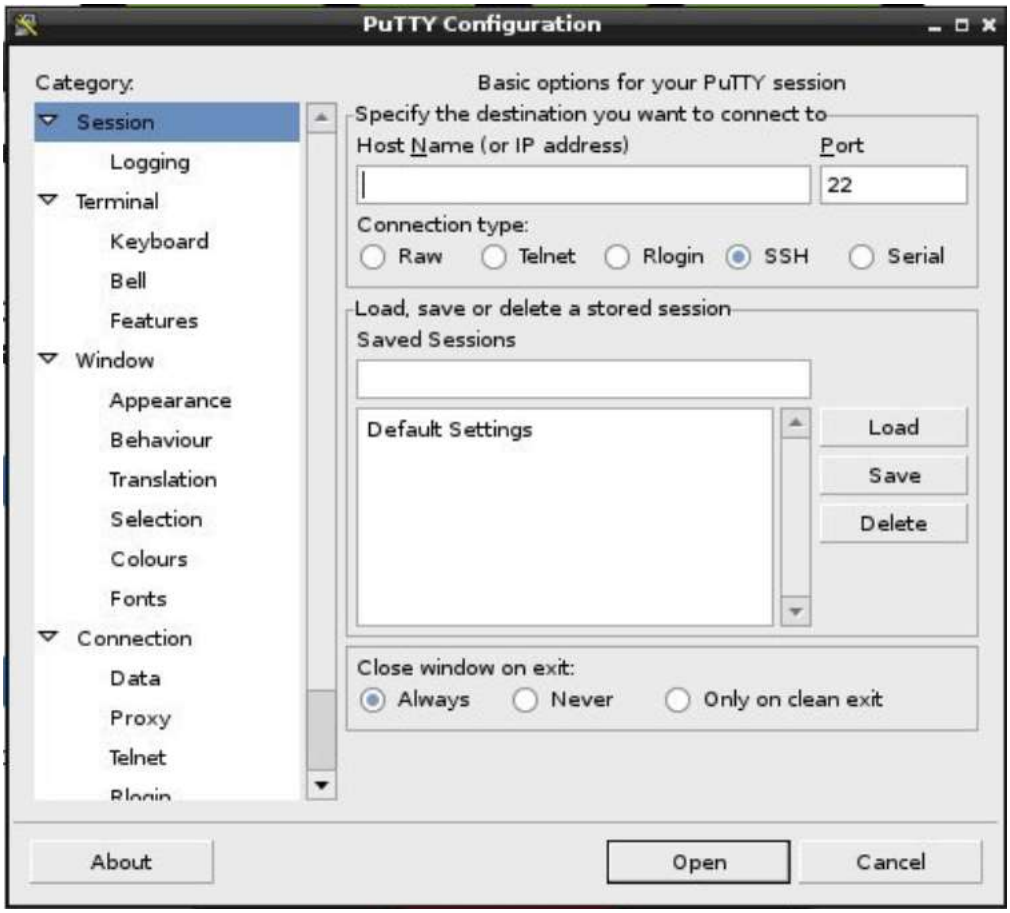

Рис. 3.7. Вікно налаштування *PuTTY*

Коли всі попередні дії зроблені, потрібно натиснути на кнопку «з'єднання». Якщо адреса, порт і тип з'єднання були вказані правильно, а мережа на двох пристроях працює справно, то через мить з'явитися запрошення командного рядка *Windows*, і через неї можна буде взаємодіяти з міні-ПК *Raspberry Pi* і стануть доступні всі команди.

Коли з'явиться консоль, потрібно пройти авторизацію в системі. Для цього потрібно, по-перше, вказати логін (ім'я користувача), а, по-друге, пароль, який призначений для відповідного користувача. Якщо вони коректні, то буде дано повний доступ до комп'ютера [23].

3.5. Встановлення та конфігурація *Domoticz*

Domoticz - це дуже легка система домашньої автоматизації, яка дозволяє вам контролювати і настроювати різні пристрої, включаючи світло, вимикачі, різні

датчики / лічильники, такі як температура, опади, вітер, ультрафіолетове (УФ) випромінювання, споживання електроенергії / виробництво, споживання газу, споживання води і багато іншого. Повідомлення / оповіщення можуть бути відправлені на будь-який мобільний пристрій, або електронну пошту.

Система *Domoticz* встановлюється на міні-ПК *Raspberry Pi*, *Windows*, *Linux*, *Mac OS X* і вбудованих пристроях. Система призначена для роботи в різних операційних системах. Має власний додаток для смартфона. Інтерфейс являє собою масштабований веб-інтерфейс *HTML5* і автоматично адаптується для настільних і мобільних пристроїв. Сумісний з усіма браузерами.

Для встановлення *Domoticz* потрібно в терміналі системи, на яку *Domoticz* встановлюється, в даному випадку *Raspbian*, ввести команду:

```
sudo curl -L install.domoticz.com | sudo bash
```

Після введення команди відобразиться екран як на рис. 3.8:

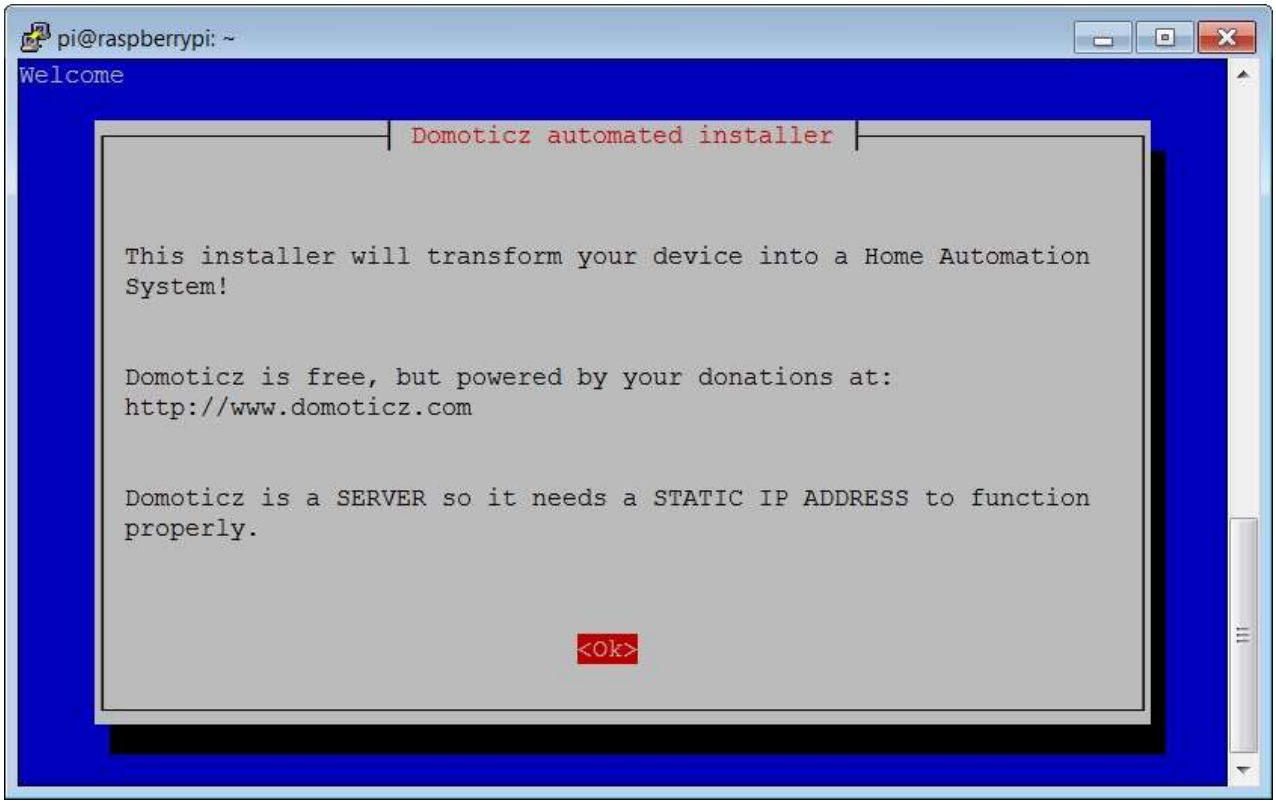

Рис. 3.8. Установка системи *Domoticz*

Після завантаження необхідних компонентів відобразиться допомогу по установці *Domoticz*, де краще залишити запропоновані значення за замовчуванням або вказати:

– протокол *HTTP* / *HTTPS*;

– порт для системи (8080 за замовчуванням);

– шлях для *Domoticz* (за замовчуванням / *home* / *pi* / *domoticz*).

Вхід на міні-ПК *Raspberry Pi* після установки здійснюється через рядок браузера.

http://IP.адрес.Raspberry:8080

Далі з'явиться вікно *Domoticz* (рис. 3.9). На цьому установка закінчена. *Domoticz* додасться в автозапуск при інсталяції. При подальшій перезавантаженні система запуститься автоматично [24, 25].

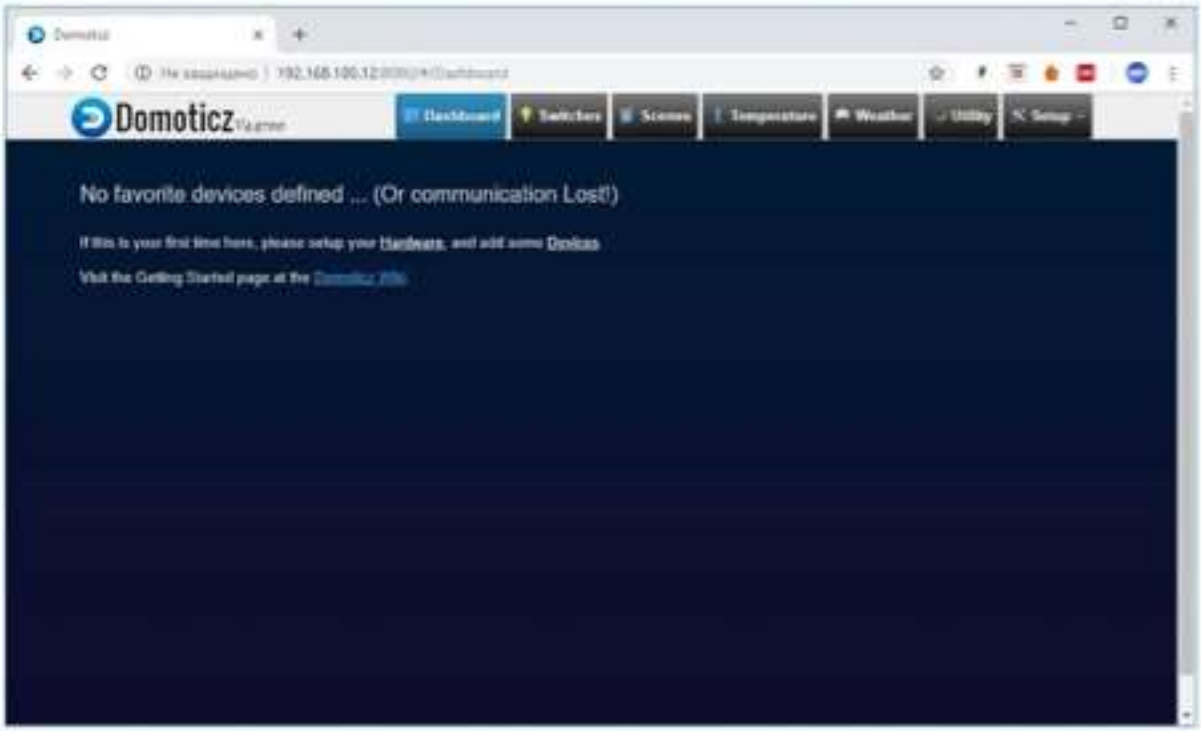

Рис. 3.9. Домашня сторінка *Domoticz*

3.6. Підключення датчиків у системи *Domoticz*

Після налаштування системи *Domoticz* потрібно підключити датчики у систему. Для цього було підібрано два датчики температури *DS*18*B*20, які підключаються через інтерфейс 1-*ware*, датчик температури та атмосферного тиску *BPM*180 та датчик освітленості *TSL*2561, який підключається за допомогою інтерфейсу I^2C.

Підключення відбувається за допомогою веб-інтерфейсу. Для цього потрібно перейти на вкладку *setup* та обрати *hardware*. У вікні, шо відкриється відображається список пристроїв, та вікно додання нових (рис. 3.10).

При доданні нового пристрою, наприклад датчика температури *DS*18*B*20, потрібно прописати його ім'я – *DS*1820 #1, тип інтерфейсу – «1-*ware* (*System*)». Інші налаштування залишаємо за замовчуванням. Для збереження змін та додання пристрою слід натиснути *add*. Аналогічно конфігуруються і підключається другий датчик.

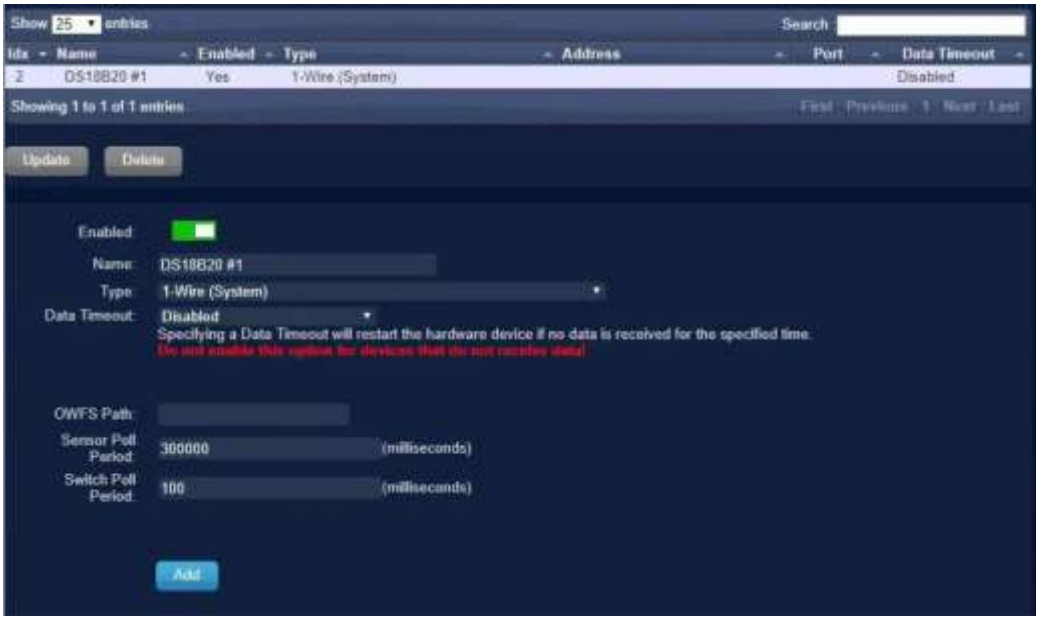

Рис. 3.10. Список підключених пристроїв

Далі конфігуруємо датчик температури та атмосферного тиску *BPM*180. У цьому випадку у пункті *Type* потрібно обрати інтерфейс I^2C *sensors* і у вкладці *SubType* – I^2C *sensors BPM*180/180 *Temp+Baro*. Таким же чином додається датчик освітлення *TSL*2561, але для нього у полі *SubType* потрібно обрати I^2C *sensor TSL*2561 *Illuminance* (рис. 3.10).

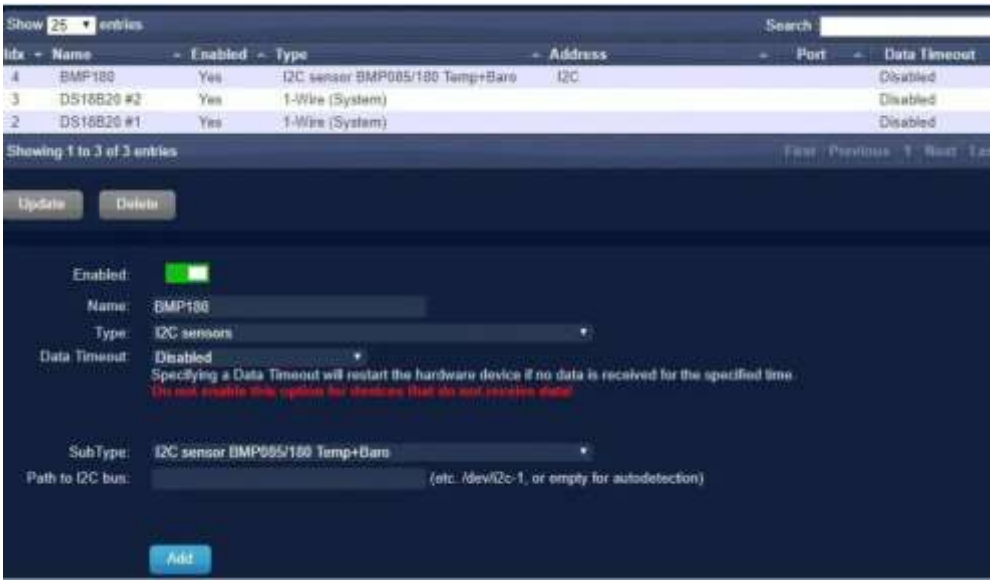

Рис. 3.10. Додавання датчика атмосферного тиску та температури *BPM*180

Датчики підключені, але на цьому не кінець. Вони знаходяться у неактивному стані. Для їх активації потрібно перейти у меню *setup*, обрати пункт *setting* та активувати датчики. В системі відбувається угруповання датчиків у відповідні категорії. Наприклад, датчики температури переходять у розділ *temperature sensors*. Також тут буде відображатись датчик температури та атмосферного тиску, але у даній категорії відобразить лише температуру, а тиск у іншому розділі – *weather sensors*. Датчик освітлення групується у розділ *utility sensors*. Також у системі є здатність вибирати датчики, які, наприклад, найчастіше використовуються у розділ «обране». Після додавання в «обране» пристрій з'явиться на дашборді (рис. 3.11).

Рис. 3.11. Панель *Dashboard* з «обраними» датчиками

Система *Domoticz* здатна також показувати динаміку зміни показань різних датчиків. Так, наприклад, на рисунку 3.12 показано, як змінювалась температура за певний проміжок часу [26].

Рис. 3.12. Динаміка зміни температури у датчика *DS18B20*

3.7. *Domoticz API*

Domoticz дозволяє взаємодіяти з усіма вашими перемикачами та датчиками за допомогою *JSON*, або інтерактивно через браузер, або програмно мовою сценаріїв. Насправді, коли користувач взаємодіє з інтерфейсом *Domoticz* через браузер, керування пристроями відбувається за допомогою *JSON*.

Формат запитів виглядає наступним чином:

`http://<username:password@>domoticz-ip<:port>/json.htm?api-call`

`<username:password@>` = ім'я користувача та пароль для доступу до *Domoticz*.

`domoticz-ip` = *IP*-адреса або ім'я хосту вашої інсталяції *Domoticz*.

`<:port>` = номер порту вашої інсталяції *Domoticz*.

Даний формат дає змогу виконувати всі методи, що у системі. Від показу статусу пристрою та ввімкнення пристрою до створення власних сценаріїв. Прикладом простого сценарію ввімкнення лампи є:

`/json.htm?type=command&param=switchlight&idx=99&switchcmd=On`

idx = ідентифікатор вашого пристрою (у цьому прикладі 99).

switchcmd = "Увімкнено" або "Вимкнено" [27].

У разі успішного виконання, у відповідь отримуємо наступний об’єкт:

```
{
   "status"  :  "OK" ,
   "title"  :  "SwitchLight"
}
```

Так як в даному дослідженні у систему додані температурні датчики, датчик атмосфернго тиску та температури, освітленості, наведемо приклади сценарію оновлення цих пристроїв.

Датчик температури:

/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP,

де

– *IDX* = *id* вашого пристрою;

– *TEMP* = Температура.

Датчик атмосферного тиску та температури:

/json.htm?type=command&param=udevice&idx=IDX&nvalue=0&svalue=TEMP;BAR;BAR_FOR;ALTITUDE,

де

– *IDX* = *id* пристрої;

– *TEMP* = Температура;

– *BAR* = Барометричне тиск;

– *BAR_FOR* = Прогноз барометра;

– *ALTITUDE* = В даний час не використовується, може бути 0.

Прогноз барометра може бути одним з:

0 = немає інформації;

1 = сонячний;

2 = похмуро;

3 = хмарно;

4 = дощ.

Датчик освітленості:

/json.htm?type=command&param=udevice&idx=IDX&svalue=VALUE,

де

IDX = ідентифікатор пристроїв;

VALUE = значення світності в *lux* [27].

Висновки до розділу

У даному розділі було проведено налаштування засобів моніторингу. А саме завантажено та встановлено операційну систему на міні-ПК *Pasbperry Pi* для його подальшої роботи. Також налаштовано мережеві параметри міні-ПК *Pasbperry Pi* та віддалений доступ по *ssh*.

Ще одним важливим кроком у даному розділі є встановлення та конфігурація *Domoticz*. *Domoticz* - це дуже легка система домашньої автоматизації, яка дозволяє вам контролювати і настроювати різні пристрої, включаючи світло, вимикачі, різні датчики / лічильники, такі як температура, опади, вітер, ультрафіолетове (УФ) випромінювання, споживання електроенергії / виробництво, споживання газу, споживання води і багато іншого.

Варто нагадати, що дане дослідження робиться з метою виявлення мінімально необхідних засобів та кроків для побудови своєї власної системи управління та моніторингу елементами *IoT*. Тому наведені вище кроки також є важливою частиною проведення дослідження.

РОЗДІЛ 4
ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Архітектура та інструменти розробки

У результаті аналізу архітектури мережі *IoT*, було виявлено, що для подальшої розробки найбільше підходить архітектура клієнт-сервер. Клієнт розуміється як додаток, який встановлений на користувацькій машині, а сервер у свою чергу може знаходитись на віддаленій машині, або на хостингу. Така архітектура дає змогу серверу бути автономним і не залежати від клієнта. Клієнт у свою чергу має легкий спосіб підключення до серверу та незалежність від інших клієнтів.

Виходячи з вибраної архітектури для додатку, який забезпечує керування системою, обрано мову *Java* та фреймоворк *Spring Boot*. Однією з причин обрання 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*.

Додаткові 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» [28, 29]. 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*) [28, 29].

За зберігання даних буде відповідальна СУБД (система управління 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. *PostgreSQL* та *Hibernate*. *PostgreSQL* - це реаляційна, об'єктна система управління 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 (Atomicity, Consistency, Isolation, Durability).*

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 СУБД, 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.

Веб-інтерфес розроблений з допомогою шаблонізатору *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.

4.1.1. Архітектура серверного додатку

Пропоновано додаток за архітектурним шаблоном *MVC* (*model, view, controller*). Така архітектура дозволяє слабо зв’язувати між собою модулі, що дає можливість легкої підтримки та заміни модулів [30, 31].

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 [32].

4.1.2. Архітектура клієнтського додатку

У клієнтському додатку пропонується використовувати монолітну архітектуру. Так як даному додатку немає необхідності мати у своєму складі бази даних та веб інтерфейс було обрано дану архітектуру. Основною задачею додатку є з'єднання з віддаленим сервером, підтримка цього з'єднання та відправка і отримання повідомлень. Тому використання мікросервісної або *MVC* архітектур є надмірним.

4.2. Система автоматизації збору даних

Основною ідеєю додатку є збір даних з декількох серверів, на яких встановлений *Domoticz*, їх систематизація в одному місці, збір статистики та можливий контроль.

У системи *Domoticz* є *API*, яке дозволяє через запити у форматі *JSON*, запитувати певні дані, або надсилати команди. Для відправки запитів буде використовуватись *RestTemplate*. Це вбудований засіб *Spring Boot*. Він дозволяє надсилати дані без явних перетворень. Клас *HttpUtil* являє собою сервіс для відправки запитів. Клас наслідується від абстрактного класу *ClosableHttpUtil* (рис. 4.1).

```
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);
        }
```

Рис. 4.1. Приклад методу для надсилання запитів

4.2.1. Інтерфейс користувача

Користувацький інтерфейс пропонується розробити з допомогою шаблонізатора *Freemarker*. Великою перевагою якого є макроси. Макроси дозволяють використовувати повторно вже написаний код, змінюючи невеликі його частини.

За допомогою анотації *@Controller* або *@RestController* відбувається оголошення контроллера (рис 4.2). Шлях, який буде обробляти контроллер, задається за допомогою анотації *@RequestMapping*(*path*). 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. 4.2. Приклад оголошення шляху, який обробляє *HTTP* метод *GET*

4.3. Налаштування системи

Конфігурація доступу до бази даних проводиться з допомогою файлу конфігурації *db.properties* (рис 4.3). Оскільки коонфігурація проаодиться під час запуску додатку, після зміни конфігураціх потрібно його перезапустити.

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. 4.3. Приклад конфігурації доступу до бази даних

Для того, щоб мати змогу налаштувати клієнтський застосунок, потрібно у файл конфігурації *application.properties* встановити такі налаштування:

1. *remote-server.ip* - IP-адреса віддаленого сервера. На нього будуть відправлятись дані.

2. *remote-server.port* - необов'язковий параметр. Його 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, Його 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.

4.4. Демонстрація роботи

Для встановлення системи потрібно щоб на машині була встановлена *JVM*, не нижче 8-мої версії. Далі 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*". Для запуску серверного додатку

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*"

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 4.4).

Рис. 4.4. Форма входу в систему

Якщо це перший запуск, потрібно увійти під стандартними ініціалами "*admin*:*admin*". Після цього необхідно провести зміну паролю, а також додати нових користувачів, якщо це необхідно. Для керування користувачами в системі додана відповідна панель (рис 4.5). На панелі можна редагувати інформацію та права користувачів, а також додавати нових.

Рис. 4.5. Панель адміністратора

Для підключення додатків-клієнтів до системи, потрібно спочатку його створити на панелі створення клієнтів (рис 4.6). Де потрібно ввести назву клієнта, що буде використовуватись в системі як ідентифікатор, та згенерувати ключ безпеки.

Рис 4.6. 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. 4.7.). 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. 4.7. 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.

На вкладці «Домашня» є можливість перегляду пристроїв. Тут відображені всі підключені на даний момент пристрої (рис. 4.8). Також є можливість сортувати пристрої за клієнтом або типом пристрою. Наприклад датчик температури, лампочка тощо.

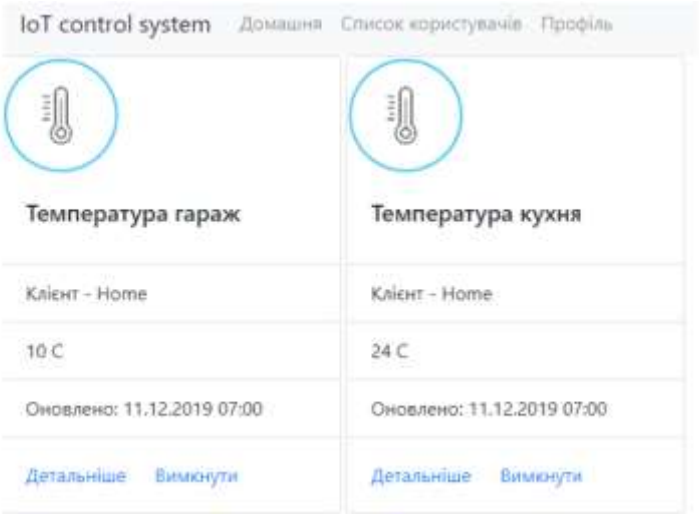

Рис. 4.8. Відображення на домашній сторінці датчиків температури

Також з домашньої сторінки є можливість перейти в детальний огляд пристроїв. Там можна вимкнути пристрій, але вимкнуті пристрої не відображаються на домашній сторінці. 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. Якщо це, наприклад, датчик температури, також можна побачити графік коливань температури за останню добу (рис. 4.9).

Рис. 4.9. Детальна інформація про пристрій та графік зміни температури

4.5. Кібербезпека *IoT*

Важливою частиною *IoT* є кібербезпека. Для моніторингу *IoT* деякі держави запроваджують спеціальні норми та правила використання таких пристроїв.

IoT уможливлює знайти будь-яку людину, спрощуючи наше життя; однак без належної впевненості в безпеці і приватності даних користувача ця система багатьма не буде прийнята. Тому для повсюдного впровадження *IoT* повинен мати сильну захисну інфраструктуру [2].

Як кібер-фізична структура інтелектуальна мережа, заснована на Інтернеті Речей, стикається з декількома проблемами:

– підміна особистості (імперсонації) - вид атаки, націленої на несанкціоновану зв'язок від імені легітимного пристрої з використанням його ідентифікаційних даних. Зловмисник може підмінити ідентифікаційні дані одного з розумних лічильників, щоб маніпулювати даними про витрату енергії;

– несанкціоноване вилучення інформації. Оскільки об'єкти (пристрої) в інтелектуальній мережі, заснованої на Інтернеті Речей, зв'язуються, часто використовуючи загальну інфраструктуру зв'язку, зловмисник може легко отримати доступ до переданих даних;

– втручання в дані - зловмисник може змінити дані обміну, такі, як динамічні значення цін, відправлені в попередні пікові періоди (роблячи їх набагато менше). Наслідками цього можуть бути збільшення витрат домоволодінь замість їх зменшення, таким чином перенагружая енергосистему;

– проблеми авторизації і контролю доступу. З тих пір, як деякі пристрої можуть контролюватися і налаштовуватися віддалено (наприклад, розумні лічильники або територіально розміщені датчики і приводи в розподільних підстанціях), зловмисник або навіть незадоволений працівник може спробувати несанкціоновано отримати права доступу, щоб використовувати їх, пошкоджуючи фізичні об'єкти (такі, як трансформатори) або приводячи до відключень електроенергії;

– проблеми приватності. Розумні лічильники і розумні прилади в житлових будинках можуть повідомляти не тільки витрата енергії. Їх згенерувала дрібномодульних інформація може порушити приватність кінцевого споживача за рахунок розголошення інформації про його звички (час підйому, сну або обіду і т.д.), про те, чи знаходиться він вдома або ж у відпустці тощо;

– компрометації і шкідливий код. Комунікаційні та обчислювальні можливості об'єктів інтелектуальних мереж є цілями для фізичного або віддаленого компрометації. Більш того, з тих пір як вони використовують різні типи програмного забезпечення, вони можуть бути метою різного виду зараження програмного забезпечення або зараження шкідливим кодом з метою контролю і маніпуляції ними. Також широко розгорнуті об'єкти з обмеженими пристроями (датчики і т.д.) зазвичай нестійкі до втручань, що робить фізичну компрометацію легким завданням;

– доступність і проблеми з *DoS*. У класичних енергомережах було важко, якщо не неможливо, вибрати метою атаки доступність активів (електричних лічильників, підстанцій і т.д.), особливо в великому масштабі. В інтелектуальних мережах, в яких кошти ІКТ інтегровані навіть в життєво важливі активи енергомережі, дозволяють вибрати їх як цілі, роблячи їх частково або повністю недоступними в результаті *DoS*-атаки. Більш того, беручи до уваги те, що більшість пристроїв (речей) підтримують протокол *IP* і не використовують власних протоколів, завдання зловмисника значно полегшується;

– кібератаки. Інтелектуальна мережа може бути розглянута як найбільша кіберфізіческая система, що включає фізичні системи подання фізичних активів інтелектуальної мережі (трансформатори, вимикачі, розумні лічильники, кабелі тощо) та ІКТ-системи, де ІКТ-елементи контролюють або управляють фізичними об'єктами [6].

Несанкціонований доступ до *RFID*. Несанкціонований доступ до міток, які можуть містити ідентифікаційну інформацію, – це головна проблема безпеки *IoT* (можливо розкриття будь-якої конфіденційної інформації про користувача), по-цьому вона повинна бути вирішена в першу чергу. Мітка може бути не тільки прочитана зчитувальних пристроїв зловмисника, але навіть модифікована або пошкоджена.

Існують кілька реальних загроз для *RFID*, які включають *RFID*-вірус, атаку за допомогою мобільного телефону і злом *SpeedPass*.

Порушення безпеки вузлів датчиків. *WSN* вразлива до деяких видів атак, по вузлам датчиків - це частина двосторонньої мережі датчиків, що означає не тільки можливість передачі, але і захоплення даних. Можливі атаки містять у собі забивання каналу, втручання, атаку Сібілли (мережева атака, при якій один з вузлів може мати кілька ідентифікаторів, тим самим порушуючи роботу системи), заповнення та деякі інші види атак, які полягають в наступному:

1) забивання ускладнює роботу всієї мережі, інтерферуючи з частотами вузлів датчиків;

2) втручання – це вид атаки, в якій інформація з вузлів може бути вилучена або змінена зловмисником для того, щоб взяти вузол під свій контроль;

3) атака Сібілли – нав'язування множинних псевдоанонімність ідентіфікаційних даних, надаючи їм велике значення;

4) заповнення – це вид *DOS*-атаки, викликаний більший обсяг трафіку, що призводить до витрачання ресурсів пам'яті.

Зловживання хмарними обчисленнями. Хмарні обчислення – це велика мережа об'єднаних серверів, яка дозволяє ділитися ресурсами один з одним. Розділення ресурсів може зіткнутися з безліччю загроз безпеки, таких як *Man-in-the-middle* атака (*MITM*), фішинг і т.д. Альянс безпеки хмарних обчислень (*CSA*) покладав ще деякі загрози, такі як шкідливий інсайдер, втрати даних, крадіжка акаунтів, неймовірне використання комп'ютерів, які поділяють ресурси, які полягають в наступному:

1) шкідливий інсайдер – це загроза того, що легальний користувач, який має доступ до даних, може бути залучений в маніпуляції з даними;

2) втрата даних – це загроза, суть якої полягає в тому, що будь-який злочинець, що має несанкціонований доступ до мережі, може змінювати або видаляти існуючі дані;

3) *Man-in-the-middle* - це вид загрози крадіжки акаунтів, суть якої в тому, що атакуючий може змінювати або перехоплювати повідомлення в обміні між двома учасниками;

4) хмарні обчислення можуть бути використані жорстоким чином, оскільки якщо нападник отримує можливість завантажити будь-яке шкідливе програмне забезпечення на сервер, використовуючи, наприклад, ботнет, це може дати нападнику контроль над багатьма пов'язаними пристроями [2].

Маючи справу з алгоритмами безпеки, протоколами та методами інтелектуальних мереж, заснованими на Інтернеті Речей, необхідно взяти до уваги наступні завдання:

– масштабованість. Інтелектуальна мережа може сягати на величезній території (кілька міст або вся країна) і включати величезна кількість розумних пристроїв і об'єктів. Це робить важким розробку масштабованих рішень безпеки, таких, як управління ключами і аутентифікація;

– мобільність. З мобільними пристроями (об'єктами), такими, як електрокари або польові технічні агенти, завжди буде необхідність аутентифікації і безпечної комунікації із змінним оточенням (розумні лічильники, що заряджають станції і т.д.);

– розгортання. При охопленні всієї країни інтелектуальної мережею об'єкти (пристрої) розміщуються на величезних територіях, працюють без контролю і можуть бути поміщені у віддалені місця без будь-якого захисту по периметру, роблячи їх легкодоступними. Рішення безпеки повинні відслідковувати будь-які спроби втручань;

– успадковані системи. Раніше розміщені пристрої та системи можуть мати незначну або відсутню зовсім підтримку безпеки з тих пір, як вони стали грунтуватися здебільшого на власних рішеннях (апаратного або програмного забезпечення), розміщуватися на ізольованих островах без зв'язку або через приватну інфраструктуру зв'язку. Інтегрування цих успадкованих систем в інтелектуальну мережу, засновану на Інтернеті речей - це нетривіальне завдання, тим більше що в більшості випадків немає можливості замінити їх новими системами або оновити їх для підтримки бажаних рішень безпеки;

– обмежені ресурси. Деякі пристроїв (об'єкти) інтелектуальної мережі, особливо масово поширені і використовуються, обмежені в ресурсах. Особлива увага

повинна бути приділена розробці рішень безпеки, щоб переконатися, що їх обмежені ресурси зможуть прийняти рішення;

– неоднорідність. Досягнення безпечної *end-to-end*-зв'язку для великої кількості видів ресурсів пристроїв (об'єктів), які використовуються інтелектуальної мережею (таких, як пам'ять, обчислювальна потужність, пропускна здатність, автономність і т.д.) і використовуваних ними протоколів і стеків зв'язку (для пристроїв, що не використовують *IP*-протокол) - це завдання, що вимагає більш частих адаптацій існуючих рішень або навіть використання шлюзів;

– сумісність, що розглядається як один із наслідків неоднорідності впроваджуваних протоколів і стеків зв'язку між пристроями (об'єктами) інтелектуальної мережі. Успадковані системи і пристрої (об'єкти), які не підтримують *TCP* / *IP*-стеки (такі, як, *Zigbee v*1, *HART*), не можуть зв'язуватися з системами, заснованими на *IP*, і пристроями (об'єктами) без використання шлюзів, роблячи *end-to -end*-безпечну зв'язок неможливою. Сумісність може бути розглянута між двома пристроями, що використовують однакові протоколи і стеки зв'язку, але різні можливості функції: одна - з повною під- Держко, інша - з частковою (такий, як *DTLS*);

– початкова завантаження - це проблема ефективного виконання початкового завантаження мільйонів пристроїв (об'єктів) інтелектуальної мережі з необхідними початковими ключе- вимі матеріалами (такими, як криптографічні ключі, криптографічні функції або алгоритми і параметри і т.д.);

– довірче управління. Об'єкти (пристрої) в інтелектуальній мережі можуть управлятися різними суб'єктами (кінцевий споживач для розумних приладів, оператор *SG* для розумних лічильників і датчиків і т.д.). Об'єкти (пристрої) не можуть зв'язуватися, якщо не встановлено мінімальний рівень довіри. У той час як об'єкти (пристрої), керовані одним підприємством, можуть легко встановити довірчі взаємини, побудова довіри між об'єктами (пристроями), що належать різним підприємствам, - це трудомістке завдання, особливо в великомасштабних мережах;

– затримка / обмеження за часом. Деякі частини *SG* повинні відповідати в реальному часі на події та повідомлення. Наприклад, електричні *SCADA*-системи

(*Supervisory Control and Data Acquisition*), що використовуються на передавальних і розподільних підстанціях, повинні відповідати в реальному часі на будь-які зміни значень струму, напруги або частоти на додаток до інших метеорологічним параметрам, що впливає на функціонування обладнання, представленими різними розумними об'єктами (датчиками, приводами і т.д.), для того щоб забезпечити безпеку і запобігти поширенню аномалій (наприклад, перевантаження по потужності або перебою) на інші частини енергомережі. Це робить операції, які витрачають багато часу (тобто операції із загальним ключем), непридатними для використання [6].

Так у США у 2019 році був переданий законопроект, згідно з яким влада отримує повний доступ до пристроїв інтернету речей. Законопроект передбачає, що розробники таких пристроїв будуть передавати владі всю інформацію про прошивку, схемою роботи, будь-яких ризики злому пристроїв, а також способи протидії можливому злому. Зроблені за межами США пристрої, чиї розробники відмовляться надати відповідну інформацію, будуть заборонені до використання[33]. Також Великобританія запровадила законопроект щодо пристроїв *IoT*. Але він направлений скоріше не на моніторинг зі сторони влади, а на створення умов безпеки для користувачів [34]. Все це говорить про те що тема моніторингу та управління розвиватиметься і надалі, тому що безпека у державі стоїть на перших місцях і здатність контролювати пристрої *IoT* залишатиметься у пріоритеті.

Висновки до розділу

У даному розділі у результаті аналізу архітектури мережі *IoT*, було виявлено, що для подальшої розробки найбільше підходить архітектура клієнт-сервер. Виходячи з вибраної архітектури для додатку, який забезпечує керування системою, обрано мову *Java* є простий, об’єктно-орієнтований синтаксис, безпечна та безвідмовна робота з пам’яттю, незалежність від платформи та архітектури, висока продуктивність.

Spring Boot Framework – фреймворк, який спрощує розробку додатків на мові *Java*. Він дозволяє легко створювати повноцінні, продуктивні *Spring*-додатки, про які можна сказати - що їх потрібно "просто запустити».

Також пропоновано серверний додаток за архітектурним шаблоном *MVC*, а клієнтський за монолітною архітектурою і запропоновані засоби за допомогою яких є можливість розробки користувацького інтерфейсу.

У підсумку розділу продемонстрована робота системи.

ВИСНОВКИ

Причиною обрання даної теми є стрімкий розвиток технологій у світі, багатомільярдні компанії об’єднуються в альянси задля проектування єдиних стандартів та протоколів роботи пристроїв інтернету речей, підприємства переходять від ручної праці до автоматизованої, у побуті з’являються пристрої здатні покращувати та полегшувати життя людей, навіть без їх втручання. Уся індустрія йде шляхом полегшення життя людей, розробки автоматизованих систем, де люди прийматимуть мінімальну участь. Це в свою чергу дозволить мінімізувати ризики та зменшити витрати. Але звичайно мають існувати способи для контролю цих пристроїв. Будь-яка електроніка може містити у собі програмні або апаратні помилки, котрі треба вирішувати у короткий термін. Також для зручності мають існувати інтерфейси керування та моніторингу пристроїв. Наприклад, у побуті для керування пристроями інтернету речей можна використовувати додаток на смартфон, але щоб це працювало, пристрої мають бути об’єднанні в одну мережу тощо.

Інтернет речей є наступним кроком на шляху до оцифрування сучасного суспільства, де предмети і люди пов'язані один з одним через комунікаційні мережі і з’являється можливість повідомляти про їх стан та стан навколишнього середовища. Розробники таких програм намагаються якомога більше удосконалювати свої роботи. Інтернет речей створює нові можливості та забезпечує конкурентні переваги для бізнесу як на поточних, так і на нових ринках.

У найближче десятиліття завдяки інтернету речей (*IoT*) очікується сплеск обсягу даних, вироблених підприємствами. Це надасть великий вплив на ефективність і конкурентоспроможність бізнесу. Додатки *IoT* відрізняються від звичайних одноадресних комунікацій, тому підприємствам необхідно відповісти на ряд важливих питань, щоб зберегти бізнес в епоху інформаційної революції.

Інтернет речей є наступним кроком на шляху до оцифрування сучасного суспільства, де предмети і люди пов'язані один з одним через комунікаційні мережі і з’являється можливість повідомляти про їх стан та стан навколишнього середовища. Розробники таких програм намагаються якомога більше удосконалювати свої роботи.

Інтернет речей створює нові можливості та забезпечує конкурентні переваги для бізнесу як на поточних, так і на нових ринках.

Моніторинг та контроль приладів є одним із важливих заходів, за яким слід пильно стежити та використовувати його в режимі реального часу для безпеки, безпеки та комфорту людей. З розвитком Інтернет-технологій та бездротових сенсорних мереж реалізується нова тенденція в епоху повсюдного поширення. Величезний приріст користувачів Інтернету та модифікації робочих технологій Інтернету дозволяють створювати мережі повсякденних об'єктів. Споживачі покладають великі надії на системи з підтримкою Інтернету. Їх переваги добре відомі. Зниження експлуатаційних витрат та витрат на обслуговування завдяки віддаленому моніторингу, діагностиці, налагодженню та модернізації мікропрограми. Зручність та безпека, що поставляються з можливістю контролювати стан розумного будинку та контролювати Інтернет-прилади, коли споживачів немає вдома.

Інтернет речей змінює спосіб розвитку промислового та споживчого ринку. Робото-технічні пристрої, дрони, автономні транспортні засоби, блокчейн, розширена та віртуальна реальність, цифрові помічники та машинне навчання (штучний інтелект) – це технології, які переводять на наступний етап розробки додатків IP.

У світі існує та ще довго існуватиме тренд на автоматизацію та віддалене керування і моніторинг власних об’єктів виробництва або навіть власного житла.

Не варто забувати, що такі потреби можуть виникати не тільки у підприємствах з багатотисячними бюджетами, а й у звичайних споживачів і їм необхідно пропонувати негірший продукт, але значно дешевший. Саме тому і було обрано дану тему, щоб дослідити, що являє собою мережа інтернету речей, які технології та протоколи там застосовуються, яке мінімальне апаратне та програмне забезпечення необхідне для функціонування системи керування та моніторингу елементами інтернету речей.

У даній магістерській роботі досліджено засоби моніторингу та управління елементами *IoT* з метою полегшення управління та моніторингу елементів інтернету речей, а також виявлення необхідних засобів для роботи такої системи.

Проведені дослідження дозволяють зрозуміти, які програмні та апаратні засоби необхідні для побудови системи управління та моніторингу елементами інтернету речей. А також, як це зробити з мінімальними матеріальними внесками.

Це дослідження є досить ефективним для використання, тому що в ньому наведені конкретні пристрої, програмні засоби та кроки, для синтезу програмних і апаратних забезпечень, що в свою чергу дає змогу використати ці результати у власних цілях та побудувати власну мережу інтернету речей.

У роботі проведений аналіз мережі інтернету речей з метою виявлення особливостей мережі інтернету речей. Проаналізовано, які протоколи, технології та типи мереж застосовуються для побудови мережі *IoT*, на якій архітектурі побудовані дані мережі, які пристрої беруть участь у роботі мережі. Проаналізовано, які задачі виконує моніторинг і навіщо його застосовувати, а також проаналізовано головні проблеми та задачі кібербезпеки мережі інтернету речей. У підсумку це дозволяє обрати необхідні протоколи та технології, на яких буде базуватися майбутня система.

У роботі пропонується система моніторингу та управління приладами на основі *Raspberry pi*. Пропонується компактна мережу з можливістю Інтернет. Система може контролювати стан датчиків через Інтернет, коли оновлення інформації на веб-сервері зчитується за розробленим алгоритмом, що подається в *Raspberry pi*, а потім система відповідає на відповідні інструкції з високим захистом. Користувач може безпосередньо входити в систему та взаємодіяти із вбудованим пристроєм у режимі реального часу. Система гнучка для розміщення широкого спектру вимірювальних приладів з відповідними інтерфейсами. Вона має різноманітні переваги, такі як енергоефективність, інтелект, низька вартість, портативність та висока продуктивність.

Також було проаналізовано апаратне забезпечення. Для забезпечення функціонування системи управління та моніторингу потрібно вибрати таке апаратне забезпечення, яке забезпечуватиме виконання поставлених задач та цілей. Інтернет речей - це не тільки безліч різних приладів і датчиків, об'єднання між собою дротяними і бездротовими каналами зв'язку і підключати чинних до мережі Інтернет, а тісна інтеграція реального та віртуального світів, в середовищі якої спілкування

здійснюється між людьми і пристроями. Тому було запропоновано міні-ПК *Pasbperry Pi*, як серце системи. Міні-ПК *Pasbperry Pi* дозволяє забезпечити усім необхідним для роботи системи, є економічним та портативним варіантом для побудови системи управління та моніторингу елементами *IoT*. Стосовно до статистичного управління процесами – методу моніторингу виробничих процесів з метою управління якістю продукції безпосередньо в процесі виробництва, міні-ПК *Raspberry Pi* є першим доступним технічним рішенням такого розміру, яке можна використовувати повсюдно для програмування на багатьох мовах і в якості мікроконтролерів для управління роботизованими пристроями. Також для повноцінної роботи міні-ПК *Pasbperry Pi* було підібране необхідне обладнання у вигляді монітору, клавіатури, маніпулятора, адаптера живлення тощо.

Іншою важливою частиною є налаштування засобів моніторингу. У частині налаштуваання було встановлено та сконфігуровано *Domoticz*. *Domoticz* - це дуже легка система домашньої автоматизації, яка дозволяє вам контролювати і настроювати різні пристрої, включаючи світло, вимикачі, різні датчики / лічильники, такі як температура, опади, вітер, ультрафіолетове (УФ) випромінювання, споживання електроенергії / виробництво, споживання газу, споживання води і багато іншого. Це є важливим у своєму практичному значені. Тобто споживач міг би скористатися даним дослідженням і зрозуміти, яке обладнання йому необхідне для побудови системи управління та моніторингу, який програмний комплекс бере у цьому участь, а також, які кроки йому необхідно пройти перш ніж система запрацює.

Також у даному дослідженні проведений аналіз, який у підсумку дає зрозуміти, яка архітектура застосунків використовується. А саме для подальшої розробки найбільше підходить архітектура клієнт-сервер. Виходячи з вибраної архітектури для додатку, який забезпечує керування системою, обрано мову *Java* є простий, об’єктно-орієнтований синтаксис, безпечна та безвідмовна робота з пам’яттю, незалежність від платформи та архітектури, висока продуктивність. Пропоновано серверний додаток за архітектурним шаблоном *MVC*, а клієнтський за монолітною архітектурою і запропоновані засоби за допомогою яких є можливість розробки користувацького інтерфейсу.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Iot based monitoring and control system for appliances* [електронний ресурс]. – 2018. – режим доступу до ресурсу: https://www.ripublication.com/acst18/acstv11n1_04.pdf.

2. Довгаль В. А. Интернет вещей: концепция, приложения и задачи / В. А. Довгаль, Д. В. Довгаль. // научный журнал «вестник агу». – 2018. – с. 129–136.

3. Как изменится система мониторинга производительности в эпоху интернета вещей (*iot*)? [електронний ресурс] – режим доступу до ресурсу: *https://networkguru.ru/sistema-monitoringa-proizvoditelnosti-interneta-veshchei-iot/*.

4. *Internet of things*: все, что нужно знать об интернете вещей и о будущем современной цивилизации [електронний ресурс] – режим доступу до ресурсу: *https://www.everest.ua/ru/internet-of-things-vse-chto-nuzhno-znat-ob-ynternete-veshhej-y-o-budushhem-sovremennoj-czyvylyzaczyy/*.

5. Боцман Я. Интернет вещей, *iot, m2m* мировой рынок [електронний ресурс] / Ярослав Боцман. – 2017. – режим доступу до ресурсу: https://channel4it.com/publications/internet-veshchey-25146.html.

6. Довгаль В. А. Проблемы и задачи безопасности интеллектуальных сетей, основанных на интернете вещей / В. А. Довгаль, Д. В. Довгаль // ежеквартальный рецензируемый, реферируемый научный журнал «вестник агу» / В. А. Довгаль, Д. В. Довгаль., 2017. – с. 140–147.

7. Технологии и протоколы интернета вещей [електронний ресурс] – режим доступу до ресурсу: *https://azure.microsoft.com/ru-ru/overview/internet-of-things-iot/iot-technology-protocols/*.

8. Денисов Д. Введение в концепцию "интернета вещей" (*iot*) [електронний ресурс] / Д. Денисов, К. Брагин. – 2020. – режим доступу до ресурсу: https://nag.ru/articles/article/107810/vvedenie-v-kontseptsiyu-interneta-veschey-iot-.html.

9. Створення інформаційної системимоніторингу забруднення атмосферногоповітря міста на основі технології «інтернет речей» / [В. Б. Мокін, Б. Ю.

Собко, М. В. Дратований та ін.]. // вінницький національний технічний університет. – 2017. – №3. – с. 49–57.

10. Горяинов А. Н. Логистический мониторинг и транспортная диагностика в эпоху интернета вещей [електронний ресурс] / А. Н. Горяинов. – 2018. – режим доступу до ресурсу: https://www.researchgate.net/publication/333531998_logisticeskij_monitoring_i_transportnaa_diagnostika_v_epohu_interneta_vesej.

11. *What is iot architecture*? [електронний ресурс]. – 2020. – режим доступу до ресурсу: https://www.avsystem.com/blog/what-is-iot-architecture/.

12. *Remote monitoring and alerting for iot* [електронний ресурс]. – 2020. – режим доступу до ресурсу: *https://cloud.google.com/solutions/remote-monitoring-and-alerting-for-iot*.

13. Петін В. А. *Arduino* и *raspberry pi* в проектах *internet of things* / Віктор А. Петин. – санкт-петербург: «бхв-петербург», 2018. – 432 с.

14. Петін В. А. *Arduino* и *raspberry pi* в проектах *intemet ofthings*. / Віктор А. Петін. – Санкт-Петербург: бхв-петербург, 2016. – 320 с.

15. Альтернативы *raspberry pi* [електронний ресурс]. – 2019. – режим доступу до ресурсу: https://habr.com/ru/post/457666/.

16. *Bmp*180 *barometric pressure/temperature/altitude sensor- 5v ready* [електронний ресурс] – режим доступу до ресурсу: https://www.adafruit.com/product/1603.

17. *Tsl*2561 — цифровой датчик освещенности (модуль *gy*-2561) [електронний ресурс]. – 2018. – режим доступу до ресурсу: *https://micro-pi.ru/tsl*2561-цифровой-датчик-освещенности/.

18. *Wifi* адаптер *edup ep-n*8531 [електронний ресурс] – режим доступу до ресурсу: http://china-electronics.com.ua/ru/edup/680-wifi-edup-ep-n8531.html.

19. Датчик температуры *ds*18*b*20 цифровой [електронний ресурс] – режим доступу до ресурсу: http://arduino.ua/prod190-datchik-temperatyri-ds18b20-cifrovoi.

20. Барометр *bmp*180 (датчик атмосферного давления) [електронний ресурс]. – 2019. – режим доступу до ресурсу: *http://arduino.ua/prod*664-*barometr_bmp*085.

21. *The official raspberry pi how to use your new computer beginner's guide* [електронний ресурс] – режим доступу до ресурсу: *https://www.raspberrypi.org/magpi-issues/beginners_guide_v1.pdf*.

22. Установка операционной системы [електронний ресурс]. – 2018. – режим доступу до ресурсу: https://raspberrypi.ru/operating_system_install.

23. Настроить *ssh* на *raspberry pi* — это легко [електронний ресурс] – режим доступу до ресурсу: https://myraspberry.ru/nastroit-ssh-na-raspberry-pi-—-eto-legko.html.

24. Система *domoticz* [електронний ресурс] – режим доступу до ресурсу: *https://domoticzfaq.ru/о-системе-domoticz/*.

25. *Domoticz* на *raspberry pi* (обзор) + первая настройка за 3 пункта [електронний ресурс]. – 2019. – режим доступу до ресурсу: https://domoticzfaq.ru/raspberry-pi-умный-дом-на-ладони/.

26. Обзор интерфейса *domoticz* [електронний ресурс]. – 2019. – режим доступу до ресурсу: https://domoticzfaq.ru/обзор-интерфейса-domoticz/.

27. *Domoticz api/json url's* [електронний ресурс]. – 2020. – режим доступу до ресурсу: https://www.domoticz.com/wiki/domoticz_api/json_url%27s.

28. *Ho C. Pro spring* 3 / *c. Ho, R. Harrop. – united kingdom*: *apress*, 2012. – 944 c.

29. Что такое *spring framework*? От внедрения зависимостей до *web mvc* [електронний ресурс]. – 2020. – режим доступу до ресурсу: https://habr.com/ru/post/490586/.

30. *Bloch J. Effective java third edition* [електронний ресурс] / *Joshua Bloch*. – 2018. – режим доступу до ресурсу: *https://kea.nu/files/textbooks/new/effective%20java%20%282017%2c%20addison-wesley%29.pdf*.

31. Будай А. Дизайн-патерни — просто, як двері [електронний ресурс] / Андрій Будай. – 2012. – режим доступу до ресурсу: https://learn.ztu.edu.ua/pluginfile.php/632/mod_resource/content/1/designpatterns_andriybuday.pdf.

32. Модель-вид-контролер *[*електронний ресурс*]*. – 2019. – режим доступу до ресурсу: *https://uk.wikipedia.org/wiki/*модель-вид-контролер.

33. Интернет вещей, *iot, m2m* мировой рынок *[*електронний ресурс*]*. – 2020. – режим доступу до ресурсу: *https://www.tadviser.ru/index.php/*статья:интернет_вещей,_*iot,_m2m*_(мировой_рынок)

34. Информационная безопасность интернета вещей (*internet of things*) *[*електронний ресурс*]*. – 2020. – режим доступу до ресурсу: *https://www.tadviser.ru/index.php/*статья:информационная_безопасность_интернета_вещей_(*internet_of_things*)