МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського
«Харківський авіаційний інститут»

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

# Пояснювальна записка до дипломної роботи

магістра
(освітній ступінь)

на тему «Експериментальне дослідження системи передачі даних на платформі інтернету речей»
ХАІ.603.695пз1.121.160–9/19

Виконав: студент 6 курсу групи №6-95Пз1
Спеціальність 121 Інженерія програмного забезпечення
(код та найменування)
Освітня програма Хмарні обчислення та Інтернет речей
(найменування)
Шевко В. С.
(прізвище й ініціали студента)
Керівник: Кузнецова Ю. А.
(прізвище й ініціали)
Рецензент: Коваленко А. А.
(прізвище й ініціали)

Харків – 2020

**Міністерство світи і науки України**
**Національний аерокосмічний університет ім. М. Є. Жуковського**
**«Харківський авіаційний інститут»**

Факультет програмної інженерії та бізнесу
(повне найменування)
Кафедра інженерії програмного забезпечення
(повне найменування)
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
(код та найменування)
Освітня програма Хмарні обчислення та Інтернет речей
(найменування)

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
_____ І. Б. Туркін
(підпис) (ініціали та прізвище)
“_____” _____ 2020 року

# З А В Д А Н Н Я
## НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Шевко Валентин Сергійович
(прізвище, ім’я, ПЗ батькові)

1. Тема дипломної роботи: «Експериментальне дослідження системи передачі даних на платформах інтернету речей», керівник дипломної роботи Кузнецова Юлія Анатоліївна к. т. н., доцент
( прізвище, ім’я, ПЗ батькові, науковий ступінь, вчене звання)

затверджені наказом Університету № _____ від “____” _____ 2020 року

2. Термін подання студентом проекту 30.11.2020

3. Вихідні дані до проекту формулювання проблемних питань експериментальних досліджень системи передачі даних на платформах інтернету речей

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити): проаналізувати проблеми експериментального дослідження передачі даних на платформах інтернету речей, провести планування експериментального дослідження передачі даних в IoT системах, виконати аналіз результатів експериментального дослідження системи передачі даних на платформі інтернету речей.

5. Перелік графічного матеріалу:

РПЗ – стор. 66, рисунків – 41 шт., таблиць – 2 шт., формул – 11 шт., слайдів презентації – 22 шт.

6. Консультанти розділів проекту

| Розділ | Прізвище, ініціали та посада Консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1 | Кузнецова Ю.А., доц. каф. 603 | | |
| 2 | Кузнецова Ю.А., доц. каф. 603 | | |
| 3 | Кузнецова Ю.А., доц. каф. 603 | | |

7. Дата видачі завдання _____ 2020 р.

8. Нормоконтроль _____ _____ «____» _____2020 р.
(підпис) (ініціали та прізвище)

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Отримання і затвердження теми диплому | 03.09.2020 | |
| 2 | Аналіз проблеми експериментального дослідження передачі даних на платформах інтернету речей | 02.09.2020–25.09. 2020 | |
| 3 | Планування експериментального дослідження передачі даних в IoT системах | 28.09.2020–15.10.2020 | |
| 4 | Розробка прототипу ПЗ | 16.10.2020–26.10.2020 | |
| 5 | Аналіз результатів експериментального дослідження системи передачі даних на платформі інтернету речей | 27.10.2020–19.11.2020 | |
| 6 | Оформлення пояснювальної записки до НДР магістра | 20.11.2020–25.11.2020 | |
| 7 | Передзахист дипломної роботи | 30.11.2020 | |
| 8 | Захист дипломної роботи | 16.12.2020 | |

**Студент** _____ Шевко В. С
(підпис) (прізвище та ініціали)

**Керівник проекту** _____ Кузнецова Ю. А.
(підпис) (прізвище та ініціали)

# РЕФЕРАТ

Дипломний проект: 66 с., 41 рисунків, 2 таблиці, 11 формул, 28 посилань.

Інтернет речей (IoT) швидко зростає в наш час. IoT це мережа, яка з'єднує фізичні пристрої, автомобілі, будівлі та інші об'єкти і в яку вмонтовано електронну систему, програмне забезпечення, датчики, виконавчі механізми та мережеве з'єднання, щоб ці об'єкти могли збирати дані та обмінюватися ними. IoT використовується в багатьох областях, таких як область транспорту та логістики, галузь охорони здоров'я, область інтелектуального середовища, особиста і соціальна область. За оцінками, до 2021 року IoT буде складатися з майже 70 мільярдів об'єктів. IoT Gateway дійсно важливий в IoT, оскільки він може з'єднувати традиційні мережі зв'язку з мережами датчиків, щоб спростити мережеву взаємодію. IoT-комунікація має життєво важливе значення в сьогоднішньому житті. Це дослідження спрямоване на інтеграцію і оцінку шлюзів Інтернету речей і систем зв'язку Інтернету речей. В цьому дослідженні пропонується сценарій, в якому шлюз IoT підключається до виконавчого механізму для керування виконавчим механізмом та передачі даних через систему зв'язку IoT, створюється демонстратор, який налаштовує зв'язок між платформою шлюзу IoT та системою зв'язку IoT, вимірює і оцінює продуктивність з точки зору затримки і пропускної здатності з використанням реалізованого сценарію, і, нарешті, робиться висновок.

Об'єкт дослідження – процес передачі даних в системах IoT.

Предмет дослідження – мережева взаємодія компонентів системи на рівні програмного забезпечення для передачі даних.

Методи дослідження. Досягнення мети роботи базується на використанні теорії системного аналізу, що дозволяє комплексно розглянути всі аспекти, пов'язані з розвитком технологій застосування Інтернету речей, теорії експерименту, що дозволяє виявити значущі фактори та відгуки експерименту, методів математичної статистики для оброблення результатів експерименту та методів інженерії програмного забезпечення, для проектування та розроблення прототипу системи інтернету речей.

Наукова новизна та інноваційність отриманих результатів. Вперше розроблено план експериментального дослідження системи передачі даних на платформах інтернету речей, який на від міну від існуючих базується на використанні відкритих технологій. Інноваційна складова дослідження полягає в використанні стеку технологій інтернету речей, що є інноваційним напрямком в розвитку інформаційних технологій.

Практичне значення отриманих результатів. Результати роботи можуть бути використані для проектування систем інтернету речей з точки зору системи передачі даних, та подальшої реалізації та тестування розроблених проектів.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ РЕЧЕЙ, ШЛЮЗ, ЗВ'ЯЗОК, АКТУАТОР, ЗАТРИМКА, ПРОПУСКНА ЗДАТНІСТЬ.

## ABSRACT

Master’s thesis: 66 pages, 41 figures, 2 tables, 11 formulas, 28 references.

The Internet of Things (IoT) is growing rapidly these days. IoT is a network that connects physical devices, cars, buildings, and other objects and in which an electronic system, software, sensors, actuators, and network connection are embedded so that these objects can collect and share data. . IoT is used in many areas, such as transport and logistics, health, intellectual environment, personal and social. It is estimated that by 2021, the IoT will consist of nearly 70 billion objects. The IoT Gateway is important in IoT because it can connect traditional communication networks to sensor networks to simplify network interaction. IoT communication is vital in today's life. This study aims to integrate and evaluate IoT gateways and IoT communication systems. This study proposes a scenario in which an IoT gateway connects to an actuator to control the actuator and transmit data over an IoT communication system, creates a demonstrator that configures communication between the IoT gateway platform and the IoT communication system, and measures and evaluates performance in terms of delay and bandwidth using the implemented scenario, and finally a conclusion is drawn.

The object of study - the process of data transmission in IoT systems.

The subject of research - network interaction of system components at the level of software for data transmission.

Research methods. Achieving the goal of the work is based on the use of the theory of systems analysis, which allows to consider all aspects related to the development of Internet of Things technologies, experimental theory, which allows to identify significant factors and responses of the experiment, methods of mathematical statistics for processing experimental results. providing, for the design and development of a prototype Internet of Things system.

Scientific novelty and innovation of the obtained results. For the first time, a plan for experimental research of the data transmission system on the Internet of Things platforms has been developed, which, unlike the existing ones, is based on the use of open technologies. The innovative component of the study is the use of the Internet of Things technology stack, which is an innovative direction in the development of information technology.

The main practical result of the work is used to design Internet of Things systems in terms of data transmission system, and further implementation and testing of developed projects.

KEY WORDS: INTERNET OF THINGS, GATEWAY, COMMUNICATION, ACTUATOR, DELAY, CAPACITY.

# ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

IoT, Internet of things – технологія Інтернет речей.
IIoT, Industrial IoT – промисловий Інтернет речей.
NGFF – форм-фактор нового покоління.
RFID – визначення радіочастоти.
SCTP – протокол передачі управління потоком.
SDK – комплект для розробки програмного забезпечення.
SDP – протокол виявлення служби.
TCP – протокол управління передачею.
UCI – універсальні ідентифікатори контексту.
UDP – протокол призначених для користувачів датаграм.
URL – універсальний покажчик ресурсів.
USB – універсальна послідовна шина.
VPN – віртуальна приватна мережа.
WSAN – бездротова мережа датчиків і виконавчих механізмів.
WSN – бездротова сенсорна мережа.
ПЗ – програмне забезпечення.

# ЗМІСТ

# ВСТУП

Інтернет речей (Internet of Thinks, IoT) – це мережа, яка з'єднує фізичні пристрої, будівлі, транспортні засоби та інші об'єкти і в яку вмонтовано електронну систему, програмне забезпечення, датчики, виконавчі механізми та мережеве з'єднання, щоб ці об'єкти могли збирати дані і обмінюватися ними [1] . IoT може сприймати об'єкти або керувати ними віддалено через існуючу мережеву інфраструктуру, створювати можливості для більш прямої інтеграції фізичного світу в системи, засновані на комп'ютерах, щоб зменшити втручання людини, підвищити точність, ефективність й економічні вигоди [2]. У ці роки ця технологія швидко розвивається та має великий вплив на повсякденне життя та поведінку. IoT використовувався в багатьох областях, таких як область транспорту та логістики, галузь охорони здоров'я, область інтелектуального середовища, особиста й соціальна область. IoT вважається третьою хвилею розвитку інформаційної індустрії після мережі мобільного зв'язку та Інтернету, характерними рисами якої є більш всеосяжний сенс і вимір, більш ретельний інтелект і функціональна сумісність [3]. Інтернет речей був визначений як «інфраструктура інформаційного суспільства» в 2013 році Глобальною ініціативою по стандартизації Інтернету речей (IoT-GSI) [4].

Прогнозується, що до 2022 року Інтернет речей буде складатися з майже 90 мільярдів об'єктів [5]. Таким чином, Інтернет речей буде мати великий вплив на те, якими ми будемо і яким буде світ. Може бути, весь світ буде пов'язаний, і це не буде спілкування, яке є в однієї людини. Все в світі може бути пов'язано і спілкуватися один з одним. Ми будемо в іншій епосі.

Інтернет речей складається з множини програмних і апаратних засобів, які взаємодіють один з одним. Ось чому Інтернет речей може працювати, так що зв'язок і взаємодія між програмним і апаратним забезпеченням дійсно важливі. Система зв'язку – це сукупність ретрансляційних станцій, кінцевого обладнання даних (DTE), систем передачі та окремих мереж зв'язку, які зазвичай взаємопов'язані і взаємодіють один з одним за певними протоколами, утворюючи єдине ціле [6].

Сценарії IoT включають логістику, інтелектуальний будинок, інтелектуальну архітектуру і так далі. Одна з основ цього успіху – система зв'язку, яка забезпечує передачу інформації між апаратною платформою та програмною платформою. Щоб була можливість керувати даними з обладнання й отримувати те, що нам потрібно, за допомогою програмного забезпечення. Також можливо працювати з інформацією, що передається від обладнання, і управляти нею. Але важливо, знати, як виміряти та оцінити ефективність комунікації. Тому що буде марно, якщо передача буде дійсно повільної, так що інформація не буде отримана вчасно та цілісно.

Дослідженням шляхів та напрямків впровадження Інтернету речей та хмарних обчислень присвячено значну кількість робіт, серед яких слід особливо відзначити роботи вчених: Д. Андерсона, Д. Баркера, Д. Еванса, К. Ештона, Н. Карра, К. Пістера, Д. Хелбінга, Е. Шмідта, В. Бикова,

Н. Богданової, С. Ваняшина, А. Гребешкова, В. Довгаля, А. Рослякова, А. Сівкова, М. Самсонова, З. Сейдаметової, С. Сейтвелієвої, І. Шадеркіної, В. Шадеркіної та ін.

Існують інструменти оцінки передачі даних в системах інтернету речей, але механізми оцінки недоступні та інколи не зрозумілі користувачу або розробнику програмного забезпечення інтернету речей, таким чином, в даний час залишається актуальним експериментальне дослідження системи передачі даних на платформах інтернету речей

Метою роботи є виявлення закономірностей в системі передачі даних на платформі інтернету речей.

Для досягнення поставленої мети потрібно вирішити наступні задачі:

1) проаналізувати найбільш поширені підходи до комунікації на платформах Інтернету речей та найбільш поширені підходи до систем зв'язку Інтернету речей;

2) провести планування експериментального дослідження систем передачі даних на платформах інтернету речей;

3) виконати експеримент з дослідження передачі даних на платформах інтернету речей;

4) проаналізувати експериментальні дані отриманні в ході експерименту з передачі даних на платформах інтернету речей.

Об'єкт дослідження – процес передачі даних в системах IoT.

Предмет дослідження – мережева взаємодія компонентів системи на рівні програмного забезпечення для передачі даних.

Методи дослідження. Досягнення мети роботи базується на використанні теорії системного аналізу, що дозволяє комплексно розглянути всі аспекти, пов'язані з розвитком технологій застосування Інтернету речей, теорії експерименту, що дозволяє виявити значущі фактори та відгуки експерименту, методів математичної статистики для оброблення результатів експерименту та методів інженерії програмного забезпечення, для проектування та розроблення прототипу системи інтернету речей.

Наукова новизна та інноваційність отриманих результатів. Вперше розроблено план експериментального дослідження системи передачі даних на платформах інтернету речей, який на від міну від існуючих базується на використанні відкритих технологій. Інноваційна складова дослідження полягає в використанні стеку технологій інтернету речей, що є інноваційним напрямком в розвитку інформаційних технологій.

Практичне значення отриманих результатів. Результати роботи можуть бути використані для проектування систем інтернету речей з точки зору системи передачі даних, та подальшої реалізації та тестування розроблених проектів.

# 1 АНАЛІЗ ПРОБЛЕМИ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ ПЕРЕДАЧІ ДАНИХ НА ПЛАТФОРМАХ ІНТЕРНЕТУ РЕЧЕЙ

## 1.1 Інтернет речей

IoT був вперше представлений в 1999 році: за допомогою радіочастотної ідентифікації (RFID), інфрачервоних датчиків, глобальної системи позиціонування, лазерного сканера, газових датчиків та іншого устаткування для зчитування інформації, відповідно до узгоджених протоколів, з'єднування будь-яких об'єктів з Інтернетом для обміну інформацією та комунікації, щоб досягти інтелектуальної ідентифікації, позиціонування, відстеження, моніторингу та керування [7].

Репрезентативну архітектуру IoT можна розділити на три області: домен додатку, мережевий домен та домен виміру, які показані на рисунку 1.1.

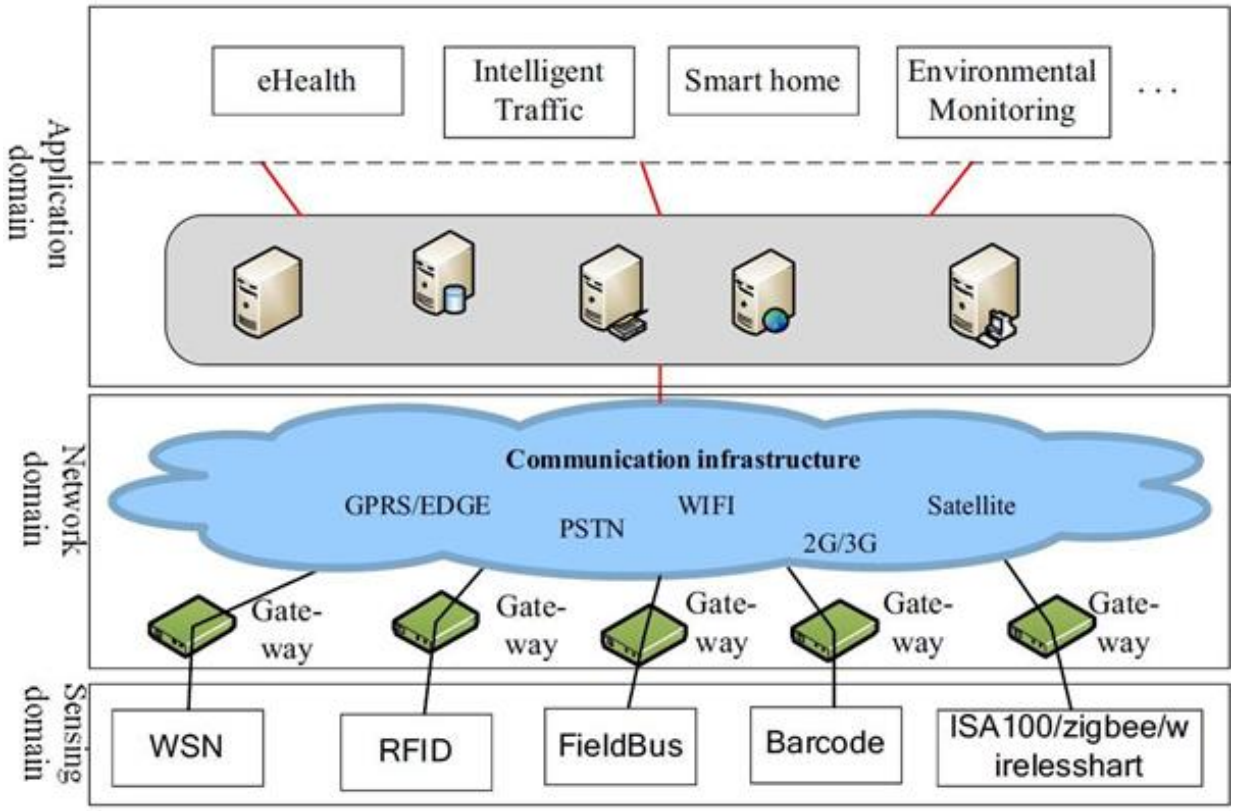



Рисунок 1.1 – Три галузі архітектури Інтернету речей

Домен додатку призначений для обробки даних з мережевого домена та надання різних послуг для всіх типів користувачів. Мережевий домен заснований на існуючій інфраструктурі зв'язку або інфраструктурі зв'язку, що розвивається, такій як супутникова, 3G (третє покоління) та LTE

(довгостроковий розвиток), основною метою яких є передача інформації, зібраної з області сенсорів, в віддалений пункт призначення.

Домен сенсорів дуже важливий для Інтернету речей, який змушує «речі» спілкуватися та взаємодіяти між комунікаційними інфраструктурами й самими собою. Цей домен використовує такі технології, як Zigbee, WSN і RFID, для збору інформації про фізичних цілях і може обробляти зібрані дані для надання цінних даних від датчиків для мережевого домену.

## 1.2 Шлюзи Інтернету речей

Шлюз IoT розроблений для усунення неоднорідності між різними мережами зв'язку та сенсорними мережами. Щоб посилити керування кінцевими вузлами та WSN, з'єднати сенсорні мережі з традиційними комунікаційними мережами, та щоб спростити мережеву взаємодію та керування обладнанням сенсорної мережі [3]. Архітектура шлюзу IoT зображена на рисунку 1.2.

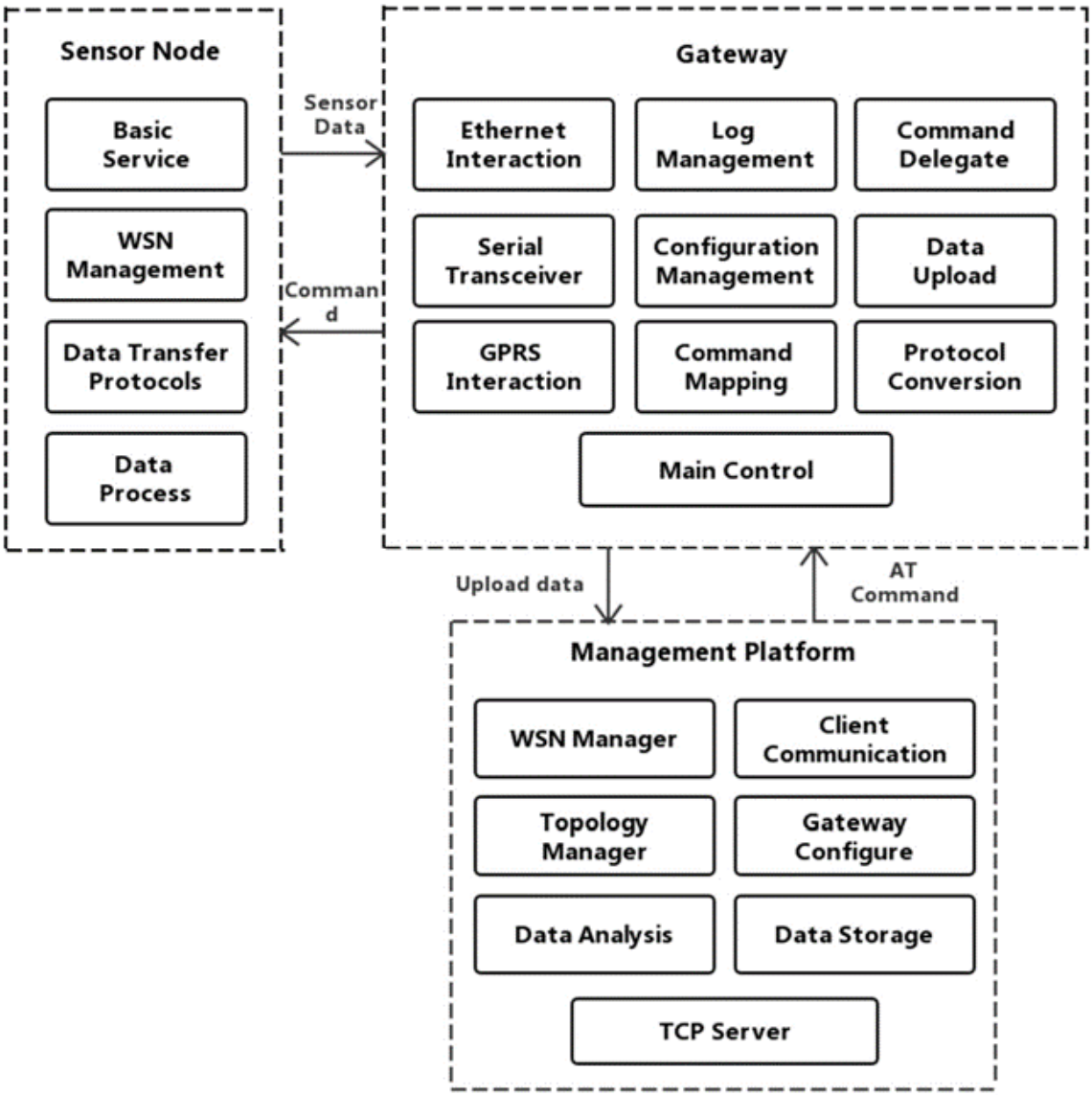



Рисунок 1.2 – Програмна архітектура шлюзу Інтернету речей

В якості моста, що з'єднує традиційні мережі зв'язку з сенсорними мережами, IoT Gateway може забезпечувати функції фізичного пристрою керування та перетворення протоколу. Загальний шлюз Інтернету речей має наступні загальні особливості [3]:

– широкий спектр можливостей доступу: сучасні стандарти комунікаційних технологій короткого діапазону варіюються та включають Rubee, Z-Wave, Zigbee тощо. Однак йому не вистачає сумісності з протоколами;

– керованість: необхідно керувати тисячами кінцевих вузлів датчиків, оскільки програма IoT має на серверах вузли датчиків та програму, що працює на серверах, щоб відчувати та контролювати фізичний світ. Керування шлюзом IoT означає не тільки керування вузлом датчика, але також означає керування пристроєм шлюзу;

– взаємодія протоколів: шлюз IoT повинен підтримувати взаємодію протоколів між WSN та традиційною мережею, оскільки їм потрібно обмінюватися інформацією в певних сценаріях IoT.

### 1.3 IoT-комунікації

У IoT важливий зв'язок між речами. Зв’язок M2M (машина – машина) є однією з найважливіших компетенцій для впровадження Інтернету речей. Категорії IoT-комунікації наведено на рисунку 1.3.

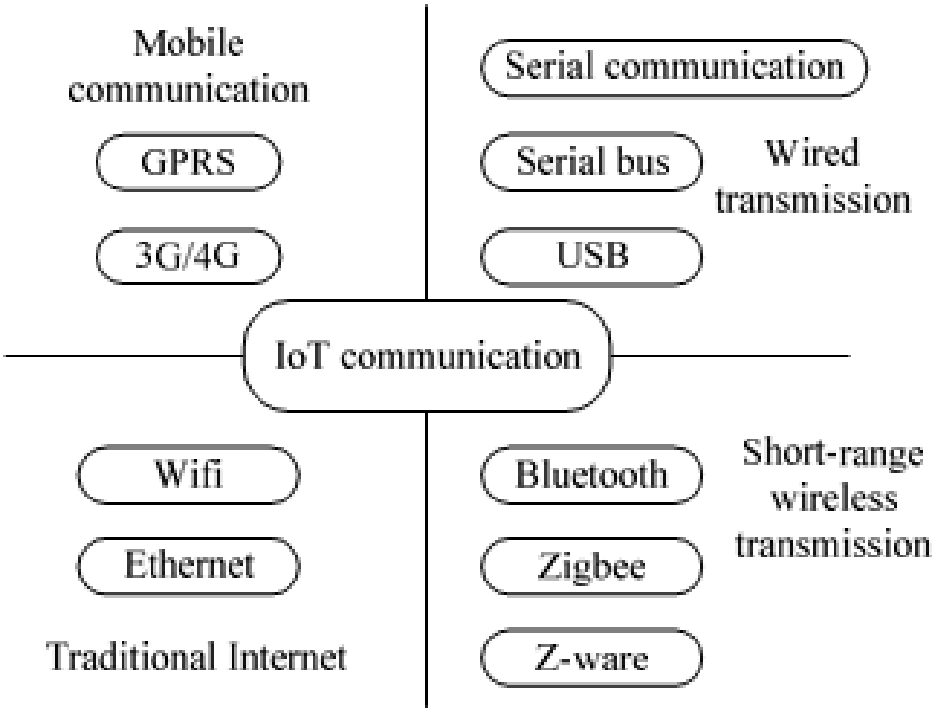



Рисунок 1.3 – Категорії IoT комунікацій

Зазвичай використовуються методи зв'язку IoT поділяються на чотири категорії: дротова передача, з яких в основному використовуються інтерфейс послідовного зв'язку, послідовна шина, універсальна послідовна шина (USB), бездротова передача на короткі відстані, яка в основному складається з таких технологій, як Bluetooth, Zigbee та Z-ware , традиційним Інтернетом для якого в основному є WIFI та Ethernet, та технологіями мобільного: 3G / 4G (четверте покоління), GPRS (General Packet Radio Service) [8].

Але е альтернативний підхід архітектури, яка базується на хмарних обчисленнях, які можна охарактеризувати як децентралізований обчислювальний процес, де його пам’ять та обчислювальна потужність знаходяться між джерелом даних та хмарою. Це вважається потужним методом, оскільки він розподіляє свої обчислювальні зусилля через край мережі. Завдяки архітектурі рішення додатку надається можливість вибору,

звідки використовувати перетворені дані. Це може бути доступно з (загальнодоступної або обмеженої) хмари або інфраструктури туману. Рисунок 1.4 відображає узагальнену концепцію SOFTWAY4IoT.

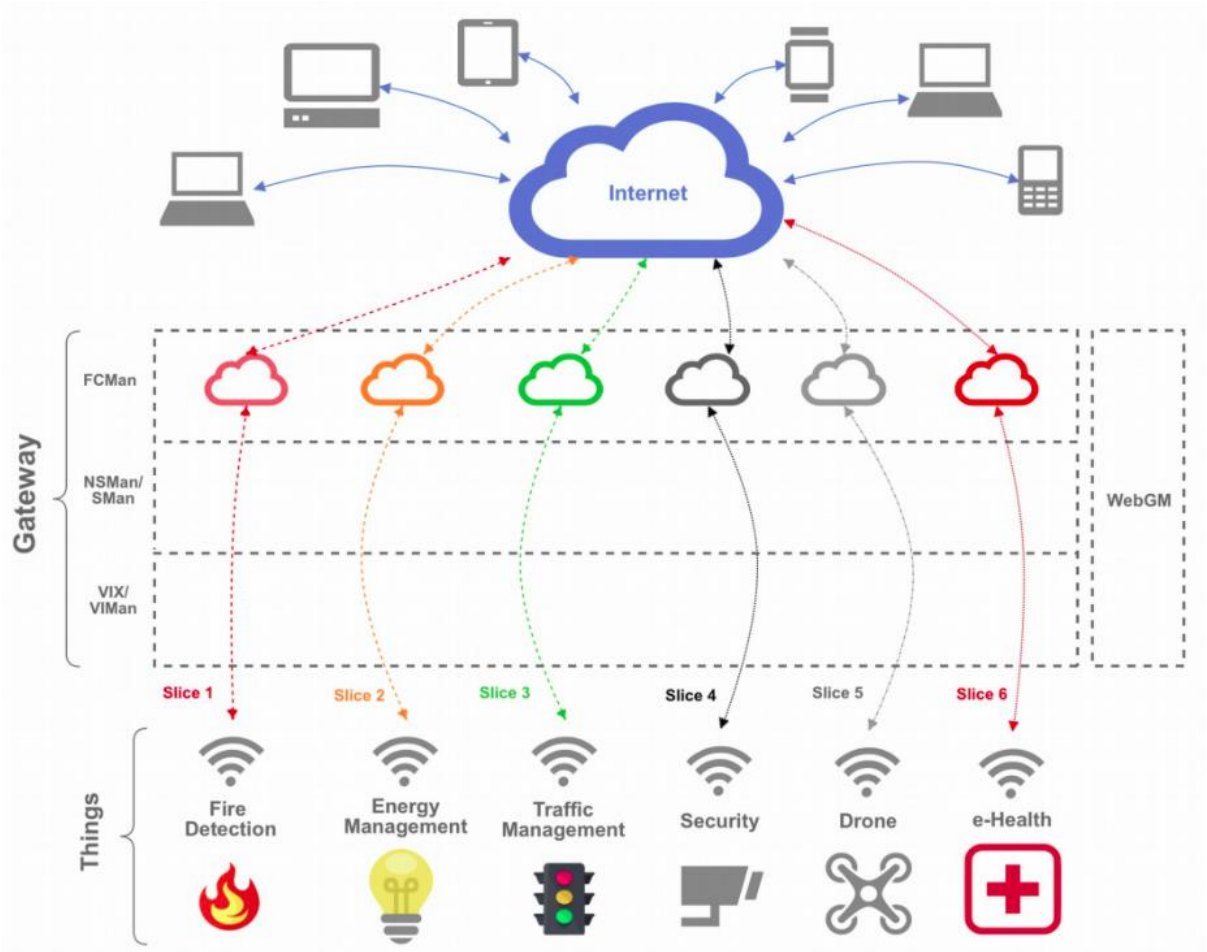



Рисунок 1.4 – Узагальнена концепція SOFTWAY4IoT

Оскільки звичайні мережі не використовують переваги SDR, кожна розглянута бездротова технологія потребує виділеного шлюзу або фізичного інтерфейсу, щоб мати можливість здійснювати зв'язок.

Оскільки SDR дозволяє багатьом технологіям працювати через один і той же шлюз і навіть додає / видаляє ці можливості лише за допомогою програмного забезпечення, то процес обробки різних бездротових технологій вважається спрощеним, як показано на рисунку 1.5.

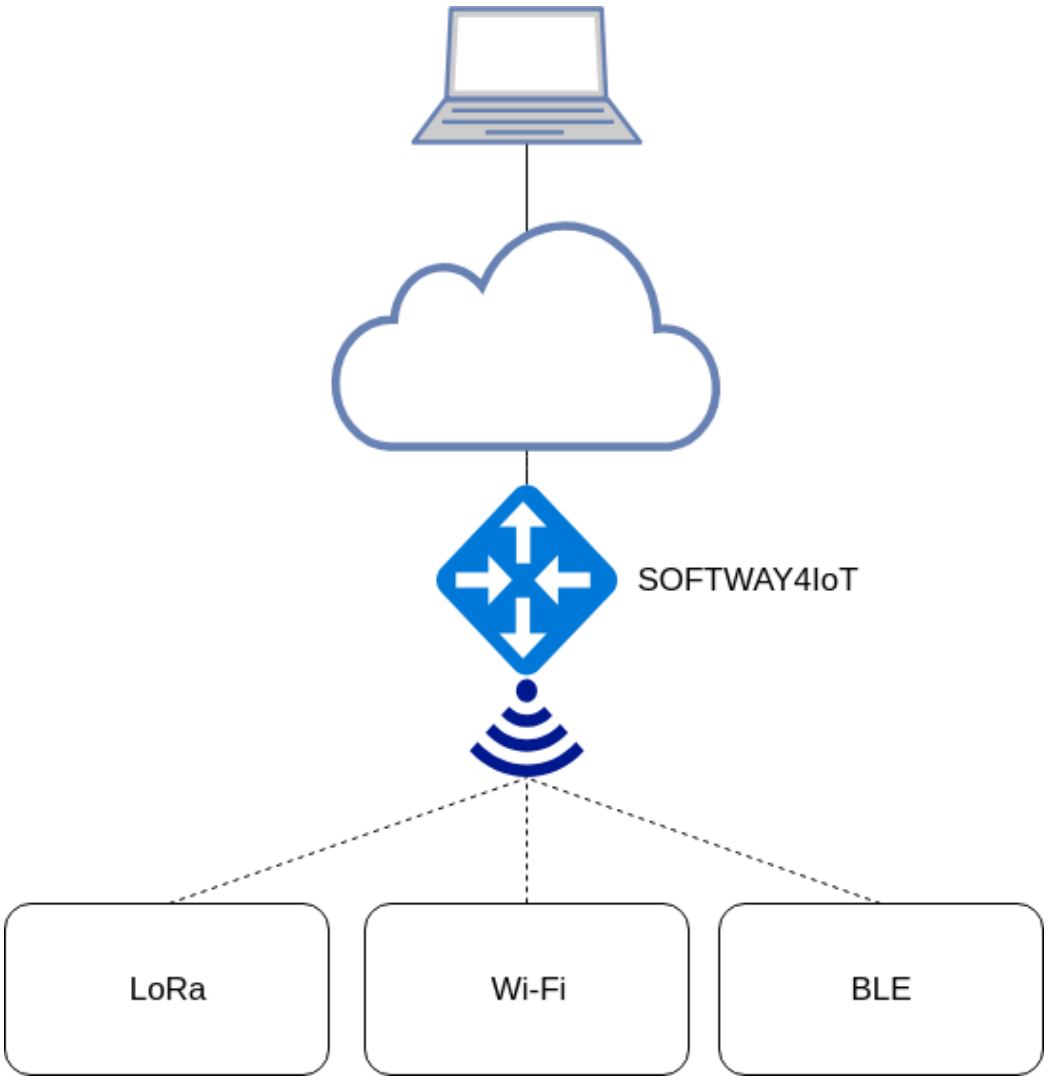



Рисунок 1.5 – Комунікація з різними бездротовими мережами

## 1.4 Огляд програмно-апаратного забезпечення для IOT комунікацій

### 1.4.1 Шлюз Vinnter IoT

Платформа інтелектуальних продуктів Vinnter була розроблена для відповідності вимогам промислового рівня, додаючи можливості підключення до існуючих і нових продуктів, маючи при цьому широкі можливості настройки. Це може прискорити підтвердження концепції і розробку продукту, а також знизити ризики проекту, вартість і час виведення на ринок. Вона має основну плату, яка забезпечує можливості підключення, а додаткові плати керування або інтерфейсу користувача поліпшить можливості платформи. Плата показана на рисунку 1.6.

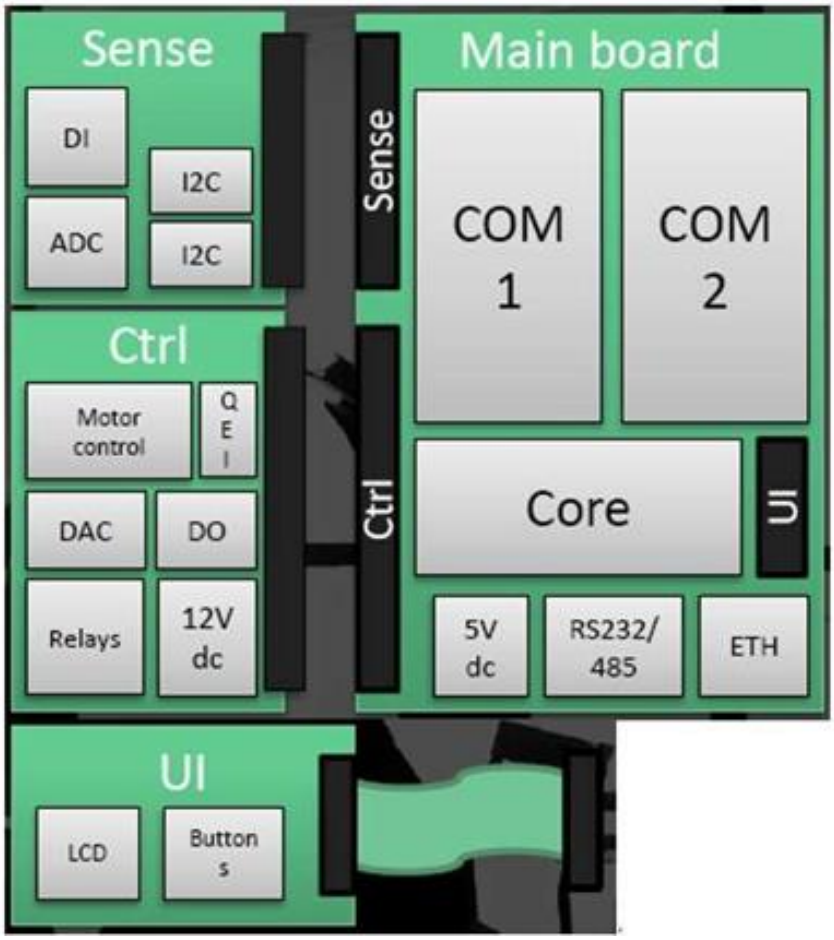



Рисунок 1.6 – Плата Vinnter IoT

Vinnter IoT шлюз є частиною інтелектуальної платформи продукту. Модернізація, підключення розроблено для багатьох інтерфейсів та модульних бездротових карток, що дозволяє використовувати будь-який модуль зв'язку NGFF (Next Generation Form Factor). Вона має вбудовану програмну платформу, яка забезпечує прошивку, драйвери та додатки для швидкого запуску. Ця плата може створювати та тестувати системи на основі інструментів з відкритим кодом, таких як OpenOCD, GoogleTest та Robot Framework.

### 1.4.2 Системи зв’язку Інтернету речей

Система зв'язку – це сукупність ретрансляційних станцій, кінцевого обладнання даних (DTE), систем передачі та окремих мереж зв'язку, які зазвичай взаємопов'язані і взаємодіють один з одним, утворюючи єдине ціле [6]. Телекомунікації – це спосіб спілкування. Ключовими компонентами системи зв'язку є передавач, канал зв'язку та приймач [9]. Компоненти системи зв'язку мають загальне призначення, технічно сумісні, з використанням

зазвичай використовуваних процедур для відповіді на управління і спільної роботи [10]. Архітектура базової системи зв'язку показана на рисунку 1.7.

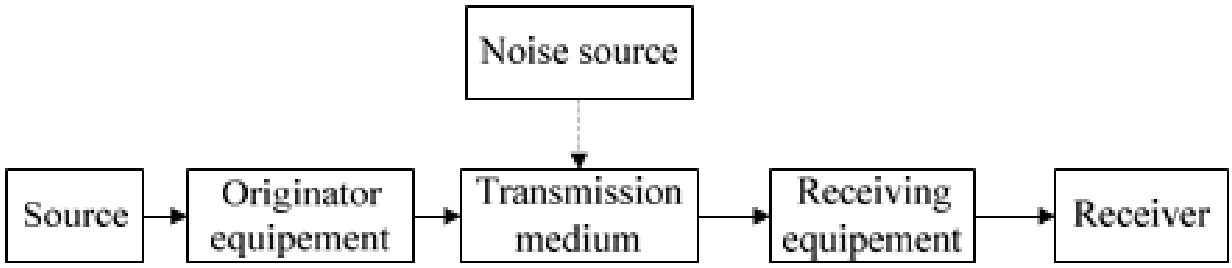



Рисунок 1.7 – Архітектура базової системи зв'язку IoT

### 1.4.3 WSN

WSN, іноді звані мережами бездротових датчиків і виконавчих механізмів (WSAN), являють собою автономні датчики, які розподілені в просторі для моніторингу умов навколишнього середовища або фізичних умов, таких як тиск, звук, температура і т. д., та для спільної передачі даних по мережі в бажане місце, де вони хочуть бути [11].

Дані пересилаються в приймач (іноді представлений як контролер або монітор), який використовує їх локально або підключений до інших мереж через шлюз, можливо, через кілька переходів [11]. Ці два типи сценаріїв показані на рисунку 1.8.

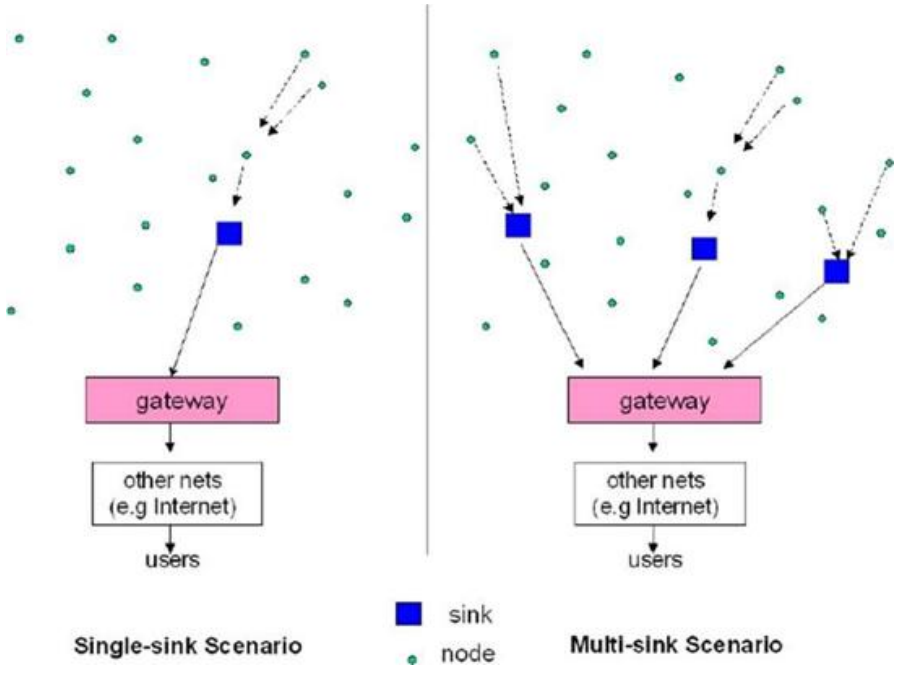



Рисунок 1.8 – Два сценарії передачі даних

## 1.5 Існуючі рішення в дослідженні і організації передачі даних IoT

Існує багато робіт, які використовують шлюзи IoT для зв'язку через системи зв'язку IoT. Більшість з них призначені для використання датчиків для збору даних, шлюзів IoT, управління виконавчими механізмами, збору даних з датчиків і пересилання даних в Інтернет і системи зв'язку IoT для передачі даних в додатки, не в останню чергу для вимірювання й оцінювання продуктивності.

### 1.5.1 Інтернет речей та хмарні обчислення в області медичних послуг

У цій статті [12] описується платформа на основі хмарних обчислень для управління переносними та мобільними датчиками охорони здоров'я.

Продемонстровано парадигми Інтернету речей стосовно охорони здоров'я.

Представлено пропоновану архітектуру системи, в якій датчики отримують інформацію про пацієнта, шлюз датчиків збирає дані з датчиків і пересилає їх в Інтернет, інтерфейси прикладного програмування (API) зв'язку, що використовуються шлюзом датчиків, відправляють дані датчиків і витягують інформацію, котрою управляє програма  та надає візуалізовану важливу інформацію і дані датчиків про стан пацієнта, а хмарна інфраструктура надає необхідні ресурси для розгортання інтерфейсів і веб-додатків для зв'язку з різними зовнішніми системами та датчиками. Потім реалізовано сенсорну частину і частину хмарної інфраструктури запропонованої архітектури в контексті простої системи моніторингу в реальному часі для її оцінки. Наведено комплексну оцінку системи.

### 1.5.2 Туманні обчислення та інтелектуальний шлюз на основі хмари речей

У роботі [13] описується архітектура розумного шлюзу з обчисленням туману, який може попередньо обробляти та обрізати дані перед відправкою в хмару, та тестує цю концепцію.

Представлена архітектура Smart Gateway з двома видами зв'язку, перший вид – це односторонній, в якому шлюз, який збирає дані та надсилає їх у Туман, а потім у хмару, безпосередньо підключений до датчиків та інших речей. А інший різновид – це мульти-стрибок, в якому підключаються кілька датчиків мереж та IoT, які мали б свої власні базові станції та поглинальні вузли, з яких шлюз збирає дані. Потім вводиться, що Fog Computing – це мережа, яка поширює традиційну парадигму хмарних обчислень до краю

мережі між основними хмарами та мережами. Нарешті, оцінюється ефективність зв'язку між хмарою та шлюзом.

### 1.5.2 Впровадження IoT для моніторингу стану навколишнього середовища в будинках

Стаття [14] представляє ефективну реалізацію IoT, яка використовується для моніторингу звичайних побутових умов за допомогою недорогих систем зондування.

Вони вперше описали рішення домашньої автоматизації, запропоноване в цій статті, включаючи ZigBee WSN, налаштований як Smart Sensing пристрої, і бездротовий зв'язок з координатором у вигляді топології сітки, маршрутизатор виконує роль шлюзу додатків IoT, який з'єднує мережу ZigBee та Інтернет-протокол мережа версії 6 (IPv6), яка використовує віртуальну приватну мережу (VPN), що використовується для підключення шлюзу додатків IoT до сервера та Інтернет-сервера на базі Windows, який збирає дані датчиків, надіслані з шлюзу додатків IoT, і зберігає їх у базі даних для подальшої обробки а потім переглянути його на веб-сайті. Потім встановили інтелектуальні датчики і встановили WSN на базі ZigBee у кількох будинках для тестування розробленої системи.

## 1.6 Висновки до розділу 1

Інформаційне суспільство важко уявити без надійних та швидких мереж зв’язку. Зв'язок на рівні комп’ютерів та пристроїв також вимагає вагомих критеріїв при його оцінці. В простих мережах виміряти та оцінити ці критерії на сьогодні не вимагає багато зусиль. Існують як програмні так і апаратні комплекси для вимірювання пропускної здатності, швидкості, часу реакції тощо. В цій роботі розглянуто передачу даних на платформах інтернету речей. Інтернет речей входить до переліку інноваційних напрямків XXI століття. Багато провідних корпорацій та співробітників наукових установ вважають, що інтернет речей це наступний виток в розвитку інформаційних технологій. Ця технологія допомагає усунути так званий «людський фактор», що є важливим в критичних системах, наприклад системах керування атомними станціями або літальними апаратами. Інтернет речей це взаємодія двох або більше машин без участі людини, що дозволяє використовувати ці технології в системах дійсного часу. Тому питання швидкості, безпечності передачі даних та дослідження цього аспекту є актуальним.

Передача даних в системах інтернету речей можлива в декількох варіаціях. Дані можна передавати з пристрою до пристрою за допомогою прямого з’єднання, але це спричиняє багато проблем, наприклад: ціна на

розгортання нової мережі, а якщо пристрої знаходяться за тисячі кілометрів один від одного, це вже стає майже не можливим. Передачу даних можливо виконувати через шлюзи, що скорочує вартість та розміри мережі для комунікації пристроїв, до одного шлюха можна підключити до 100 пристроїв. Але самим простим способом передачі даних в мережі інтернету речей – є інтернет або локальна мережа. До переваг можна віднести наявну інфраструктуру мережі, що приводить до мінімуму затрат на розгортання рішень інтернету речей. До недоліків можна віднести безпеку передачі даних та надійність.

До найбільш розповсюджених платформ інтернету речей можна віднести платформи на основі шлюзу інтернету речей та хмарних серверів. Найбільш популярна платформа на основі шлюзу інтернету речей це платформа Vinnter IoT. З рішеннями на базі хмари є багато нюансів, але найбільше популярні рішення це Azure IoT та Blumix IoT.

Відкриті програмні реалізації в області інтернету речей знаходяться тільки на початку зародження, однією з стабільних платформ є SensibleThings.

Проаналізувавши доступні механізми та засоби оцінки передачі даних на платформах IoT, було виявлено, що інструментарії стеження та оцінки для хмарних платформ існують та мають деякий функціонал, але не можуть повністю покрити питання якості зв’язку, наприклад пропускної здатності для всього рішення IoT та мають прихований механізм роботи, що не дає змогу дослідникам зрозуміти закономірності роботи рішення в цілому, що також підтверджує актуальність да доцільність проведення експериментального дослідження передачі даних в системі інтернету речей.

# 2 ПЛАНУВАННЯ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ ПЕРЕДАЧІ ДАНИХ В ІОТ СИСТЕМАХ

## 2.1 Основні теоретичні положення про етапи планування експерименту

Методи планування експерименту дозволяють мінімізувати число необхідних випробувань, встановити раціональний порядок і умови проведення досліджень в залежності від їх виду і необхідної точності результатів. Якщо ж з яких-небудь причин число випробувань вже обмежено, то методи дають оцінку точності, з якою в цьому випадку будуть отримані результати. Методи враховують випадковий характер розсіювання властивостей випробовуваних об'єктів і характеристик використовуваного обладнання. Вони базуються на методах теорії ймовірності та математичної статистики.

Планування експерименту включає ряд етапів.

Встановлення мети експерименту (визначення характеристик, властивостей і т.п.) та його виду (визначальні, контрольні, порівняльні, дослідницькі).

Уточнення умов проведення експерименту (наявне або доступне обладнання, терміни робіт, фінансові ресурси, чисельність і кадровий склад працівників і т.і.). Вибір виду випробувань (нормальні, прискорені, скорочені в умовах лабораторії, на стенді, полігонні, натурні або експлуатаційні).

Виявлення і вибір вхідних та вихідних параметрів на основі збору і аналізу попередньої (апріорної) інформації. Вхідні параметри (фактори) можуть бути детермінованими, тобто реєстрованими і керованими (залежними від спостерігача), і випадковими, тобто реєстрованими, але некерованими. Поряд з ними на стан досліджуваного об'єкта можуть впливати незареєстровані і некеровані параметри, які вносять систематичну або випадкову похибку в результати вимірювань. Це помилки вимірювального обладнання, зміна властивостей досліджуваного об'єкта в період експерименту, наприклад, через старіння матеріалу або його зносу, вплив персоналу і т. і.

Встановлення потрібної точності результатів вимірювань (вихідних параметрів), області можливої зміни вхідних параметрів, уточнення видів впливів. Вибирається вид зразків або досліджуваних об'єктів, враховуючи ступінь їх відповідності реальному виробу за станом, влаштуванню, формою, розмірами та іншими характеристиками.

На призначення ступеня точності впливають умови виготовлення і експлуатації об'єкта, при створенні якого будуть використовуватися ці експериментальні дані. Умови виготовлення, тобто можливості виробництва,

обмежують найвищу реально досяжну точність. Умови експлуатації, тобто умови забезпечення нормальної роботи об'єкта, визначають мінімальні вимоги до точності.

Точність експериментальних даних також істотно залежить від обсягу (кількості) випробувань – чим випробувань більше, тим (при тих же умовах) вище достовірність результатів.

Для ряду випадків (при невеликому числі факторів і відомому законі їх розподілу) можна заздалегідь розрахувати мінімально необхідну кількість випробувань, проведення яких дозволить отримати результати з необхідною точністю.

Складання плану та проведення експерименту – кількість і порядок випробувань, спосіб збору, зберігання і документування даних.

Порядок проведення випробувань важливий, якщо вхідні параметри (фактори) при дослідженні одного й того ж об'єкта протягом одного досліду приймають різні значення. Наприклад, при випробуванні на втому при ступінчастій зміні рівня навантаження межа витривалості залежить від послідовності навантаження, так як по-різному йде накопичення пошкоджень, і, виходячи з цього, буде різна величина межі витривалості.

У ряді випадків, коли систематично діючі параметри складно врахувати і проконтролювати, їх перетворюють в випадкові, спеціально передбачаючи випадковий порядок проведення випробувань (рандомізація експерименту). Це дозволяє застосовувати до аналізу результатів методи математичної теорії і статистики.

Порядок випробувань також важливий в процесі пошукових досліджень: в залежності від обраної послідовності дій при експериментальному пошуку оптимального співвідношення параметрів об'єкта або якогось процесу може знадобитися більше або менше дослідів. Ці експериментальні завдання подібні математичним завданням чисельного пошуку оптимальних рішень. Найкраще розроблені методи одновимірного пошуку (однофакторні однокритеріальні завдання), такі як метод Фібоначчі, метод золотого перетину.

Статистична обробка результатів експерименту, побудова математичної моделі поведінки досліджуваних характеристик.

Необхідність обробки викликана тим, що вибірковий аналіз окремих даних, поза зв'язком з іншими результатами, або ж некоректна їх обробка можуть не тільки знизити цінність практичних рекомендацій, а й призвести до помилкових висновків. Обробка результатів включає:

– визначення довірчого інтервалу середнього значення і дисперсії (або середнього квадратичного відхилення) величин вихідних параметрів (експериментальних даних) для заданої статистичної надійності;

– перевірка на відсутність помилкових значень (викидів), з метою виключення сумнівних результатів з подальшого аналізу. Проводиться на

відповідність одному зі спеціальних критеріїв, вибір якого залежить від закону розподілу випадкової величини та виду викиду;

– перевірка відповідності дослідних даних раніше апріорно введеному закону розподілу. Залежно від цього підтверджуються обраний план експерименту і методи обробки результатів, уточнюється вибір математичної моделі.

Побудова математичної моделі виконується у випадках, коли повинні бути отримані кількісні характеристики взаємопов'язаних вхідних і вихідних досліджуваних параметрів. Це завдання апроксимації, тобто вибору математичної залежності, яка найкраще відповідає експериментальним даним. Для цих цілей застосовують регресивні моделі, які засновані на розкладанні шуканої функції в ряд з утриманням одного (лінійна залежність, лінія регресії) або декількох (нелінійні залежності) членів розкладання (ряди Фур'є, Тейлора). Одним з методів підбору лінії регресії є широко поширений метод найменших квадратів.

Для оцінки ступеня взаємозв'язку факторів або вихідних параметрів проводять кореляційний аналіз результатів випробувань. В якості міри взаємозв'язку використовують коефіцієнт кореляції: для незалежних або нелінійно залежних випадкових величин він дорівнює або близький до нуля, а його близькість до одиниці свідчить про повний взаємозв'язок величин і наявності між ними лінійної залежності.

При обробці або використанні експериментальних даних, представлених в табличному вигляді, виникає потреба отримання проміжних значень. Для цього застосовують методи лінійної та нелінійної (поліноміальної) інтерполяції (визначення проміжних значень) і екстраполяції (визначення значень, що лежать поза інтервалом зміни даних).

Пояснення отриманих результатів та формулювання рекомендацій по їх використанню, уточнення методики проведення експерименту.

## 2.2 Мета експериментального дослідження

Для реалізації цілей, поставлений у цьому дослідженні, необхідно провести наступні кроки.

Крок 1. Запропонувати сценарій, в якому шлюз IoT підключається до виконавчого механізму для управління виконавчим механізмом та передачі даних через систему зв'язку IoT. Буде запропоновано сценарій, в якому платформа Vinnter підключається до виконавчого механізму, з метою керування приводом та передачі даних через платформу SensibleThings. Виконавчі механізми можуть бути світлодіодними (LED), і клієнт буде контролювати стан світлодіода. І дані, отримані з платформи Vinnter, будуть відображатися в клієнті приймача.

Крок 2. Створення демонстратора шляхом настройки зв'язку між платформою шлюзу IoT і системою зв'язку IoT, створюємо середу для MyEclipse для запуску програми. А потім за допомогою мови Java створимо демонстратор з двома проектами Java для реалізації мети управління приводом та передачі даних через систему зв'язку.

Крок 3. Вимірювання та оцінка продуктивності з точки зору затримки і пропускної здатності з використанням реалізованого застосунку. Проведемо експеримент за допомогою програми, щоб виміряти затримку і пропускна здатність передачі як послідовного зв'язку, так і всієї системи. Оцінемо продуктивність, обчисливши середнє значення і стандартне відхилення затримки і пропускної здатності, виміряних раніше, щоб побачити, чи достатньо швидко вони задовольняють вимогу і чи є пропускна здатність достатньою.

Крок 4. Перевіримо, чи достатньо надійна передача між джерелом і приймачем. Потім перевіримо, чи менше затримка, ніж в більшості інших ситуацій, і вище пропускна здатність, ніж у інших. Оцінимо практичну значимість дипломної роботи.

Крок 5. Оцінка всієї роботи. Наскільки платформа Vinnter, задовільнили все вимоги експерименту. Або треба розширити кількість платформ для проведення дослідження. Порівняємо, чи є інструментарій достатнім вибором для вимірювання та оцінки затримки і пропускної здатності. Загальний висновок на основі даних проведеного експерименту.

## 2.3 Сценарії проведення експериментального дослідження передачі даних в IOT системах

Сценарії, які використовуються у цій роботі, платформа Vinnter підключена до світлодіоду, щоб контролювати стан світлодіода. Платформа Vinnter буде підключена до системи зв'язку, щоб контролювати стан світлодіода на клієнті і передавати отримані дані назад клієнту через систему зв'язку.

### 2.3.1 Вибір варіантів зв’язку шлюзу Інтернета речей

Розглянем три варіанти зв’язку через шлюз IoT, а саме послідовний зв'язок, UDP та Bluetooth.

Послідовний зв'язок – це процес передачі даних послідовно по одному біту за раз по комп'ютерній шині або каналу зв'язку при передачі даних і телекомунікації, що протилежно паралельної зв'язку, коли кілька бітів передаються по каналу з декількома паралельними каналами в цілому [15]. Він в основному використовується для комп'ютерних мереж і всіх міжміських

комунікацій, де труднощі з синхронізацією і вартість кабелю роблять паралельний зв'язок затратною технологією, й навіть на більш коротких відстанях послідовний зв'язок також стає все більш поширеною технологією. Приклади паралельного інтерфейсу та послідовного інтерфейсу (спочатку старший біт) представлені на рисунку 2.1.

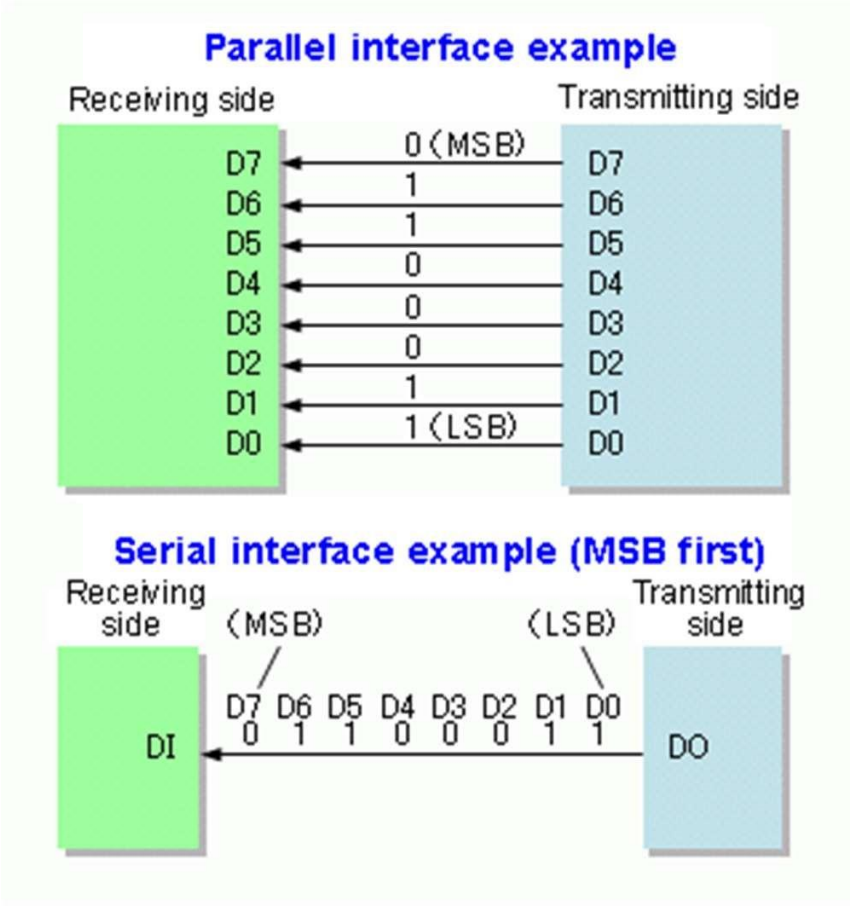



Рисунок 2.1 – Паралельний та послідовний інтерфейс зв’язку

Щоб задовольнити особливі потреби вбудованих систем, були розроблені послідовні протоколи. Ethernet, межінтегральная схема (I2C) та USB. Це дуже поширені послідовні інтерфейси, які можна розділити на дві групи: синхронні, які з'єднують свою лінію (лінії) даних з тактовим сигналом, щоб всі пристрої синхронної послідовної шини могли спільно використовувати звичайний годинник або асинхронні, передача даних яких не підтримується зовнішнім синхросигналом [16]. Якщо обидва пристрої на послідовній шині не налаштовані на використання одних й тих же протоколів, дані можуть бути відправлені послідовно через множину цих механізмів сигналізації.

Існує три режими послідовного зв'язку: симплекс, в якому дані можуть передаватися тільки від передавача до приймача, а не навпаки, полудуплекс, в якому дані можуть передаватися тільки в одному напрямку за раз, або від передавача до приймача, або від приймача. на передавач, але не на обидва, і в

повнодуплексному режимі, в якому дані можуть передаватися одночасно від передавача до приймача і приймача до передавача [17]. На рисунку 2.2 наведено три режими послідовного зв'язку.

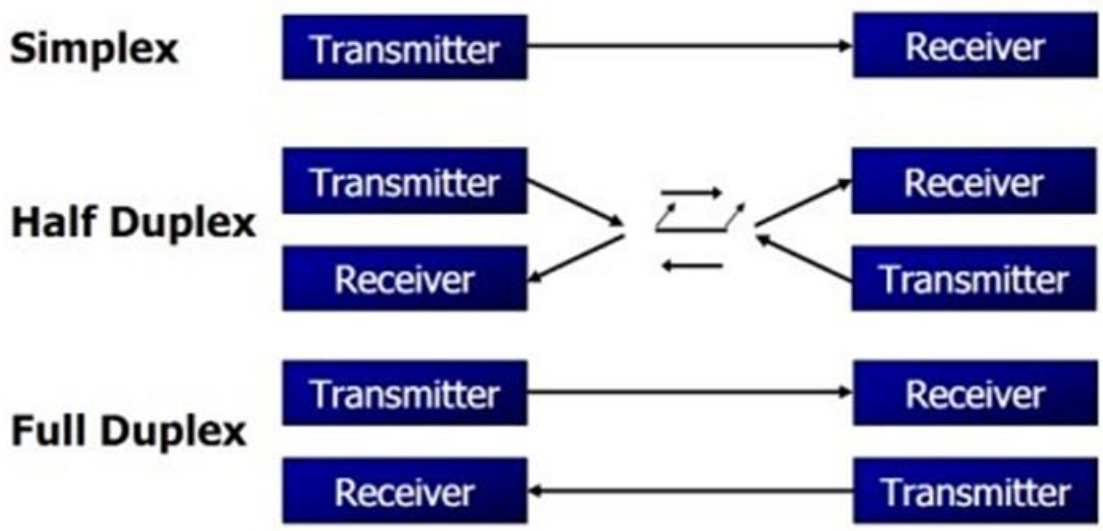



Рисунок 2.2 – Режими послідовного зв’язку

В електронному зв'язку протокол призначених для користувача дейтаграм (UDP) є одним із основних протоколів із набору інтернет-протоколів (IP) на транспортному рівні, за допомогою яких комп'ютерні програми можуть надсилати повідомлення, так звані дейтаграми в цьому випадку на інші хости в IP-мережі без попередньої зв'язку для настройки шляхів даних або каналів передачі [18].

UDP використовує просту модель передачі без встановлення з'єднання з мінімальним механізмом протоколу. Він має номера портів для обробки різних функцій в джерелі і одержувачі дейтаграми та контрольних сум, щоб гарантувати цілісність даних. Без встановлення зв'язку він піддає програму користувача проблемам базової мережі: таким чином, не гарантує передачу, доставку і захист від дублювання. Якщо на рівні мережного інтерфейсу потрібний засіб виправлення помилок, застосунок може використовувати протокол передачі управління потоком (SCTP) або протокол управління передачею (TCP), які призначені для цієї мети, щоб замінити UDP.

Протокол UDP підходить для тих випадків, коли перевірка і виправлення помилок виконується на прикладному рівні, або не виконується взагалі. Тому що це дозволяє уникнути накладних витрат на цю обробку на рівні мережевого інтерфейсу. В результаті чутливі до часу програми зазвичай використовують UDP, тому що відкидання пакетів перевершує очікування затриманих пакетів, що може бути не дуже хорошою ідеєю в системах реального часу.

Пакет UDP викликається як користувацька дейтаграмма з 8-байтовий заголовком, формат якої показаний на рисунку 2.3, в якому перші 8 байтів містять інформацію заголовка, що включає номер порту джерела, номер порту призначення, загальну довжину та поле контрольної суми, кожне з яких становить 2 байта, а інші байти містять дані [19].

У UDP-з'єднанні клієнт встановлює унікальний номер вихідного порту на основі програми, з якою він запускає з'єднання. Протокол UDP не обмежується взаємодією «один-до-одного», але за допомогою широкомовної або багатоадресної адресації, також може бути досягнуто взаємодія «один-до-багатьох», через множину клієнтів, взаємодіючих з одним сервером, взаємодія «багато-до-одного» може бути досягнуто й через розширення цих методів, що дозволяє також досягти взаємодії «багато до багатьох».

Bluetooth – це стандарт бездротової технології для обміну даними на малих відстанях через бездротовий зв'язок 2,4 ГГц для недорогого обладнання та створення персональних мереж (PAN), який спочатку розглядався як бездротова альтернатива кабелям даних RS-232 [20]. Bluetooth – це протокол на основі пакетів зі структурою провідний-ведений, коли один ведучий може обмінюватися даними з сімома відомими пристроями в пікомережі, яка представляє собою мережу, що самоорганізується, яка зв'язує групу бездротових користувачів з використанням протоколів технології Bluetooth з усіма пристроями, спільно використовують батьківський годинник. Провідний пристрій координує зв'язок по всій пікомережі та може відправляти дані будь-якого зі своїх ведених пристроїв, а також запитувати дані від них, в той час як ведені пристрої можуть бути відправлені й отримані тільки від свого ведучого пристрою і не можуть розмовляти з іншими відомими пристроями в пікомережі [21]. Приклад пікомережі наведено на рисунку 2.4.

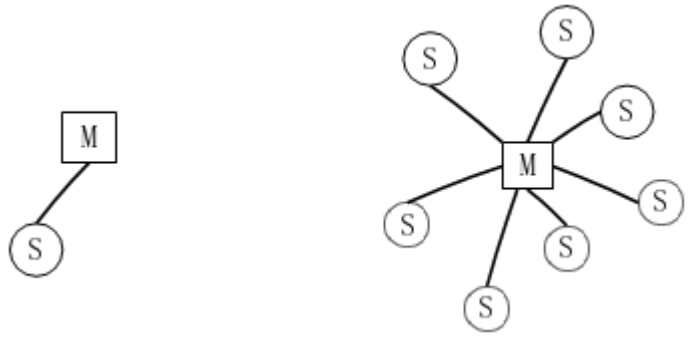



Рисунок 2.4 – Приклади топологій пікомережі Bluetooth master / slave

Bluetooth визначається як архітектура протоколу рівня, яка складається з прийнятих протоколів, протоколів управління телефонією, протоколів заміни кабелю і основних протоколів, обов'язкові протоколи для всіх стеків Bluetooth: SDP (протокол виявлення послуг), L2CAP (протокол управління і адаптації

логічних каналів) і LMP (протокол диспетчера каналів) [20]. Стек протоколів Bluetooth показаний на рисунку 2.5.

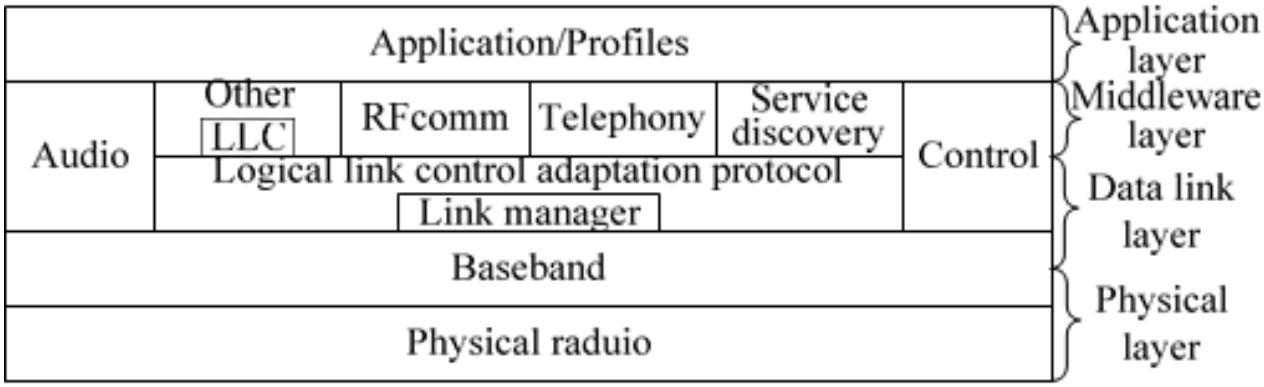



Рисунок 2.5 – Стек протоколів Bluetooth

### 2.3.2 Системи зв’язку IoT

Для систем зв'язку IoT існує три поширених платформи: це SensibleThings, Microsoft Azure та Kaa.

Платформа SensibleThings – це платформа, яка створює швидкі і ефективні програми Інтернету речей і з'єднує виконавчі механізми і датчики з різними типами додатків [22]. Платформа надає середовище з відкритим вихідним кодом для з'єднання виконавчих механізмів і датчиків разом для розширюваних контекстно-залежних додатків в реальному часі [23]. Її можна використовувати в різних сценаріях, таких як охорона здоров'я, соціальні програми, відстеження об'єктів, інтелектуальна домашньої автоматизації і так далі.

Платформа SensibleThings може бути розділена на вхідні компоненти, а різні рівні включають рівень датчика / виконавчого механізму, мережевий рівень, рівень поширення, рівень надбудови і рівень інтерфейсу. Це показано на рисунку 2.6. Шар датчиків і виконавчих механізмів може підключати різні виконавчі механізми і датчики до платформи. Мережевий рівень може з'єднувати різні об'єкти через поточну інфраструктуру, засновану на IP. Рівень поширення може передавати інформацію між усіма об'єктами, які підключені до платформи і беруть участь в системі. Шар надбудови може допомогти розробникам додавати в платформу алгоритми оптимізації і додаткові функції. І, нарешті, рівень інтерфейсу – це публічний інтерфейс, який дозволяє додаткам взаємодіяти з платформою SensibleThings [24].

Зокрема, платформа SensibleThings заснована на наступних технічних принципах: вона має зв'язок на основі IP.

В цей час Інтернет, повністю розподілена архітектура DHT (розподілена хеш-таблиця), за допомогою якої можна досягти глобального маштабування,

P2P (тимчасового) зв'язку, яка відправляє дані в першому пакеті, методи проникнення NAT (трансляція мережевих адрес), з допомогою яких можна досягти безшовного підключення та полегшена реалізація для роботи на обмежених пристроях [22]. Просто використовуючи шар інтерфейсу, ви можете створити простий додаток.

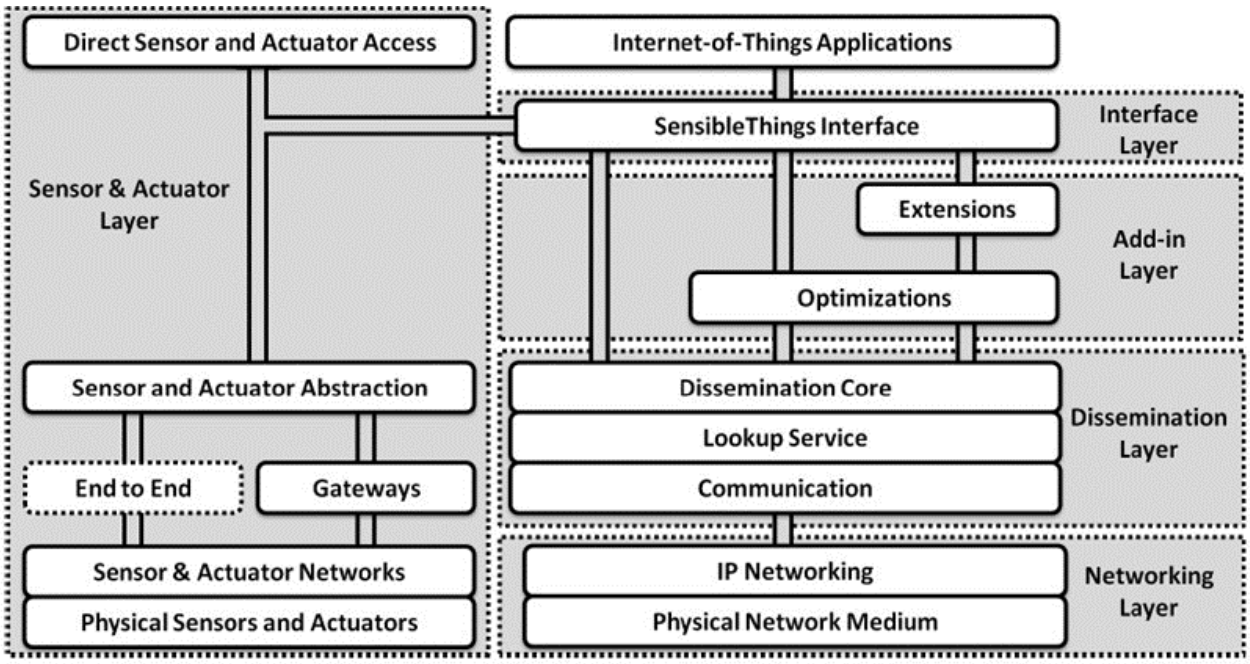



Рисунок 2.6 – Архітектура платформи SensibleThings

Microsoft Azure – це зростаючий набір інтегрованих хмарних сервісів, створених Microsoft, які розробники та фахівці з IT (інтернет-технологій) використовують для створення, розгортання та управління додатками і сервісами в глобальній мережі центрів обробки даних, керованих Microsoft [25].

Як показано на рисунку 2.7, компоненти платформи служб Azure можуть використовуватися додатками, що працюють в локальних або хмарних системах, в тому числі [26]:

1) Windows Azure: надає платформу для запуску додатків Windows та зберігання даних на великій кількості серверів в центрах обробки даних Microsoft;

2) Microsoft .NET Services: пропонує послуги розподіленої інфраструктури для програм, які потребують хмарні обчислення або виконуються локально;

3) Microsoft SQL (мова структурованих запитів): надає служби даних на основі SQL Server в хмарі;

4) Live Services: забезпечує доступ до даних з додатків Microsoft Live та інших додатків через Live Framework.

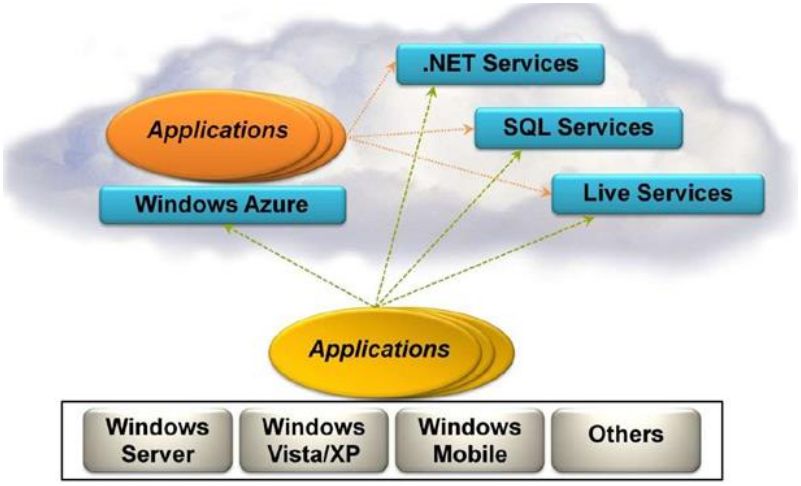



Рисунок 2.7 – Додатки, які підтримуються платформою служб Azure

Kaa – це багатофункціональна платформа проміжного програмного забезпечення для Інтернету речей, яка створює повні наскрізні рішення Інтернету речей, поєднуючи інтелектуальні продукти та додатки [27]. Вона надає відкритий інструментарій для розробки продукту, який значно знижує пов'язані ризики, витрати і час виведення продукту на ринок.

Kaa управляє даними внутрішньої інфраструктури і підключених об'єктів, надаючи SDK кінцевої точки (комплект розробки програмного забезпечення) і серверні компоненти. Kaa SDK вбудований практично в будь-який тип підключеного пристрою і забезпечує двонаправлений обмін даними в режимі реального часу з сервером для забезпечення всіх внутрішніх функцій, необхідних для роботи рішення IoT [27].

Кластер Kaa повинен використовувати екземпляри баз даних NoSQL і SQL та вузли Kaa, які використовують Apache ZooKeeper для координації таких служб, як Bootstrap, Operations і Control, які адміністратори Kaa можуть індивідуально включати або відключати [28]. Високорівнева архітектура Kaa наведена на рисунку 2.8.

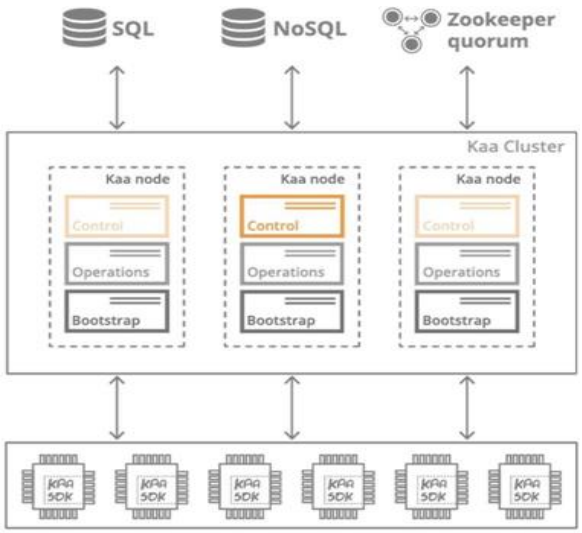



Рисунок 2.8 – Архітектура високого рівня Kaa

## 2.4 Програмне забезпечення для експериментального дослідження передачі даних на IoT платформах

Хоча є кілька альтернатив для зв'язку через шлюз IoT та систем зв'язку IoT. Для реалізації програмного забезпечення використано тільки послідовний зв'язок та платформу SensibleThings.

У цій дипломній роботі мета полягає в тому, щоб встановити зв'язок між обладнанням та програмним забезпеченням IoT та перевірити якість зв’язку та оцінити параметри, які на нього впливають. Створено два Java-проекту, приймач та джерело. Проект джерела – це клієнтська програма. Вона відправить команду початкового проекту. Джерело відправить команду на платформу Vinnter після отримання команди. Платформа Vinnter буде діяти по команді та отримає відповідь. Джерело прочитає відповідь, яку було надіслано і відправить її в приймач. В результаті буде встановлено зв'язок між програмним забезпеченням Iot та обладнанням. Загальна блок-схема показана на рисунку 2.9.

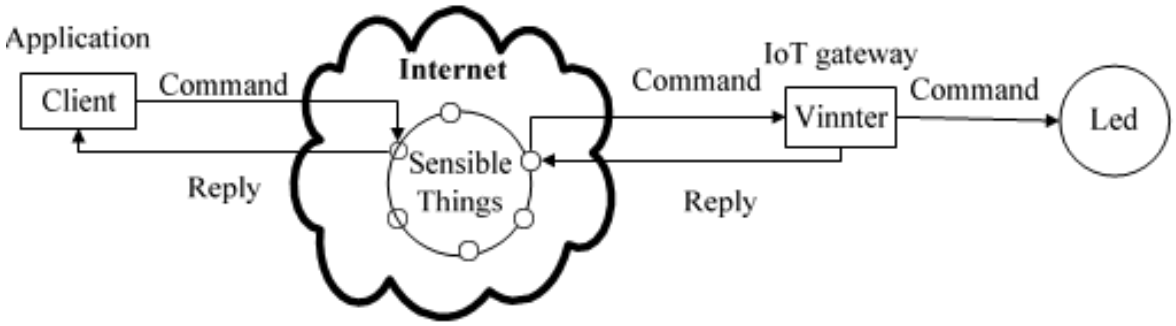



Рисунок 2.9 – Загальна блок схема передачі даних

Рисунок 2.9 дає тільки загальне представлення блок-схеми роботи системи. Для реалізації запропонованих цілей потрібна більш детальна інформація про реалізацію, яка наведена на рисунку 2.10.

Алгоритм роботи системи наступний:

1. Вихідний вузол реєструє універсальні ідентифікатори контексту (UCI).

2. Вузол прийому дозволяє UCI вихідного вузла, який запускає приймач відповіді дозволу.

3. Sink Node отримує відповідь у функції ResolveResponse.

4. Вузол Sink викликає функцію Set для установки значення приводу, який запускає функцію SetEvent в вузлі джерела.

5. Вихідний вузол отримує задане значення у функції SetEvent.

6. Вихідний вузол записує значення на платформу Vinnter.

7. Платформа Vinnter перемикає стан світлодіода.

8. Вихідний вузол зчитує отримані дані з платформи Vinnter.

9. Вузол джерела викликає функцію повідомлення, щоб відправити отримане значення, яке викличе функцію GetResponse в вузлі приймача.

10. Sink Node отримує дані відповіді у функції GetResponse.

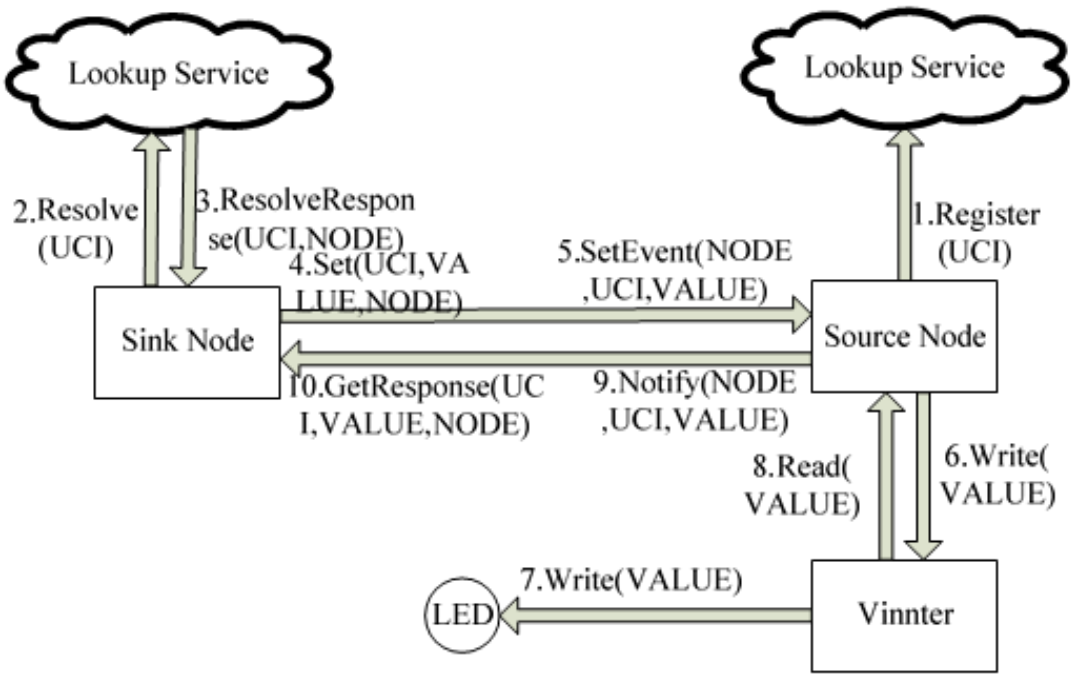



Рисунок 2.10 – Деталізована блок-схема

Знадобиться PuTTy, програму для rlogin, SSH (Secure Shell), Telnet, підключення через послідовний інтерфейс, установник драйверів та платформа Vinnter. Використовуючи PuTTy, отримуємо номер послідовного порту, який використовує платформа Vinnter, який є COM3, швидкість передачі даних, для якої підходить послідовний порт платформи Vinnter, і

деякі інші параметри послідовного порту платформи Vinnter. Команди платформи Vinnter наведено на рисунку 2.11.

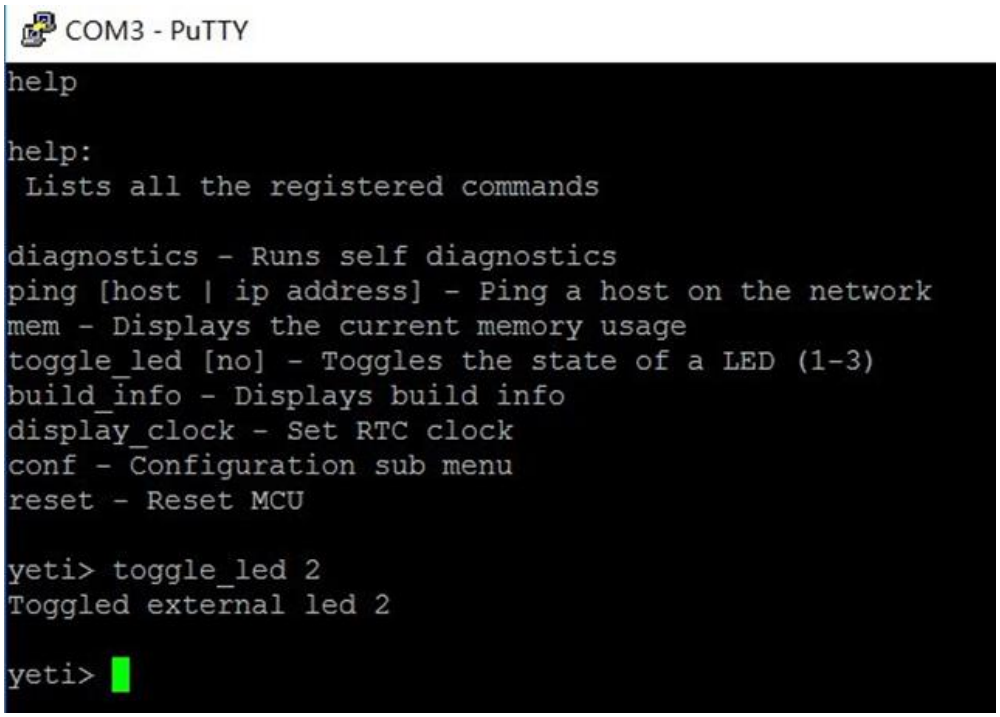



Рисунок 2.11 – Команди платформи Vinnter

Можливо перемикати стан світлодіода. Підключаємо світлодіод до одного з інтерфейсів платформи Vinnter та переключаємо світлодіод, щоб визначити, правильне з'єднання чи ні.

### 2.4.1 Впровадження послідовного зв’язку в IoT

Для послідовного інтерфейсу в Java потрібен API послідовного зв'язку, званий JavaComm. На жаль, поточна версія JavaComm не підтримує Windows. Але, є RXTX, сумісний з реалізацією API пакету послідовного зв'язку javax.comm. Таким чином, щоб викликати метод зв'язку через послідовний порт за допомогою Java, мені потрібно завантажити драйвер послідовного порту. Там буде три файли: RXTXcomm.jar, rxtxParallel.dll і rxtxSerial.dll. Створіть папку в проекті та скопіюйте ці три файли в цю папку. Після цього побудуйте шлях до Java, додайте в проект RXTXcomm.jar і задайте розташування власної бібліотеки. Після імпорту gnu.io. * В проект можливо використовувати послідовний зв'язок в Java.

Потрібно спочатку створити проект Java, а потім оголосити клас з ім'ям Vinnter.java, який реалізує клас SerialPortEventListener, що є класом в RXTX, необхідним при читанні вхідних даних та розширює потік, щоб дозволити

потоку зависнути при необхідності. Щоб реалізувати послідовний зв'язок між платформою Vinnter та додатком виконаємо наступні кроки:

1) знайдіть потрібний послідовний порт;

2) налаштуйте підключення до послідовного порту;

3) запустіть потік вводу-виводу  та прослуховувач подій.

4) запис в послідовний порт;

5) зчитування з послідовного порту;

6) завершіть роботу.

Крок 1. Створено функцію, яка відображає всі доступні порти для JavaComm, використовуючи доступні функції, а потім перевіряє, чи є тип порту послідовним, щоб вибрати доступний послідовний порт.

Крок 2. Створено функцію, отримання ім'я класу, створення послідовного пору та його відкриття. У разі успіху порт буде збережений,

Крок 3. Створено функцію для ініціалізації вхідного та вихідного потоків відкритого послідовного порту, для зчитування та запису в послідовний порт зв’язку. У разі невдачі генерується виключення IOException. Розробленно функцію для додавання та прослуховувач подій, щоб перевірити, чи є серійною подія. У разі невдачі буде видано виняток, що слухачів занадто багато. Встановіть режим прослуховування, щоб розбудити потік прослуховування під час надходження даних.

Крок 4. Розроблено функцію для запису даних. У функції є масив байтів для зберігання байтів даних, необхідних для запису.

Крок 5. Розроблено функцію для зчитування даних та їх повернення. Ця функція знаходиться відразу після функції запису, щоб отримувати дані, коли операція запису виконується на платформі Vinnter, і відправляти дані відповіді назад в буфер.

Крок 6. Після завершення всіх комунікацій послідовний порт повинен бути відключений, тому що деякі порти можуть залишатися відкритими.

Для реалізації функцій комунікації створено два проекти: вихідний проект та проект-приймач. Після встановлення зв'язку команди та відповідь можуть передаватися між джерелом і приймачем. І вихідна програма викличе додаток Vinnter для відправки команд та отримання даних. В результаті інтеграція апаратних і програмних платформ IoT завершена.

### 2.4.2 Реалізація SensibleThing

Щоб використовувати платформу SensibleThings, потрібно завантажити комплект розробника SensibleThings та додати SensibleThingsBeta6.jar як в оригінальну програму, так і в додаток-приймач.

SensibleThings організовано в кільцеву структуру, схожу на Chord DHT. Вузлом може бути множин речей, наприклад мобільний телефон, комп'ютер і

т. д. Коли новий вузол потрібно приєднатися до кільця та спілкуватися з іншими, це можна виконати за чотири кроки, показаних на рисунку 2.12.

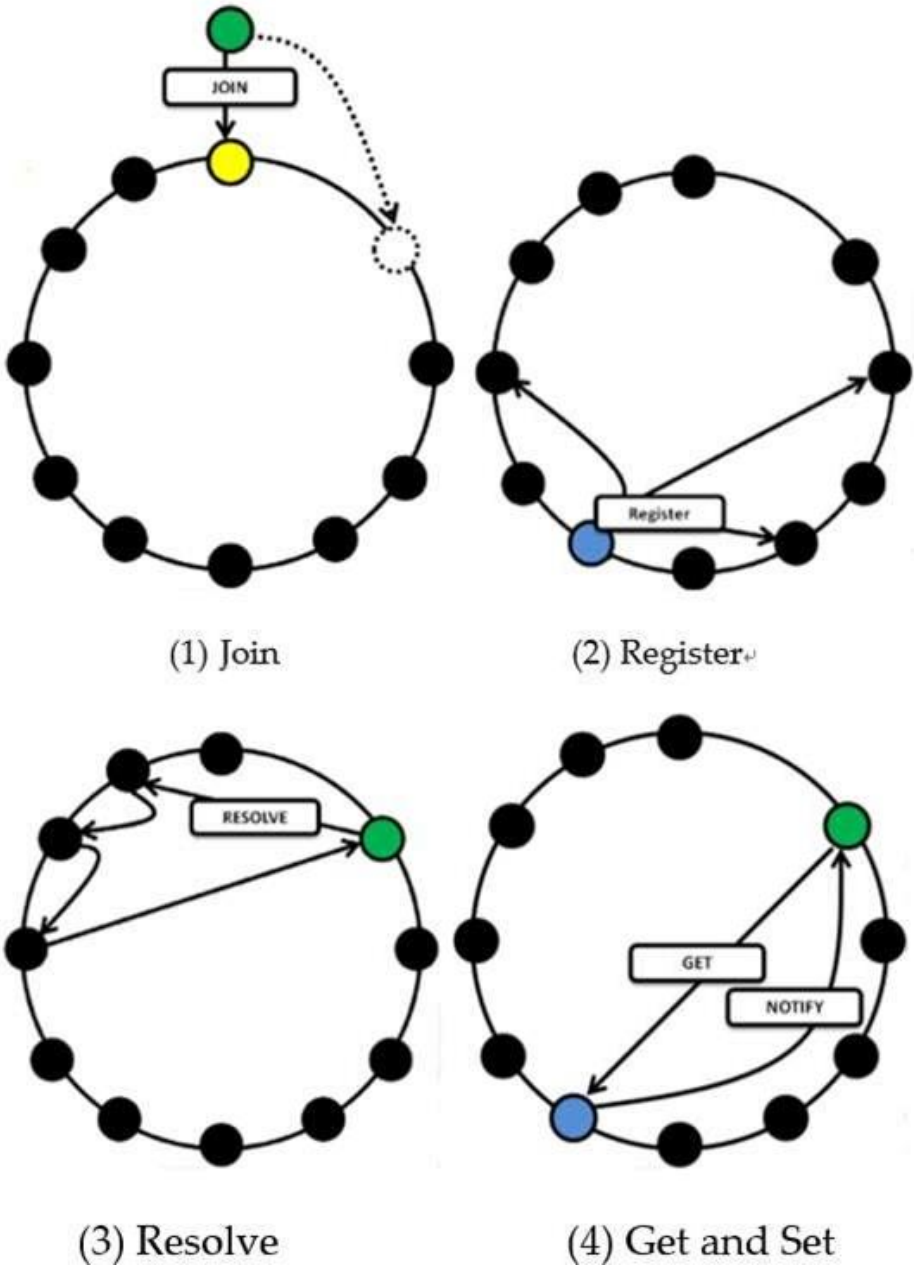



Рисунок 2.12 – Робота з кільцем SensibleThings

Новий вузол може приєднатися до кільця SensibleThings, підключившись до будь-якого іншого вузла в цьому кільці. На щастя, для стабільності та зручності завжди працює вузол, званий вузлом початкового завантаження. Так що в будь-який момент вузол може приєднатися до кільця.

Можливо зареєструвати UCI, який схожий на адресу електронної пошти та URL-адресу (універсальний покажчик ресурсів), об'єднані в одне ціле для датчика або виконавчого механізму. Завдяки UCI весь DHT знає, який UCI якому вузлу належить.

Після реєстрації будь-який інший вузол в кільці може дозволити UCI, щоб знайти вузол з конкретним виконавчим механізмом або датчиком. І метод reslolve запустить функцію resolveResponse для відправки назад адреси вузла, перекладеного з UCI за допомогою DHT. Це схоже на звичайний DNS (система доменних імен), але тут немає централізованого сервера. Це просто однорангова технологія.

Після того, як вузол дізнається, який інший вузол володіє UCI, він може встановити спеціальний робочий сеанс з іншим вузлом для установки виконавчих механізмів або отримання значень датчиків. Тут необхідні примітиви відповідно SET та GET. Можливо виконувати дії GET та SET в функції resolveResponse після успішного вирішення. Дії SET викличуть функцію setEvent на іншій стороні. І після того, як інший вузол отримав повідомлення set, він може встановити і відправити значення назад вузлу-джерелу, викликавши функцію повідомлення. Функція повідомлення викличе функцію getResponse. Аналогічним чином можна використовувати примітив GET.

Налаштувати з'єднання. Потрібно спочатку створити проект Java, пакет і оголосити клас, який повинен реалізувати SensibleThingsListener, додати SensibleThingsBeta6.jar в додаток та імпортувати se.sensiblethings.interfacelayer. *, Щоб використовувати платформу SensibleThings та отримати зворотний виклик функції. Потім створити платформу SensibleThingsPlatform та запустити її за допомогою конструктора. Він автоматично згенерує чотири функції, деякі з яких будуть використані.

Також можливо налаштувати з'єднання, використовуючи об'єкт платформи Vinnter. Створити об'єкт платформи Vinnter для виклику функції в Vinnter.java. Реалізувати у функції запуску реєстрацію UCI – «project@miun.se / actator». Після цього він буде доступний для доступу з всіх вузлів системи. Також викликати функцію в Vinnter.java для пошуку послідовного порту, настройки послідовної зв'язку, ініціалізації потоку введення-виведення та додавання прослуховувачів подій. Створенно BufferedReader для зчитування з клавіатури, щоб закрити всі завдання та відключитися від порту.

Потрібно переписати функцію setEvent, щоб вона діяла як відповідь на функцію set з приймача. І в цій функції спочатку викликати функцію запису в Vinnter.java для запису даних з приймача на платформу Vinnter, а потім викликати функцію зчитування в Vinnter.java. Потім викликати функцію повідомлення, щоб відправити у відповідь дані в приймач.

Після того, як все це було зроблено. Завершено інтеграцію апаратно-програмної платформи IoT. Потім можна виконати вимір та оцінку

## 2.6 Методи обробки результатів

### 2.6.1 Попередня обробка даних

Методи обробки даних спостережень базуються на положеннях теорії ймовірності та математичної статистики. Попередня обробка результатів вимірювань або спостережень необхідна для того, щоб в подальшому з найбільшою ефективністю, а головне коректно, використовувати для побудови емпіричних залежностей статистичні методи.

Математичною статистикою називається наука, що займається розробкою методів отримання, опису і обробки експериментальних даних з метою вивчення закономірностей випадкових масових явищ.

Всі задачі математичної статистики умовно можна розчленити на дві групи.

Першою з них є розробка методів збору і групування статистичних даних, отриманих в результаті спостережень, опрацювання статистичних звітів чи даних в результаті спеціально поставлених експериментів.

Друга задача полягає в розробці методів аналізу статистичних даних залежно від мети. Сюди належать:

а) оцінка ймовірності події; знаходження функції розподілу випадкової величини; оцінка залежності випадкової величини від інших випадкових величин, тощо; оцінка невідомих параметрів розподілу;

б) перевірка статистичних гіпотез про зроблені вище припущення.

Висновки за допомогою методів математичної статистики, зроблені зі зібраних статистичних даних, повинні правильно відображати загальні ймовірнісні характеристики процесу, що досліджується.

Для кожної характеристики з набору значень передбачається отримувати основні статистичні характеристики.

Мінімальне значення характеристики.

Максимальне значення характеристики.

Математичне сподівання (МС), яке характеризує середнє зважене значення випадкової величини, визначається для того, щоб отримати характеристику цієї ознаки для всієї досліджуваної групи в цілому:

$$M = \frac{1}{n}\sum_{i=1}^{n} x_i. \qquad (2.1)$$

де M – математичне сподівання,

n – обсяг вибірки,

$x_i$ – окремі значення випадкової величини.

Дисперсію, яка характеризує математичне сподівання квадрата відхилення випадкової величини від її математичного очікування:

$$D_b = \frac{\sum_{i=1}^{k} n_i \cdot (x_i - \bar{x}_b)^2}{n}; \quad (2.2)$$

де $D_b$ – вибіркова дисперсія, розрахована за даними спостережень,

$x_i$ – окремі значення,

$x_b$ – середнє арифметичне вибірки.

Середнє квадратичне відхилення величин вихідних параметрів можна обчислити як:

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2. \quad (2.3)$$

де $S^2$ – середнє квадратичне відхилення, незміщена оцінка средньоквадратичного відхилення випадкової величин X відносно її математичного сподівання,

$x_i$ – і-ий елемент вибірки,

$\bar{x}$ – середнє арифметичне вибірки,

n – розмір вибірки.

Чим вище дисперсія або стандартне відхилення, тим сильніше розкидані значення змінної відносно середнього.

Середнє арифметичне – найбільш поширена оцінка середнього значення розподілу. Вона є результатом ділення суми всіх спостережуваних числових величин на їх кількість. Для вибірки, що складається з чисел $X_1, X_2, \ldots, X_n$, вибіркове середнє (позначається символом x $\bar{X}$) та обчислюється як:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}, \quad (2.4)$$

де $\bar{X}$ – вибіркове середнє,

n – об’єм вибірки,

$X_i$ – і-й елемент вибірки.

**Відсів грубих помилок вимірювань.** Кожен результат вимірювання – випадкова величина. Відхилення результату спостереження від істинного називається помилкою спостереження. Помилка спостереження також випадкова величина, вона є результатом дії тільки випадкових факторів, та визначається як:

$$\Delta X = X - a, \tag{2.5}$$

де ΔX – помилка спостереження,

X – результат вимірювань,

a – істинний результат.

Розрізняють помилки трьох видів:

– грубі помилки виникають внаслідок порушення основних умов вимірювання. Результат, який містить грубу помилку, різко відрізняється за величиною від інших вимірів;

– систематичні помилки, які є постійними у всій серії вимірювань або змінюються за певним законом. Вони можуть бути усунені введенням відповідних поправок в результати вимірювань;

– випадкові помилки – помилки вимірювання, що залишаються після усунення всіх виявлених грубих і систематичних помилок. Вони викликаються великою кількістю таких факторів, ефекти дії яких настільки незначні, що їх не можна виділити окремо.

При обсязі вибірки n> 25 може бути використаний метод, заснований на застосуванні розподілу Стьюдента.

Послідовність дій при використанні даного методу наступна:

1) З ряду спостережень (вимірювань) вибирається спостереження, що має найбільше відхилення (позитивне чи негативне):

$$d_{\max} = \left| x_{\max(\min)} - \bar{x} \right|. \tag{2.6}$$

2) Обчислюється максимальне відносне відхилення:

$$\tau = d_{\max} / \bar{S}. \tag{2.7}$$

3) По таблиці розподілу Стьюдента знаходяться процентні точки t-розподілу Стьюдента t (p, n - 2), де p - процентна точка нормованого вибіркового відхилення. Приймають дві точки p = 5% і p = 0,1%.

4) Обчислюється критичне значення відносного відхилення p, яке виражається через критичне значення розподілу Стьюдента t (p, n - 2) за формулою:

$$\tau_{\{p,n\}} = \frac{t(p,n-2)\sqrt{n-1}}{\sqrt{n-2+\left[t_{(p,n-2)}\right]^2}}. \qquad (2.8)$$

де $\tau_{(p,n)}$ – критичне значення розподілу Стьюдента,

t (p, n - 2) – процентні точки t-розподілу Стьюдента.

5) Порівнюється значення τ з обчисленими критичними значеннями τ (5%, n) та τ (0,1%, n).

Максимальні відносні відхилення, отримані в процесі обчислення, можуть бути розділені на три групи:

1) $\tau \le \tau$ (5%, n);

2) $\tau$ (5%,n) $\le \tau \le \tau$ (0,1%, n);

3) $\tau \ge \tau$ (0,1%, n).

Спостереження, що потрапили в першу групу, не можна відсівати ні в якому разі. Спостереження другої групи можна відсіяти, якщо тільки є які-небудь інші дані на користь цієї процедури.

Спостереження третьої групи, як правило, відсівають завжди.

Після виключення того чи іншого спостереження характеристики емпіричного розподілу повинні бути перераховані за даними скороченої вибірки. Після чого повторюють процедуру перевірки для наступного за абсолютною величиною найбільшого відхилення dmax.

### 2.6.2 Метод вимірювання та оцінки послідовного зв’язку в IoT системі

Створити об'єкт класу, щоб викликати функції цього класу. Для цього викликати функцію, щоб дізнатися послідовний порт COM3, підключитися до порту, відформатувати потоки введення і виведення і додати прослуховувач подій. Отже, тепер з'єднання встановлено успішно, тепер можливо писати і читати дані. Виконати три етапами.

Етап 1. Протестуйте функції запису і читання.

Етап 2. Виміряти та оцінити затримку.

Етап 3. Виміряти та оцінити пропускну здатність.

Щоб перевірити правильність функції запису та читання. Напишемо «help \ r \ n» платформі Vinnter та прочитаємо отримані дані. Після цього викликаємо функцію сну, щоб зробити паузу на 500 мілісекунд для читання даних. З попередньої операції в PuTTy відомо, що можна перемикати стан світлодіода 2. Тому відаємо команду «toggle_led 2 \ r \ n» платформі Vinnter.

Якщо функція запису та читання правильні. Потім вимірюємо та оцінюємо затримку й пропускну здатність з'єднання.

Вимірювання затримки відбувається за формулою (2.9). В якій час початку ($start_k$) – це час безпосередньо перед функцією запису для запису «toggle_led 2 \ r \ n" на платформу Vinnter, щоб переключити світлодіод 2, викликаючи функцію nanoTime, щоб отримати поточний системний час. $End_k$ – це час відразу після виконання функції зчитування. Таким чином, інтервал між часом початку та часом закінчення – це затримка передачі між додатком і платформою Vinnter. Виміримо час n раз в циклі, щоб отримати більш точну середню затримку.

$$latency=\sum^{n}(end_k-start_k)/10 \quad (2.9)$$

Розрахуємо стандартне відхилення latency, використовуючи формулу (2.10), щоб побачити ступінь дисперсності latency. При обчисленні квадрата різниці між кожним інтервалом та середнім значенням потрібно розділити різницю на 10, щоб уникнути переповнення результату.

$$std = J\sum_{k=1}^{n}\sqrt{\text{interval}_k - \text{average})^2 / 10}. \quad (2.10)$$

Вимірювання проводиться в двох місцях з двома різними швидкостями мережі. Перше місце – це робочий офіс, в якій швидкість мережі (1 Гб/с), друге місце це дім де швидкість мережі 100 Мб/с, для гарантування достовірності результату було виконю близько 100 симуляцій.

Пропускна здатність, яку вимірюємо в цьому експерименті, – це різновид пропускної здатності, який показує, скільки раз проект може відправити повідомлення на платформу Vinnter та отримати відповідь, на повідомлення впродовж 10 секунд.

Значення пропускної здатності розраховується за формулою (2.11). В якій start1 – це час в наносекундах, коли записується перша команду на платформу Vinnter. А end1 – це час отримання останніх даних. Таким чином, інтервал між start1 та end1 це загальний час, протягом якого відправлено n команд й отримані дані останньої відповіді.

$$throughput=10^9/(end_1-start_1)*100 \quad (2.11)$$

## 2.7 Висновки до розділу 2

У розділі 2 вирішена задача планування експериментального дослідження системи передачі даних на платформах інтернету речей, яка полягає в обґрунтованому виборі інструментальних засобів, необхідних для дослідження та розробки плану експерименту.

Розглянута теорія планування експерименту та базові методи математичної статистики для аналізу даних.

Поставлена мета експерименту та описані сценарії його проведення. Кожний сценарій передбачає зміні фактори впливу на експеримент, а саме варіант зв’язку(вид зв’язку), системи зв’язку та місце проведення експерименту. Також для кожного сценарію вхідним факторам є кількість даних, що передається.

Відгуками експерименту є пропускна здатність та час очікування відгуку системи.

Варіанти зв’язку які плануються – це послідовний зв'язок, UDP/TCP та Bluetooth.

Системи зв'язку IoT це три поширених платформи: SensibleThings, Microsoft Azure та Kaa.

Після аналізу факторів було обрано два види зв’язку, а саме послідовний зв'язок, UDP/TCP та одна платформа SensibleThings, так як інші фактори мають приблизно так самі значення впливу на експеримент та вважаються не суттєвими.

Для оцінки системи в цілому вирішено розробити прототип системі інтернету речей, який включає в себе джерло (клієнтський пристрій, може бути більше одного), IoT шлюз, платформу керування та оброблення даних SensibleThings(приймай).

Розглянуто методи попередньої обробки даних та наведено методи вимірювання відгуків експерименту.

# 3 АНАЛІЗ РЕЗУЛЬТАТІВ ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ НА ПЛАТФОРМІ ІНТЕРНЕТУ РЕЧЕЙ

У цьому розділі представлено результати експериментального дослідження системи передачі даних на платформі інтернету речей, які включають тестування, вимірювання та оцінку затримки та пропускної здатності каналу зв’язку.

## 3.1 Результати передачі даних через шлюз Інтернету речей

### 3.1.1 Тестування функцій зчитування та запису

Протестовано правильність роботи функцій зчитування та запису після установки з'єднання. Результат показаний на рисунку 31. Це відповіді платформи Vinnter при написанні «help \ r \ n» і «toggle_led 2 \ r \ n» на платформі Vinnter. Результат зчитування правильний. Також горить світлодіод 2.

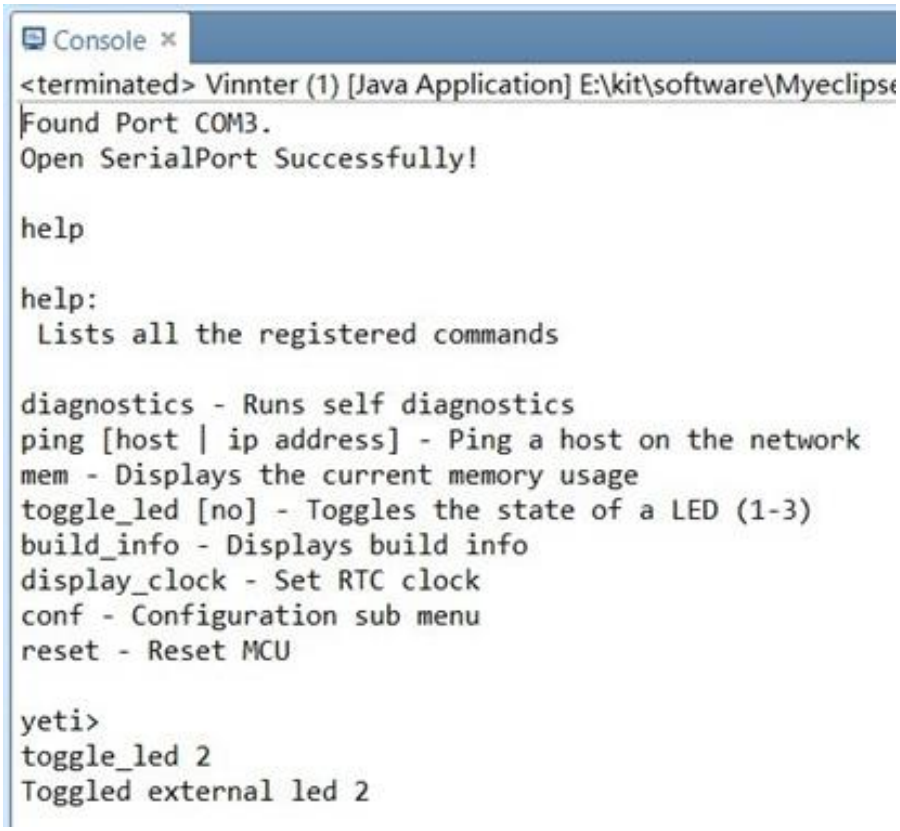



Рисунок 3.1 – Результати запису та зчитування

### 3.1.2 Аналіз отриманих значень

Середня затримка, виміряна на роботі, варіюється, між двох значень, одне з яких складає близько 4,7*10^7 наносекунд, а друге – 4, *10^7 наносекунд, що відображено на рисунках 3.2–3.3. Також є значення, яке можна віднести до виключення, тому що воно набагато більше, ніж інші та його відхилення вище стандартного відхилення за обраною вибіркою. Таким чином, якщо не враховувати значення виключення, затримка повинна бути близько 4,7*10^7 наносекунд.

```
yeti>
    start time,      end time,      interval:
1: 103325167088965 103325215063715 4.797475E7
2: 103325216893776 103325262152663 4.5258887E7
3: 103325262284306 103325312177028 4.9892722E7
4: 103325312342586 103325360199972 4.7857386E7
5: 103325360455672 103325407238048 4.6782376E7
6: 103325407486161 103325456317260 4.8831099E7
7: 103325456531459 103325503342395 4.6810936E7
8: 103325503557932 103325550378685 4.6820753E7
9: 103325550597347 103325600442766 4.9845419E7
10: 103325600660089 103325647466562 4.6806473E7
The average latency is: 4.76880801E7 nanoseconds.
The standard deviation of latency is: 1412517 nanoseconds.
```

Рисунок 3.2 – Затримка при передачі на роботі перша вибірка

```
yeti>
    start time,      end time,      interval:
1: 48138641002218 48138688996156 4.7993938E7
2: 48138753092942 48138799096169 4.6003227E7
3: 48138799374628 48138848077207 4.8702579E7
4: 48138848325767 48138897136785 4.8811018E7
5: 48138897377759 48138959205821 6.1828062E7
6: 48138959462414 48139008198016 4.8735602E7
7: 48139008437651 48139055215118 4.6777467E7
8: 48139055443150 48139103332666 4.7889516E7
9: 48139103577210 48139152269972 4.8692762E7
10: 48139152570297 48139199403991 4.6833694E7
The average latency is: 4.92267865E7 nanoseconds.
The standard deviation of latency is: 4303392 nanoseconds.
```

Рисунок 3.3 – затримка при передачі на роботі друга вибірка

Середня затримка, виміряна вдома, варіюється в широкому діапазоні. Дві вибірки показані на рисунках 3.4 та 3.5, які сильно відрізняються одна від одної через дані про винятки(викиди).

```
yeti>
    start time,      end time,      interval:
1: 7912974127498 7913022115995 4.7988497E7
2: 7913024358386 7913071175039 4.6816653E7
3: 7913071426276 7913118206783 4.6780507E7
4: 7913118450434 7913167189519 4.8739085E7
5: 7913167487612 7913214233758 4.6746146E7
6: 7913214493474 7913263345905 4.8852431E7
7: 7913263922456 7913310345073 4.6422617E7
8: 7913310595864 7913359435801 4.8839937E7
9: 7913359681236 7913406382758 4.6701522E7
10: 7913406623731 7913455504723 4.8880992E7
The average latency is: 4.76768387E7 nanoseconds.
The standard deviation of latency is: 1017061 nanoseconds.
```

Рисунок 3.4 – Затримка вдома, вибірка один

```
yeti>
    start time,      end time,      interval:
1: 7816383286358 7816431279764 4.7993406E7
2: 7816433447185 7816494363009 6.0915824E7
3: 7816494611122 7816543453736 4.8842614E7
4: 7816543722823 7816590496190 4.6773367E7
5: 7816590745196 7816639568622 4.8823426E7
6: 7816639817181 7816686599920 4.6782739E7
7: 7816686852942 7816735638436 4.8785494E7
8: 7816735935636 7816782631357 4.6695721E7
9: 7816782887502 7816831650238 4.8762736E7
10: 7816831914863 7816878766769 4.6851906E7
The average latency is: 4.91227233E7 nanoseconds.
The standard deviation of latency is: 4034921 nanoseconds.
```

Рисунок 3.5 – Затримка вдома, вибірка два

Проаналізувавши дані, можна зробити висновок, що затримка не пов'язана зі швидкістю мережі. Тому що при різній швидкості мережі затримки схожі одна на іншу. Але є дані про винятки, які показують, що затримка може бути пов'язана з використанням процесора або деякими

іншими аспектами. Тому розглянем дані затримки та стандартного відхилення затримки (таблиця 3.1). Проаналізувавши дані отримуємо, що час очікування та стандартне відхилення вдома мало відріжняється від того що на роботі.

Таблиця 3.1 – Значення затримки та стандартного відхилення

| Місце проведення | Затримка / нс | Стандартне відхилення / нс |
|---|---|---|
| На роботі (1) | $4{,}7 * 10^7$ | $1{,}4 * 10^6$ |
| На роботі (2) | $4{,}9 * 10^7$ | $4{,}3 * 10^6$ |
| Вдома (1) | $4{,}7 * 10^7$ | $1{,}0 * 10^6$ |
| Вдома (2) | $4{,}9 * 10^7$ | $4{,}0 * 10^6$ |

Виходячи з аналізу даних, ми можемо зробити висновок, що послідовний зв'язок не пов'язаний зі швидкістю мережі.

Один з результатів вимірювання пропускної здатності показаний на рисунку 3.6. Пропускна здатність становить 602 рази за десять секунд. Всі результати, які я отримано, приблизно 60 раз/сек. Один з таких результатів, для циклу рівному 50, наведено на рисунку 3.7., де пропускна здатність становить 2378 разів за 50 секунд. Оскільки запусків було 40 разів, можливо оцінити середнє арифметичне, яке склало 48 разів на секунду.

```
Start time is: 44781651557159
End time is: 44781817488632
Interval is: 1.65931473E8
The throughput is 602 times in 10 seconds.

Close successfully!
```

Рисунок 3.6 – Результат вимірювання пропускної здатності на інтервалі в 10 секунд

```
Start time is: 44907079317437
End time is: 44907289539458
Interval is: 2.10222021E8
The throughput is 2378 times in 10 seconds.

Close successfully!
```

Рисунок 3.7 – Результат вимірювання пропускної здатності для 5 запитів по 10 секунд

Оцінка пропускної здатності виявила, що при циклічній передачі даних загальна пропускна здатність каналу зросла в 5 разів, в порівняні з передачею пакету.

Одним із цікавих результатів було б виявити, якою є максимально досяжна пропускна здатність, моєї реалізації за замовчуванням. Тест проводився для двох сценаріїв, використовуючи протоколи транспортного рівня TCP та UDP. В кожному випадку використовувався лише один клієнт Iperf3, який поступово збільшував пропускну здатність з'єднання. Результат тесту зображений на рисунку 3.8.

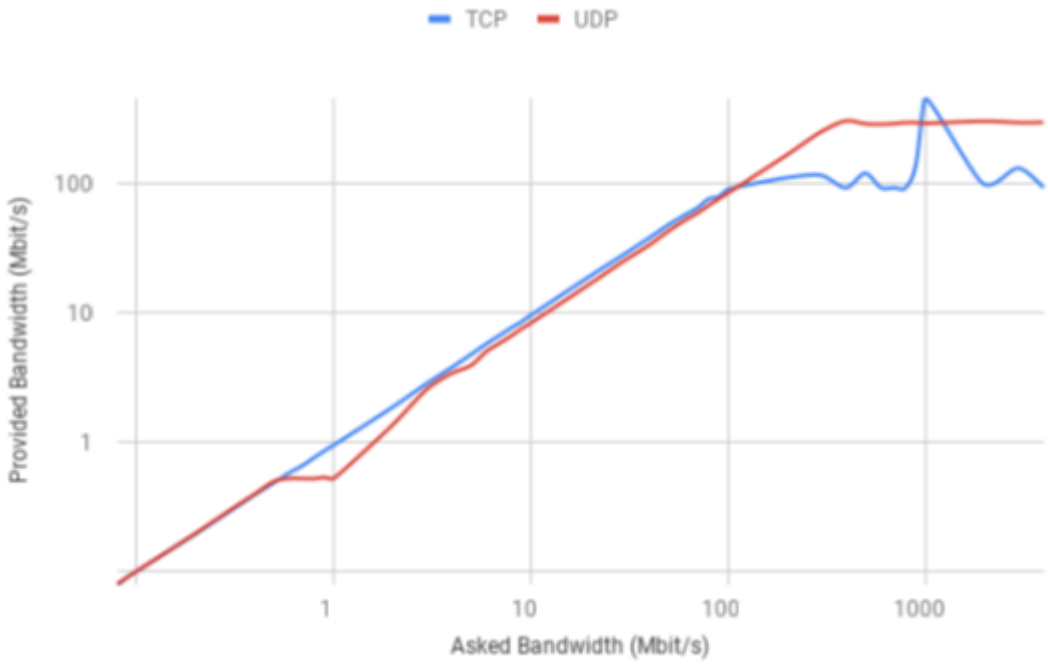



Рисунок 3.8 – Максимально досяжна швидкість передачі даних при використанні протоколів TCP та UDP.

Незважаючи на один помітний пік 160 Мбіт/с, TCP приблизно стабілізувався швидкістю 100 Мбіт / с, тоді як UDP зміг досягунти приблизно на рівні 300 Мбіт / с. На основі отриманих даних найкращий протокол TCP, у

якого менший бітрейт, аніж UDP. Запропоноване рішення стосується двох сценаріїв, де допускається існування кількох кінцевих пристроїв, які можуть обслуговувати широкий спектр застосувань.

### 3.2 Результати передачі даних для всієї системи

Після додавання частини SensibleThings проект завершено. Можлна запустити весь проект в цілому та перевірити загальні метрики передачі даних. Вихідний проект та проект приймача виконуються на двох комп'ютерах. Потрібно спочатку запустити вихідний проект, щоб зареєстрований UCI для приймача, і налаштувати послідовний зв'язок для підготовки до вхідних командам на платформу Vinnter. Результат відображені на рисунку 3.9, який демонструє, що джерело працює. Потім запустимо додаток стоку, весь зв'язок буде налаштований.


Рисунок 3.9 – Успішний запуск всього проекту

Зміна стану до та після надання команди відповідно показані на рисунку 3.10. Отже, зв'язок між приймачем та платформою Vinnter успішно встановлений, а інтеграція обладнання та програмного забезпечення IoT завершена.

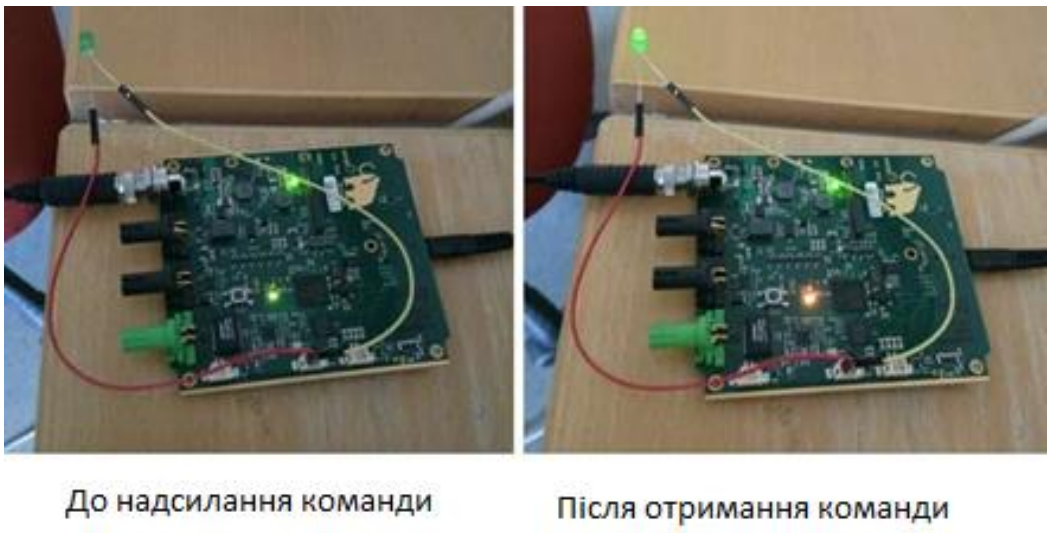



Рисунок 3.10 – Змін стану світлодіоду

### 3.2.1 Оцінка затримки передачі даних

Результат тесту для приймача показаний на рисунку 3.11, де відображено, що дані відправлені на платформу Vinnter успішно, та отримані дані також вірні. Після запуску приймача джерело також відповість на задану дію приймача.

```
Console ×
Sink (1) [Java Application] E:\kit\software\Myeclipse\binary\com.sun.java.jdk7.win
SensibleThings Sink is running
Press any key to shutdown

[ResolveResponse]: project  iun  e/actuator, 193.10.119.42:37690

[GetResponse]: help

help:
 Lists all the registered commands

diagnostics - Runs self diagnostics
ping [host | ip address] - Ping a host on the network
mem - Displays the current memory usage
toggle_led [no] - Toggles the state of a LED (1-3)
build_info - Displays build info
display_clock - Set RTC clock
conf - Configuration sub menu
reset - Reset MCU

yeti>

[GetResponse]: toggle_led 2
Toggled external led 2
```

Рисунок 3.11 – Результат тесту для приймача

Результат тесту в джерелі показаний на рисунку 3.12. Результат в джерелі також правильний. Відправивлено повідомлення «toggle_led 2» джерелу т отримано відповідь. Таким чином, стан світлодіода 2, який раніше був підключений до платформи Vinnter, було переключено.


Рисунок 3.12 – Результат тесту джерела

Однак, якщо спочатку запусти приймач без запущеного джерела, він просто покаже, що приймач запущений, та відповідь на дозвіл UCI, який є адресою вузла джерела. Оскільки немає запущеного джерела, немає відповіді джерела на задану дію. Це показано на рисунку 3.13.


Рисунок 3.13 – Запуск проекту без джерела

Після того, як всі впровадження та тестування зроблені. Можна оцінити всю систему в цілому. Затримка та пропускна здатність мають варіювання. Два результату вимірювання затримки в проекті приймача на роботі показані на рисунку 3.14 та рисунку 3.15. Затримка становить близько 130 і 180 мілісекунд. Стандартне відхилення на рисунку 3.15 вище, ніж на рисунку 3.14.

```
yeti>
  start time,   end time,     interval
1 1495197718747 1495197718882 135.0
2 1495197719247 1495197719378 131.0
3 1495197719748 1495197719889 141.0
4 1495197720248 1495197720381 133.0
5 1495197720748 1495197720881 133.0
6 1495197721249 1495197721381 132.0
7 1495197721749 1495197721867 118.0
8 1495197722250 1495197722386 136.0
9 1495197722750 1495197722878 128.0
10 1495197723250 1495197723372 122.0
The average latency is: 130.9 milliseconds.
The standard deviation of latency is: 6.39 milliseconds.
```

Рисунок 3.14 – Результати вимірювання затримки на роботі, вибірка один

```
yeti>
  start time,   end time,     interval
1 1495198075457 1495198075649 192.0
2 1495198075958 1495198076129 171.0
3 1495198076458 1495198076642 184.0
4 1495198076959 1495198077141 182.0
5 1495198077459 1495198077662 203.0
6 1495198077959 1495198078145 186.0
7 1495198078460 1495198078659 199.0
8 1495198078960 1495198079153 193.0
9 1495198079461 1495198079651 190.0
10 1495198079961 1495198080125 164.0
The average latency is: 186.4 milliseconds.
The standard deviation of latency is: 11.34 milliseconds.
```

Рисунок 3.15 – Результати вимірювання затримки на роботі, вибірка два

Середня затримка, виміряна вдома, варіюється в більш широкому діапазоні. Дві вибірки показані на рисунку 3.16 та рисунку 3.17. Значення вибірок сильно відрізняються одні від одного. Та стандартні відхилення сильно відрізняються один від одного.

```
yeti>
  start time,   end time,     interval
1 1495144961164 1495144961405 241.0
2 1495144961665 1495144961913 248.0
3 1495144962165 1495144962408 243.0
4 1495144962666 1495144962907 241.0
5 1495144963166 1495144963399 233.0
6 1495144963667 1495144963912 245.0
7 1495144964167 1495144964408 241.0
8 1495144964667 1495144964904 237.0
9 1495144965168 1495144965414 246.0
10 1495144965668 1495144965927 259.0
The average latency is: 243.4 milliseconds.
The standard deviation of latency is: 6.64 milliseconds.
```

Рисунок 3.16 – Результати вимірювання затримки вдома, вибірка один

```
yeti>
  start time,   end time,     interval
1 1495145117711 1495145118218 507.0
2 1495145118211 1495145118414 203.0
3 1495145118712 1495145119121 409.0
4 1495145119212 1495145119437 225.0
5 1495145119713 1495145120067 354.0
6 1495145120213 1495145120435 222.0
7 1495145120713 1495145120994 281.0
8 1495145121214 1495145121603 389.0
9 1495145121714 1495145121933 219.0
10 1495145122215 1495145122529 314.0
The average latency is: 312.3 milliseconds.
The standard deviation of latency is: 96.07 milliseconds.
```

Рисунок 3.17 – Результати вимірювання затримки вдома, вибірка два

На рисунку 3.18 показаний результат в джерелі, який повідомляє нам про задану подію та правильне повернення результату.

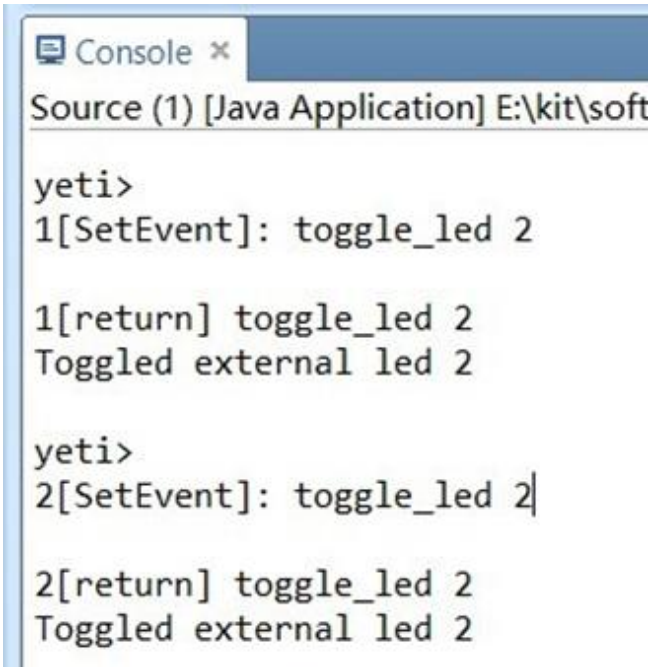



Рисунок 3.18 – Результати функціонування джерела

Значення затримки та стандартного відхилення затримки наведено в таблиці 3.2. Час очікування та стандартне відхилення вдома відрізняються від тих, що на роботі. Отже, робимо висновок, що затримка пов'язана зі швидкістю мережі, тому що при високій швидкості мережі затримка набагато менше, ніж при низькій швидкості мережі.

Таблиця 3.2 – Час очікування та стандартне відхилення для всього проекту

| Місце експерименту | Затримка / мс | Стандартне відхилення / мс |
|---|---|---|
| На роботі(1) | 130,9 | 6,39 |
| На роботі(2) | 186,4 | 11,34 |
| Вдома (1) | 243,4 | 6,64 |
| Вдома (2) | 312,3 | 96,07 |

У порівнянні з послідовним зв'язком, середня затримка всієї системи становить близько 200 мілісекунд, що набагато більше, ніж у послідовного зв'язку через шлюз, для якого середня затримка становить близько 47 мілісекунд.

Отже, виходячи з цих даних, можна зробити висновок, що на затримку всієї системи впливає швидкість використаної мережі.

### 3.2.2 Оцінка пропускної здатності усього проекту

Оцінка пропускної здатності в приймачеві при використанні команди 10 разів та 20 разів, яка відображена відповідно на рисунках 3.19 і 3.20. Пропускна здатність вварюється в широкому діапазоні, тому важко зробити висновки, яке значення є сталим. Для оцінки пропускної здатності використаємо замість середнього значення моду вибірки. Друга проблема що зі збільшенням часу роботи пропускна здатність стає менше. Причина може бути в тому, що платформа не встигає виконувати команди та вони накопичуються в черзі, а стандартні механізми керування чергою, працюють повільно.

```
13 Start time is: 1494773115571
End time is: 1494773116196
Interval is: 625
The throughput is 160 times in 10 seconds.

14 Start time is: 1494773115571
End time is: 1494773116295
Interval is: 724
The throughput is 138 times in 10 seconds.
```

Рисунок 3.19 – Пропускна здатність при 10 командах

```
Start time is: 1495403729296
End time is: 1495403730426
Interval is: 1130.0
The throughput is 88 times in 10 seconds.

Start time is: 1495403729296
End time is: 1495403730487
Interval is: 1191.0
The throughput is 83 times in 10 seconds.
```

Рисунок 3.20 – Пропускна здатність для 20 команд

На рисунку 3.21 показаний результат в джерелі, який повідомляє нам, що встановлене значення є правильним.

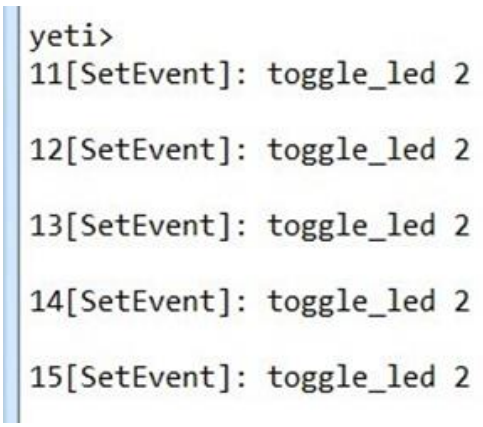



Рисунок 3.21 – Результат команд

### 3.4 Висновки до розділу 3

У розділі розглянуті результати експериментального дослідження системи передачі даних на платформі інтернету речей.

Протестована програмно-апаратна платформа інтернету речей, для забезпечення надійного функціонування при проведені експерименту. А саме протестовані функція з’єднання компонентів системи та функції передачі даних та керуючих команд.

В результаті тестування передачі даних між двома пристроями через шлюз інтернету речей. Отримані наступні результати. Затримка при передачі даних для тестового набору даних склала 4,7*10^7 наносекуннди. Провівши порівняння отриманих значень для різних місць тестування, а саме різної швидкості мережі в якій здійснювалось тестування, можна відмітити, що затримка не залежить від швидкості мережі, якщо її швидкість не менше 50 Мбіт/с. Пропускна швидкість в залежності від об’єму пакету даних зростає при збільшенні пакету даних, але є критичні точки, при яких виявлено різкий спад пропускної можливості.

Дані отримані в результат використання системи в цілому мають відмінності від результату тестування окремої частини. А саме у порівнянні з послідовним зв'язком, середня затримка всієї системи становить близько 200 мілісекунд, що набагато більше, ніж у послідовного зв'язку через шлюз, для якого середня затримка становить близько 47 мілісекунд, проаналізувавши дані виявлено, що робота системи в загалом має залежність від максимальної швидкості мережі, на відміну від підсистеми послідовного зв’язку. Пропускна здатність для цілої системи вища в 5 разів, але має характер накопичення даних в черзі, при цьому продуктивність самої системи падає.

# ВИСНОВКИ

Виконано аналіз найбільш поширених підходів для комунікації на платформах інтернету речей. А саме широкий спектр можливостей доступу надають сучасні стандарти комунікаційних технологій короткого діапазону, які варіюються та включають Rubee, Z-Wave, Zigbee.

Одним з втілень цих технологій є платформа інтелектуальних продуктів Vinnter, яка була розроблена в відповідності вимог промислового рівня, надаючи можливості підключення до існуючих та нових продуктів, маючи при цьому широкі можливості налаштувань. Що дозволило прискорити підтвердження концепції та розробки продукту, а також знизити ризики проекту, вартість та час виведення на ринок. Ця платформа має основну плату, яка забезпечує можливість підключення додаткових плат керування або плат інтерфейсу користувача, які розширять можливості платформи.

Виділено одне з сучасних рішень в області зв’язку інтернету речей, а саме архітектура Smart Gateway з двома видами зв'язку, перший вид – це односторонній, в якому шлюз, який збирає дані та надсилає їх у туман, а потім у хмару, безпосередньо підключену до датчиків та інших речей. А інший різновид – це мульти-стрибок, в якому підключаються кілька датчиків мережі та IoT.

Проаналізувавши доступні механізми та засоби оцінки передачі даних на платформах IoT, було виявлено, що інструментарії стеження та оцінки для хмарних платформ існують та мають деякий функціонал, але не можуть повністю покрити питання якості зв’язку, наприклад пропускної здатності для всього рішення IoT та мають прихований механізм роботи, що не дає змогу дослідникам зрозуміти закономірності роботи рішення в цілому, що також підтверджує актуальність да доцільність проведення експериментального дослідження передачі даних в системі інтернету речей.

Проведено планування експериментального дослідження систем передачі даних на платформах інтернету речей. В результаті, якого поставлена мета експерименту та описані сценарії його проведення. Кожний сценарій передбачає зміні фактори впливу на експеримент, а саме варіант зв’язку(вид зв’язку), системи зв’язку та місце проведення експерименту. Також для кожного сценарію вхідним факторам є кількість даних, що передається. Відгуками експерименту є пропускна здатність та час очікування відгуку системи. Варіанти зв’язку які плануються – це послідовний зв'язок, UDP/TCP та Bluetooth.

Після аналізу факторів було обрано два види зв’язку, а саме послідовний зв'язок, UDP/TCP та одна платформа SensibleThings, так як інші фактори мають приблизно так самі значення впливу на експеримент та вважаються не суттєвими.

Для оцінки системи в цілому вирішено розробити прототип системі інтернету речей, який включає в себе джерло (клієнтський пристрій, може бути більше одного), IoT шлюз, платформу керування та оброблення даних SensibleThings(приймай).

Розглянуто методи попередньої обробки даних та наведено методи вимірювання відгуків експерименту.

Розроблено прототип рішення інтернету речей для проведення експериментального дослідження передачі даних на платформах інтернету речей та проведено його тестування. Проведено експеримент з використанням розробленого прототипу.

Проаналізовано експериментальні дані отриманні в результаті експерименту з передачі даних на платформах інтернету речей, а саме затримка при передачі даних для тестового набору даних склала 4,7*10^7 наносекуннди. Провівши порівняння отриманих значень для різних місць тестування, а саме різної швидкості мережі в якій здійснювалось тестування, можна відмітити, що затримка не залежить від швидкості мережі, якщо її швидкість не менше 50 Мбіт/с. Пропускна швидкість в залежності від об’єму пакету даних зростає при збільшенні пакету даних, але є критичні точки, при яких виявлено різкий спад пропускної можливості.

Дані отримані в результат використання системи в цілому мають відмінності від результату тестування окремої частини. А саме у порівнянні з послідовним зв'язком, середня затримка всієї системи становить близько 200 мілісекунд, що набагато більше, ніж у послідовного зв'язку через шлюз, для якого середня затримка становить близько 47 мілісекунд, проаналізувавши дані виявлено, що робота системи в загалом має залежність від максимальної швидкості мережі, на відміну від підсистеми послідовного зв’язку. Пропускна здатність для цілої системи вища в 5 разів, але має характер накопичення даних в черзі, при цьому продуктивність самої системи падає.

## ПЕРЕЛІК ПОСИЛАНЬ

1 A1tzori L. Understanding the Internet of Things: definition, potentials, and societal role of a fast evolving paradigm / L. Atzori, A. Lera, G. Morabito. // Ad Hoc Networks. – 2017. – №56. – С. 121–140.

2 Rouse M. Internet of things (IoT) / Margaret Rouse. – New York, 2019. – 624 с

3 OT Gateway: Bridging Wireless Sensor Networks into Internet of Things / [Z. Qian, W. Ruicong, C. Qi та ін.]. // IEEE/IFIP International Conference on Embedded and Ubiquitous Computing. – 2010. – С. 2–38.

4 ITU's Telecommunication Standardization Sector [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.itu.int/dms_pub/itu-t/opb/gen/T-GEN-OVW-2014-PDF-E.pdf.

5 Evans D. The Internet of Things How the Next Evolution of the Internet Is Changing Everything [Електронний ресурс] / Dave Evans // Cisco. – 2011. – Режим доступу до ресурсу: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

6 Communications System [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.techopedia.com/definition/18430/communications-system.

7 Weber R. Internet of things – Need for a new legal environment? / Rolf Weber. // Computer Law & Security Report. – 2009. – №25. – С. 522–527.

8 Chen H. A BRIEF INTRODUCTION TO IOT GATEWAY / H. Chen, X. Jia, H. Li. // IET International Conference on Communication Technology and Application. – 2011. – С. 54–98.

9 Goldsmith A. Wireless Communications / Goldsmith. – Cambridge: Cambridge University Press, 2012. – 644 с.

10 Xie Y. Распространенные методы связи IoT [Електронний ресурс] / Yong Xie. – 2016. – Режим доступу до ресурсу: https://www.cnblogs.com/legahero/p/IOT.html.

11 An overview on wireless sensor networks technology and evolution. Sensors. / C.Buratti, A. Conti, D. Dardari, R. Verdone. – Basel, 2009. – 869 с.

12 Charalampos D. Bringing IoT and Cloud Computing towards Pervasive Healthcare / D. Charalampos, M. Ilias. // Sixth International Conference on Innovative Mobile and Internet Service in Ubiquitous Computing. – 2012. – С. 55–98.

13 Eui-Nam H. Fog Computing and Smart Gateway Based Communication for Cloud of Things / H. Eui-Nam, M. Aazam. // International Conference on Future Internet of Things ad Cloud. – 2014. – С. 25–62.

14 Dieter S. Towards the Implementation of IoT for Environmental Condition Monitoring in Homes / S. Dieter, T. Kelly, N. Kumar. // IEEE Sensors Journal. –

2013. – №13. – С. 14–56.

15 Serial communication [Електронний ресурс] // Wikipedia. – 2019. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Serial_communication.

16 JIMBLOM. Serial communication [Електронний ресурс] / JIMBLOM – Режим доступу до ресурсу: https://learn.sparkfun.com/tutorials/serial-communication/all.

17 Serial Communication – Introduction [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: http://maxembedded.com/2013/09/serial-communication-introduction/.

18 Kurose J. Computer Networking: A Top-Down Approach / J. Kurose, K. Ross., 2016. – 864 с. – (7).

19 Communication Networks/TCP and UDP Protocols [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikibooks.org/wiki/Communication_Networks/TCP_and_UDP_Protocols.

20 Bluetooth [Електронний ресурс] – Режим доступу до ресурсу: https://www.bluetooth.com/about-us/.

21 Bluetooth Basics [Електронний ресурс] – Режим доступу до ресурсу: https://learn.sparkfun.com/tutorials/bluetooth-basics.

22 Fully Distributed Ubiquitous Information Sharing on a Global Scale for the Internet-of-Things / V.Kardeby, S. Forsström, P. Österberg, U. Jennehag. // International Journal on Advances in Telecommunications. – 2014. – №7. – С. 69–81.

23 MediaSense – an Internet of Things Platform for Scalable and Decentralized Context Sharing and Control / [S. Forsström, V. Kardeby, J. Walters та ін.]. // The Seventh International Conference on Digital Telecommunications. – 2012. – С. 27–32.

24 Challenges when Realizing a Fully Distributed Internet-of-Things – How we Created the SensibleThings Platform / S.Forsström, V. Kardeby, P. Österberg, U. Jennehag. // The Ninth International Conference on Digital Telecommunications. – 2014. – С. 13–18.

25 What is Azure? [Електронний ресурс] // Microsoft Corporation. – 2020. – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/overview/what-is-azure/.

26HAPPELL D. Introducing the azure services platform / David Chappell., 2009. – 32 с. – (Microsoft Corporation).

27 Kaa IoT Platform Features for Enterprise IoT Projects [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaaproject.org/overview.

28 Design reference [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://docs.kaaproject.org/display/KAA/Design+reference.

# ДОДАТОК А

Ілюстративні матеріали

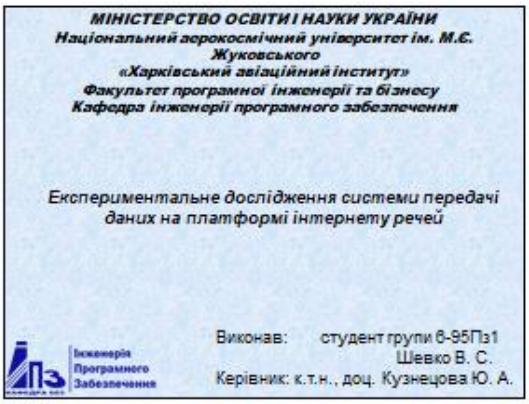




Рисунок А1 – Слайд 1–2

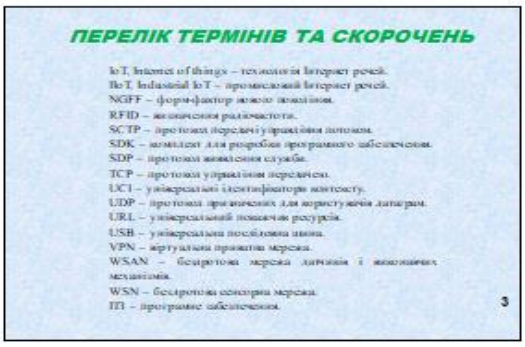
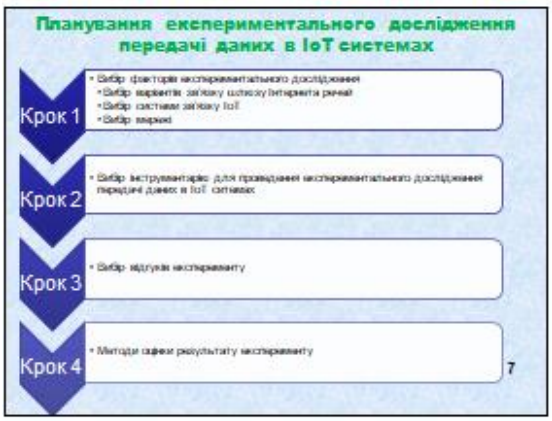




Рисунок А2 – Слайд 3-4



Рисунок А3 – Слайд 5-6



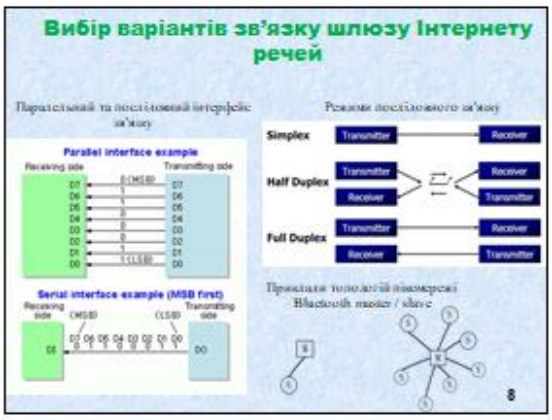




Рисунок А4 – Слайд 7-8

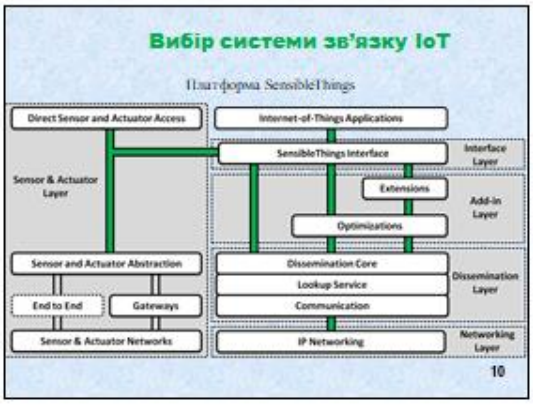
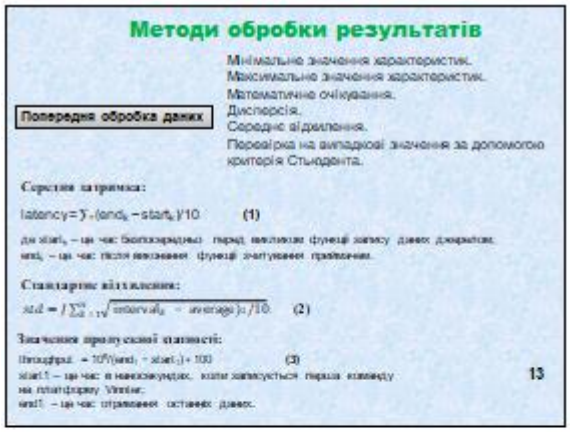




Рисунок А5 – Слайд 9-10



Рисунок А6 – Слайд 11-12



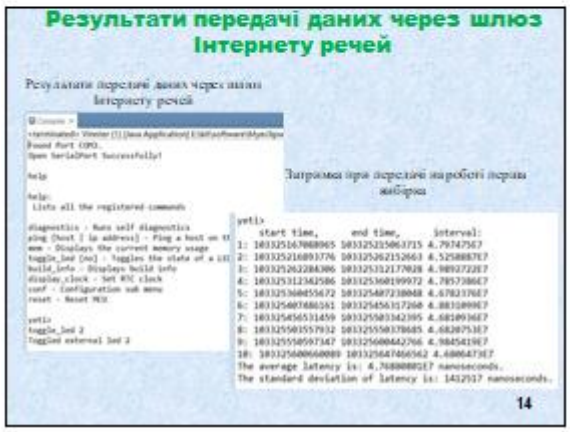




Рисунок А7 – Слайд 13-14

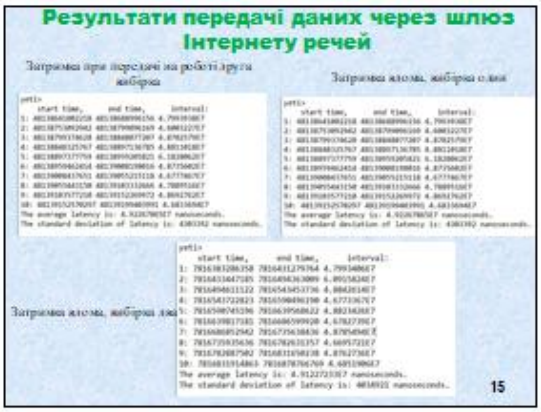
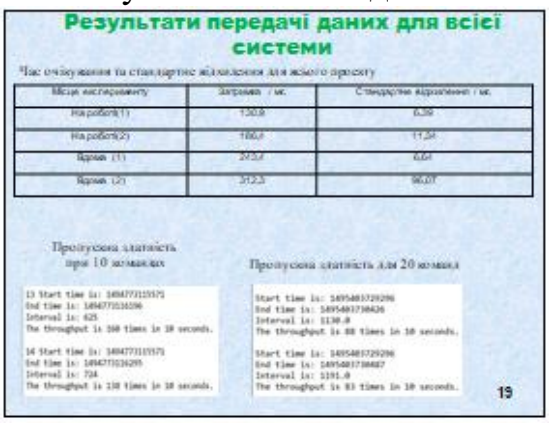




Рисунок А8 – Слайд 15-16



Рисунок А9 – Слайд 17-18



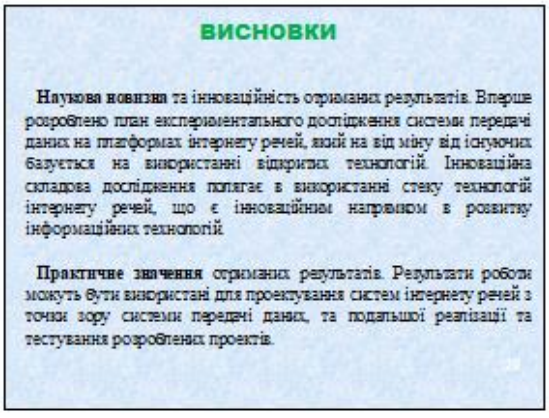




Рисунок А10 – Слайд 19-20

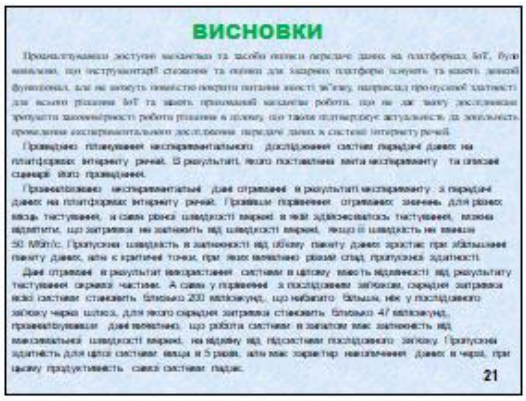




Рисунок А11 – Слайд 21-22