Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Електронної та біомедичної інженерії
(повна назва)

Кафедра Мікроелектроніки, електронних приладів та пристроїв
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Моделювання процесу передачі та прийому даних
в пристроях Інтернету речей (IoT)
(тема)

Виконав:
студент 2 курсу, групи ЕППм–18–1
Нагай Володимир Вячеславович
(прізвище, ініціали)
Спеціальність 171 Електроніка
(код і повна назва спеціальності)
Тип програми освітньо–професійна
Освітня програма «Електронні прилади та пристрої»
Керівник проф. Стрілкова Т.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Бондаренко І.М.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Електронної та біомедичної інженерії

Кафедра Мікроелектроніки, електронних приладів та пристроїв
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 171 «Електроніка»
(код і повна назва)

Тип програми освітньо–професійна
(освітньо–професійна або освітньо–наукова)

Освітня програма «Електронні прилади та пристрої»
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20__ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Нагай Володимиру Вячеславовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Моделювання процесу передачі та прийому даних в пристроях Інтернету речей (IoT)»

затверджена наказом по університету від 04 листопад 2019 р. № 1635 Ст
2. Термін подання студентом роботи до екзаменаційної комісії _____ грудня 2019 р.
3. Вихідні дані до роботи
Різні розміри масивів для протоколів HTTP та MQTT
4 ядерні процесори
4 Гб ОЗУ
Безпровідна передача даних LTE 3G
Одноплатний мікрокомпьютер Raspberry Pi 4 Model B
Тонкий клієнт HP t630
Десктоп сервер ProLiant ML30
4. Перелік питань, що потрібно опрацювати в роботі:
Характеристики протоколів HTTP і MQTT
Розрахунок ймовірності помилки (BER – Bit Error Rate) в залежності від реалізованого значення S/N.
Методологія розрахунків продуктивності мереж
Модель керування потоками протоколів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Вступ
Розрахунку найбільш популярних протоколів для датчиків та соціальних мереж
Імітаційне моделювання мережі в програмі
Порівняння різних пристроїв для створення сервера та його тестування
Принцип роботи кластера та його програмне забезпечення
Спосіб управління пульсуючими потоками протокольних блоків даних

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційну роботу	04.11.19	Виконано
2	Аналіз предметної області, робота з літературою	06.11.19 – 15.11.19	Виконано
3	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	16.11.19 – 21.11.19	Виконано
4	Дослідження можливостей протоколів даних для взаємодії IoT пристроїв	22.11.19 – 25.11.19	Виконано
5	Порівняння характеристик серверних пристроїв	26.11.19 – 27.11.19	Виконано
6	Навантажувальне тестування зібраного віртуального кластера	28.11.19 – 29.11.19	Виконано
7	Розробка моделі передачі даних в пристроях IoT	30.11.19 – 01.12.19	Виконано
8	Оформлення дипломної роботи	01.12.19 – 03.12.19	Виконано
9	Проходження нормоконтролю і отримання рецензії	04.12.19 – 10.12.19	Виконано
	Підготовка до захисту атестаційної роботи	11.12.19 – 23.12.19	Виконано

Дата видачі завдання __04__ ___листопада___2019 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: робота виконана на 81 сторінках, містить 36 ілюстрацій, 12 таблиці. При підготовці використовувалася література з 24 різних джерел.

МІНІ ПК, ВЕБ–СЕРВЕР, ПРОТОКОЛ, МІКРОКОМП’ЮТЕР, IOT, RASPBERRY PI, КЛАСТЕР, DOCKER

Об’єктом дослідження – є пропускна здатність пристрою при пульсуючих потоків даних у IoT пристроях.

Предметом дослідження – є моделі побудови архітектури мережі, методи реалізації використання різних протоколів взаємодій та міні комп’ютері як IoT пристроїв.

Метою магістерської дипломної роботи є проведення систематизації, аналізу, порівняння та навантажувального тестування існуючих на ринку популярних серверів та міні комп’ютерів, дослідження поширених відкритих протоколів які взаємодіють з IoT пристроями, та також розробка зразка IoT системи та надання практичних рекомендацій, що до розробки подібних систем.

Методи дослідження – в ході роботи моделювання процесу передачі та прийому даних в пристроях інтернету речей (IoT) було ілюстровано архітектуру інтернет речей, використано розрахунки за формулами, які демонструють інтенсивність пропускної здатності і часу відгуку системи та різниця параметрів між протоколами, параметри протоколів записані в таблиці, та відображені на графіках і гістограмах потоків; при розробці методів підвищення ефективності системи застосовано методи статистичного синтезу, аналіз літератури, аналіз подібних конструкцій, комп'ютерне моделювання та проектування.

ABSTRACT

Explanatory note: the work is made on 81 pages, contains 12 illustrations, 36 tables. In preparation, literature from 24 different sources was used.

MINI PC, WEB SERVER, PROTOCOL, MICRO COMPUTER, IOT, RASPBERRY PI, CLUSTER, DOCKER

The object of the study is the bandwidth of the device when pulsing data flows in IoT devices.

The subject of the study is models for building network architecture, methods for implementing the use of various interaction protocols and mini computers as IoT devices.

The purpose of the master's thesis is to systematize, analyze, compare and load testing existing popular servers and mini–computers on the market, to study common open protocols that interact with IoT devices, as well as to develop a sample IoT system and provide practical recommendations for the development of such systems.

Research Methods – in the course of modeling the process of transmitting and receiving data on Internet of Things (IoT) devices, the architecture of the Internet of Things was illustrated, formulas were used to demonstrate the bandwidth and response time of the system and the difference between protocols, protocol parameters recorded in the table. , and displayed on flow charts and bar graphs; the methods of statistical synthesis, analysis of literature, analysis of similar structures, computer simulation and design were applied in the development of methods for improving the efficiency of the system.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

БД	База даних;
ВЧ	Високочастотний;
ПП	Передавач перешкод;
РЕЗ	Радіоелектронні засоби;
САПР	Систе́ма автоматизо́ваного проектува́ння – автоматизована система, призначена для автоматизації технологічного процесу проектування виробу;
BER	Bit error rate;
CPU	Central processing unit – функціональна частина комп'ютера, що призначена для інтерпретації команд;
FIFO	First–in, first–out;
GPIO	General Purpose Input/Output;
HTTP	HyperText Transfer Protocol;
IoT	Internet of Things, Інтернет речей;
IT	Інформаційні технології;
LAN	LocalAreaNetwork – об'єднання певного числа комп'ютерів на відносно невеликій території;
MQTT	Message Queue Telemetry Transport;
S/N	Signal to noise;
SNR	Signal noise ratio;
TCP/IP	Transmission Control Protocol / Internet Protocol – набір протоколів мережі Інтернет;
UDP	User Datagram Protocol – один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання.

ВСТУП

На цей час IoT напрям розвитку інформаційних технологій є надзвичайно популярний. З кожним роком все більше компаній виходять на ринок з різноманітними рішеннями у цій сфері. Впровадження IoT технологій у навколишнє середовище може суттєво покращити стан використання невідновлювальних енергоносіїв, спростити життя та підвищити комфорт життя. Усе це створює чудові передумови до росту цієї галузі.

Інтернет був створений 29 жовтня 1969 року, виглядав він як канал зв'язку між двома комп'ютерами в різних університетах США, комп'ютери вже тоді існували але займали дуже багато місця, тоді вчені були зацікавлені з збільшенні продуктивності і зменшенні розмірів комп'ютерів. Прогрес не стояв на місці і в 2006 році були винайдені, одноплатні мікрокомп’ютери розміром з банківську карту і працездатністю звичайного персонального комп'ютера.

З'явилися такі комп'ютери в 2011 році як малопотужні дуже маленькі універсальні пристрої, спочатку їх хотіли використовувати тільки для навчання студентів основам програмування і будови комп'ютерів, але Raspberry (так називався цей комп'ютер) отримав більш широкої поширення у різних користувачів, після великої кількості заявок на придбання даної моделі її поставили на широке виробництво. Raspberry має необмежену сферу застосування так як до нього можна підключити різні модулі і будь–які деталі. Так само разом з пристроями типу Raspberry використовують таку технологію як "Інтернет речей" яка є останнім етапом існування інтернету, коротко це концепція обчислювальних мереж яка пов'язує всі мобільні і не тільки пристрої і забезпечує їх автономну роботу без участі людини [9].

Дуже великі гроші витрачаються на придбання серверних комплексів обладнання для забезпечення безперебійної роботи різних інтернет ресурсів компаній, таких як інтернет–сайти, веб–додатки, інтернет–сервіси. Найчастіше частина ресурсів орендованого сервера є надлишкова, або не використовується

взагалі. При цьому малі IT компанії, групам веб–розробників, різним інформаційним відділам невеликих компаній, нічого не залишається окрім покупки готових рішень від великих виробників, для того що б протестувати продукт або, коли необхідно вивести інтернет ресурс в глобальну мережу, який поки ще має малу відвідуваністю. Таким чином, необхідно пристрій поєднує в собі низьку ціну, що володіє повною функціональністю і просте у використанні.

Актуальність обраної теми дипломної роботи обумовлена тим, що комп'ютери використовуються повсюдно, технологія IoT дозволяє з легкість контролювати всі аспекти життя людини. Ця сфера галузі динамічно розвивається та має великі перспективи. Раціональній вибір протоколу дозволяє значний зекономити час передачі даних, а раціональний вибір пристрою, який буде виконувати ці операції зекономить не тільки гроші а і споживання електроенергії.

Завданням випускної магістерської роботи є:

– розгляд поняття як працює глобальна мережа і реалізує передачу даних між пристроями;

– розрахунки передачі і прийому даних широко використовуваних протоколів, який доцільніше використовувати і до якого об’єму даних;

– порівняння різних типів пристроїв як серверного обладнання;

– вирішення проблеми з пульсуючим трафіком мережі, та відображення схеми корисної моделі, яка вирішує цю проблему.

За результатами наукових досліджень була зроблена публікація «Спосіб управління пульсуючими потоками протокольних блоків даних» на науково – практичній конференції «Інноватика в сучасній освіті та науці: теорія та практика» [19].

1 АНАЛІЗ АРХІТЕКТУРИ ІНТЕРНЕТ РЕЧЕЙ

1.1 Що таке IoT та її архітектура

Завдяки технологіям IoT, різні пристрої, центри обробки даних і користувачі можуть об'єднуватися між собою і передавати від одного іншому різну інформацію. У глобальному сенсі мережу Інтернету речей з'єднує в одну систему різні лінії зв'язку, мережеві шлюзи, маршрутизатори і кінцеві пристрої.

Інтернет речей не завжди виключає дії людини, а повністю автоматизує технології, так як орієнтований на потрібність людини і надає людині можливості доступу до речей. А кожна річ в IoT має свій унікальний ідентифікатор, mac – адресу, котрі утворюють багато інтернет речей, здатних взаємодіяти одне з одним, створюючи свої мережі (коротко чи довго строкові).

Основа будь–якої радіозв'язку – це її протокол. Він регламентує її топологію, маршрутизацію, адресацію, порядок доступу вузлів до каналу передачі даних, систему захисту та інші показники[1].

Вважається, що першу в світі інтернет–річ створив один з батьків протоколу TCP/IP Джон Ромкі в 1990 році, коли він підключив до мережі свій тостер. Але тільки в 21 столітті в зв'язку з бурхливим розвитком інформаційно–комунікаційних технологій сформувалася концепція IoT і отримала своє практичне втілення. Все почалося з необхідності оптимізації системи логістики та управління системою постачання підприємств. Друга хвиля інновацій була обумовлена необхідністю скорочення витрат в системах спостереження, безпеки, транспорту та ін. Третя була викликана потребою в геолокаційних сервісах [10]. Четверта хвиля буде обумовлена необхідністю дистанційної присутності людини на місці події і вимагатиме його уваги, яка стане можливим завдяки мініатюрним вбудованим процесорам. З розвитком Інтернету речей все більше предметів будуть підключатися до глобальної мережі, тим самим створюючи нові можливості в сфері безпеки, аналітики та управління,

відкриваючи все нові і більш широкі перспективи і сприяючи підвищенню якості життя населення. Передбачається, що в майбутньому «речі» стануть активними учасниками бізнесу, інформаційних і соціальних процесів, де вони зможуть взаємодіяти і спілкуватися між собою, обмінюючись інформацією про навколишнє середовище, реагуючи і впливаючи на процеси, що відбуваються в навколишньому світі, без втручання людини.

Однак до сих пір багато хто плутає IoT архітектуру з архітектурою автоматизації, де головним завданням є отримання інформації з датчиків, і на їх основі здійснюється управління виконавчими механізмами.

IT архітектура включає в себе дві, на перший погляд несумісні речі: з одного боку – це велика кількість периферійних пристроїв з малими обчислювальними потужностями, низьким енергоспоживанням, високою швидкістю реакції на події, а з іншого боку – хмарні сервера з високою обчислювальною потужністю для обробки великого масиву даних, їх зберігання та класифікації, часто з елементами машинного інтелекту і аналітики. Ці два світи використовують абсолютно різні принципи побудови і внутрішньої архітектури. Вони виглядають несумісними і сьогодні на ринку праці мало фахівців однаково добре знають Embedded і Cloud рішення. Це свого роду "Full Stack". Але в цьому знанні криється сила, яка об'єднує дві, на перший погляд абсолютно не пов'язаних технології. Від їх інтеграції ми отримуємо повноцінний проект.

На (рисунку 1.1) представлена загальна архітектура IoT рішення. Вельми передбачувано, що все починається з датчиків. Більш того, чим краще підходить датчик для виконання свого завдання, тим ефективніше буде система далі. Важливо відзначити, що датчик реєструє зміну навколишнього середовища, а не її статичний стан. Датчики поділяються на активні – випромінюють самі сигнали і приймають відображення; і пасивні – працюють тільки на прийом. Природно, що останні значно виграють за параметрами енергоспоживання.

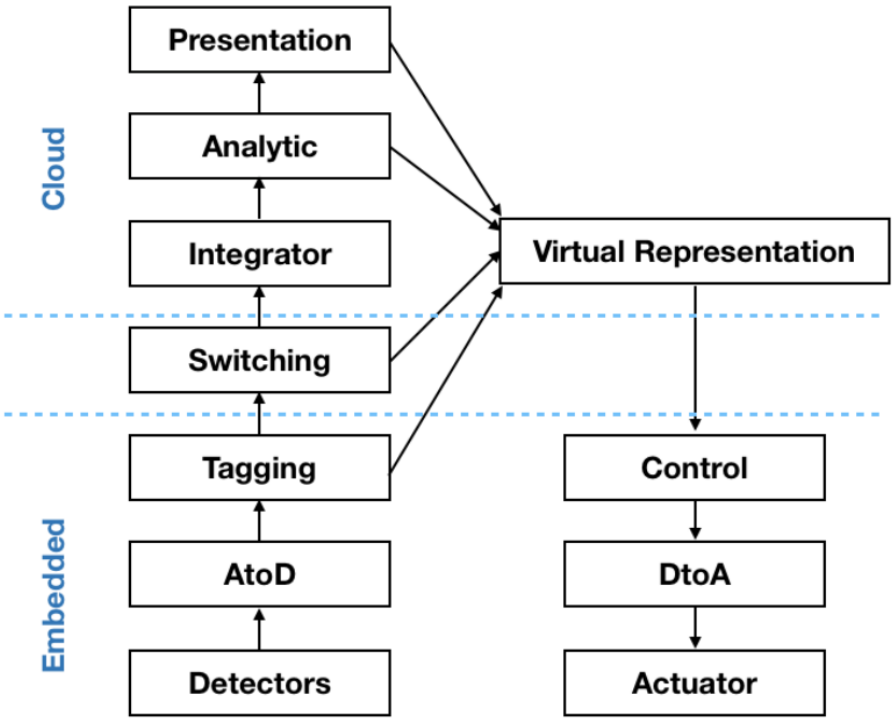


Рисунок 1.1 – Загальна архітектура IoT

У будь–якому випадку, датчик формує аналоговий сигнал, який необхідно перевести в цифру для подальшої обробки, чим і займається AtoD перетворювач. Після отримання цифрової інформації вона повинна бути оброблена локальним процесором периферійного пристрою. Головне його завдання проставити мітку отриманої інформації або просто класифікувати її. Мітки можуть бути найпростішими, як наприклад – є рух, так і більш складними – рух + швидкість. Іноді потрібні багатовимірні мітки – Рух, Машина. Чим складніший мітка, тим природно більше потужність периферійного процесора і відповідно енергоспоживання. З іншого боку, чим більше інформативна мітка, тим менше необхідна кількість інформації, що передається в хмару і відповідно потрібна менше смуга пропускання, а з цього збільшується швидкість реакції на подію.

Наступна ланка може знаходитися як в хмарі, так і на периферії, а іноді в обох частинах. Комутатор перенаправляє отриману інформацію в різні об'єкти, класифікуючи мітки. Цими об'єктами можуть бути сервера, черги, лямбда або

просто сховище. До сих пір робота велася з інформацією від конкретного периферійного пристрою і фактично нічим не відрізнялася від роботи автоматизованих систем управління. Однак на наступному рівні – інтеграції, починається якісну відмінність. Інформація від різних периферійних пристроїв підсумовується по однотипним міткам. При цьому самі типи периферійних пристроїв можуть бути навіть різними. Важливо, що мітки потрапляють в єдину точку, що відповідає за прийом відповідного події – мітки.

Надалі інформація від всіх об'єктів, що підсумовують мітки, систематизується аналітичним блоком. У ньому полягає основна логіка. Там знаходиться AI, Machine Learning, тощо. Результат роботи з аналітичного блоку передається в блок презентації для відображення користувачеві. Це може виглядати, як відправка повідомлення на мобільний пристрій, графік на WEB або інше [5].

Оскільки IoT система є розподіленою і пов'язана ненадійним каналом зв'язку, доводиться мати механізми гарантованої доставки інформації. У тому випадку, якщо не вдається передати інформацію від переферійного пристрою в хмару, здійснюються повторні спроби передачі. Те саме має відбуватися і в іншу сторону. Для цих цілей вводиться блок віртуального уявлення периферійного пристрою, в який записується інформація для передачі периферійних пристроїв або новий його стан. Часто це просто текстовий файл, але може бути і більш точна модель уявлення. Зміни в модулі віртуального уявлення можуть бути ініційовані з різних модулів вхідного ланцюга.

Після того як ми розібрали фізичну блок схему IoT архітектури, можна розглянути її логічну схему (рисунок 1.2):

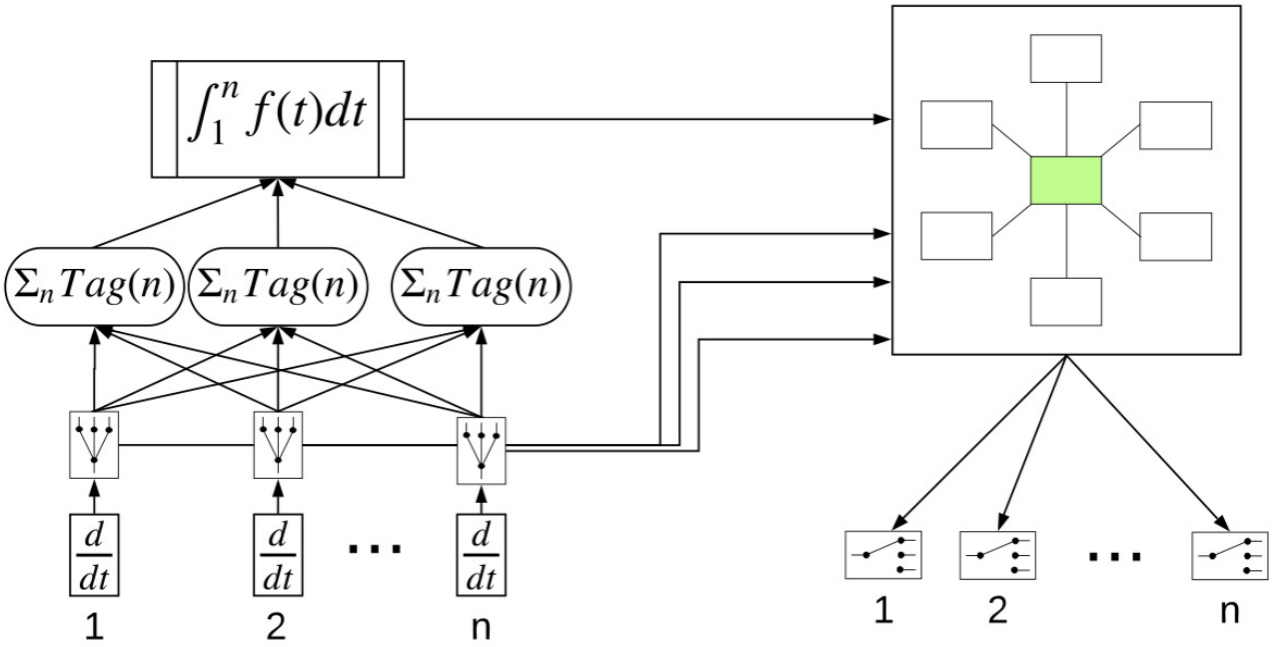


Рисунок 1.2 – Логічна схема IoT

Отже, все знову починається з датчиків, які реєструють зміну навколишнього середовища в часі. Наступний модуль міткування виробляє первинне сегментування на певні події системи. В принципі розробка архітектури IoT додатки і повинна починатися зі списку цих подій. Підсумовування міток проводиться по групі периферійних пристроїв з однотипними мітками. Модуль інтегрування призначений для винесення рішення по апроксимації (передбачення подальших подій) або детермінованість (виявлення ситуації з безлічі варіантів). Ця інформація служить своєрідним ключем–коефіцієнтом для модуля віртуальної моделі периферійного пристрою, в якому актуальна інформація від самого периферійного пристрою перетворюється на підставі ключа–коефіцієнта в новий стан периферійного пристрою.

Тепер трохи про те, що не потрапило в схему архітектури описаної вище, але безумовно про що варто мати на увазі це зберігання інформації має відбуватися, як на рівні периферійного пристрою, так і в хмарі. Периферійний пристрій зберігає свою програму, настройки, стан та тимчасово зберігає

інформацію від датчиків, поки вона гарантовано не передана в хмару. Безпека та авторизація повинно мати кожний периферійний пристрій який може авторизуватися в системі, причому індивідуально.

1.2 Як працюють мережі?

Інтернет (Interconnected networks) тобто з'єднані мережі – це з'єднання мереж в одне, що б кожен користувач міг отримати інформацію або поділитися нею. Кожна з цих мереж, в свою чергу складається з хостів (Host), які об'єднані кабелями, роутерами, світча і т.д., а host – це просто назва пристрою яке підключено до мережі. Тобто за допомогою мільйона цих самих мереж йде об’єднання цих host один з одним, і на виході цих об'єднань виходить інтернет. Не так давно host мали desktop і сервери, але на даний час все це змінилося і до мережі підключені навіть холодильники не кажучи вже про телевізори і про автомобілі та смартфони.

Всі ці хости або кінцеві системи з'єднай між собою (роутерами чи іншими пристроями), які в свою чергу з'єднай різними типами проводів наприклад (крученої пари, оптоволокна), основна мета всіх цих з’єднань хостів, в єдину мережу – це обмін даними. Спочатку інтернет створювався для військових цілей, але потім виявилося що це, куди більша технологія і на даних момент інтернетом користується майже кожна людина на планеті.

Дивлячись на інтернет з боку інженерії, ця система буде цікава своєю складністю архітектури і реалізацією [6].

Дані передаються по мережі за допомогою різного типу кабелів до роутера до іншого роутера, в так званих пакетах (Packet) це сутність, в якій, в середньому близько півтора кілобайт розміру і складається з декількох частин типу дані для роутера, що б той в свою чергу зміг направити пакет до наступного роутеру, в потрібному напрямку його розмір, адреса відправника, адреса одержувача і частина даних яка передається, тобто якщо потрібно

передати по мережі mp3 файл, то він розіб'ється на ці самі пакети і передасться частинами, з точки А в точку В.

Крім хостів, є ще й інші частини які відповідають за об'єднання цих машин за передачу пакетів тощо. На даний момент прийнято використовувати два види пристроїв при необхідності з'єднати кілька комп'ютерів між собою це світч (switch) і роутер (router), раніше так само ще були хаби (hub), але в подальшому від цієї ідеї транспортування даних відмовилися оскільки передана інформація через хаб надходила на всі пристрої підключеному до нього. Схема реалізації переданої інформації через хаб представлена на (рисунке 1.3). Щоб передати інформацію наприклад від Host_1 до Host_2 через хаб, то ця інформація передається і на Host_3 і на Host_4.

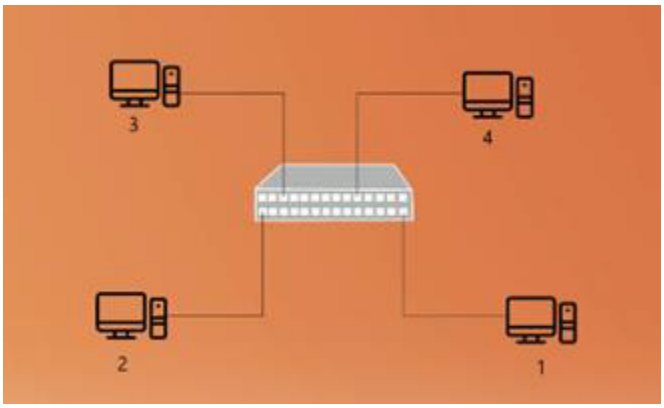


Рисунок 1.3 – Реалізація передачі інформації через hub

Світчі (switch) такої проблеми не мають, як хаби, бо світч має свою mac–адресу і може керувати хостами підключеними до нього, за допомогою запам'ятовування mac–адрес тобто хто є хто.

Роутер, або маршрутизатор, виконує ту ж задачу – об'єднує кілька комп'ютерів в одну мережу, у нього так само є порти для підключення пристроїв, але крім того він ще й забезпечує доступ в інтернет. Маршрутизатор, в наш час, є найпопулярнішим і продуманим пристроєм для побудови невеликої, домашньої мережі, так як в ньому враховані і виправлені всі

недоліки попередників. Слід відзначити той факт, що більша частина роутерів комплектується wi–fi антенами для підключення до інтернету бездротових пристроїв.

У мережі є крім mac–адресу ще й IP–адреса проводячи аналогію відмінності цих двох адрес, IP–адреса динамічний і може змінюватися з часом, а mac–адресу є унікальним для кожного пристрою в локальній мережі.

Навіщо потрібні всі ці адреси, наприклад заходячи на сайт і отримуючи інформацію від сайту, інформація приходить на цей пристрій завдяки лише двом цим адресам одне з яких є унікальним, оскільки з даного IP–адреси можуть зайти і інші пристрої і отримувати інформацію з інтернету.

1.3 Web–сервер

Поняття веб–сервера можна розшифрувати з двох позицій. З точки зору апаратної частини, веб–сервер можна описати як пристрій, який зберігає в собі різні ресурси, такі як сайти (html структури, css стилі, javascript файли, php сценарії) та інші файли, при цьому надають цим ресурси кінцевим клієнтам (веб–браузер, різні програми, які самостійно звертаються до сервера, мобільні телефони та іншим цифровим пристрої). Як правило пристрій підключено до мережі і відповідно є доступним через доменні імена або IP–адреси. Говорячи про веб–сервер, не можна забувати про розгляд його з точки зору програмної частини. З точки зору програмного забезпечення, веб– сервер – це сервіс необхідний для контролювання доступу до ресурсів, розташованим на сервері, це є HTTP сервером. HTTP–сервер – це програмне забезпечення, яке розуміє URL запити, а також повністю оперує HTTP протоколом.

Якщо сказати все вище перераховане простими словами, то вийде, що якщо наприклад браузеру необхідний будь–який медіа ресурс, розміщений на веб–сервері, браузер робить запит через HTTP. Після того, коли HTTP запит досягає необхідного веб–сервера (апаратна частина), сервер HTTP (програмна

частина) надає запитуваний медіа–ресурс браузеру, так само через HTTP протокол.

Для завантаження веб–сторінки, браузер створює і відправляє запит до веб–сервера, який в свою чергу ініціалізує процедуру пошуку необхідного файлу в своєму розміченому під зберігання розділі пам'яті. Знайшовши файл, сервер виробляє процедури зчитування, далі йде обробка файлу, так як необхідно серверу, наступним етапом сервер направляє файл до браузеру клієнта. В цьому і полягає схема роботи веб–сервера, яка відображена на (рисунке 1.4).

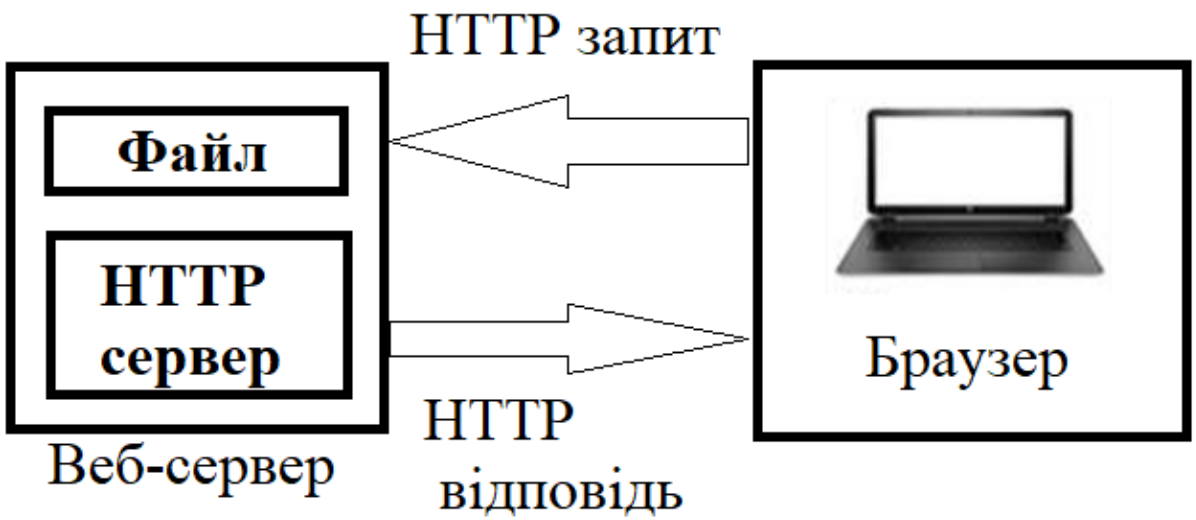


Рисунок 1.4 – Проста схема роботи веб–сервера

Веб–сервер, здійснює зберігання всіх файлів веб–сайту, тобто всі HTML документи, CSS стилі, шрифти, зображення, відео, javascript, php, xml і багато інших файли [4].

Можна зберігати всі перераховані вище файли на своєму комп'ютері, але існує кілька величезних плюсів спеціально виділених під це веб–серверів, а саме:

– постійна і безперервна робота пристрою;

– постійна присутність в мережі Інтернет;

– статичний IP адреса;

– обслуговування на стороні сервера.

Крім усього вищевикладеного веб–сервер володіє різними додатковим функціоналом, наприклад:

– веде автоматизований роботу веб–сервера;

– веде обліковий журнал звернень клієнтів до ресурсів;

– виробляє аутенфікацію і авторизацію користувачів;

– підтримує динамічно генеруються сторінки;

– підтримує протокол безпеки HTTPS, для створення захищеного з'єднання з клієнтом.

1.4 Розгляд OSI and TCP/IP

У 1982 році міжнародною організацією зі стандартизації (International Organization for Standardization, ISO) за підтримки ITU–T було започатковано новий проект в галузі мережевих технологій, який був названий Open System Intercommunication (OSI). Ця модель є першим кроком до міжнародної стандартизації протоколів, використовуваних на різних рівнях 1983 році. Потім вона була переглянута в 1995 році. Модель OSI встановлює глобальний стандарт, який визначає склад функціональних рівнів при відкритій взаємодії між комп'ютерами.

У моделей OSI і TCP є багато спільних рис. Обидві моделі засновані на концепції стека незалежних протоколів. Функціональність рівнів також багато в чому схожа. Наприклад, в кожної моделі рівні, починаючи з транспортного і вище, надають наскрізну, не залежну від мережі транспортну службу для процесів, які бажають обмінюватися інформацією. Ці рівні утворюють постачальника транспорту. Також в кожній моделі рівні вище транспортного є прикладними споживачами транспортної служби.

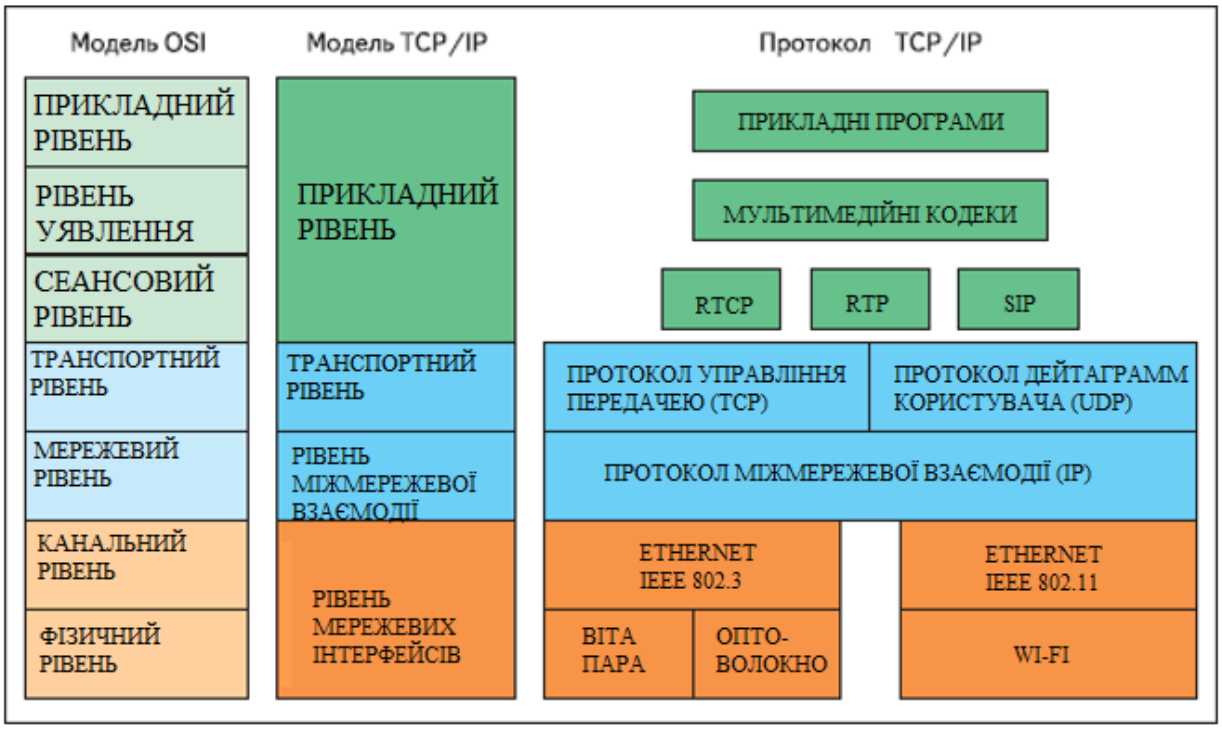


Рисунок 1.5 – Схема моделей OSI and TCP/IP

Прикладний рівень: забезпечує перетворення даних, специфічних для кожної програми. Відповідає за доступ додатків в мережу такі протоколи як: HTTP, gopher, NTP, Telnet, DNS, SNMP, CMIP, FTP, TFTP, SSH, IRC, AIM, NFS, NNTP, ITMS, ModbusTCP, BACnetIP, IMAP, POP3, SMB, MFTP, BitTorrent, eD2k.

Рівень представлення: здійснює перетворення даних загального характеру (кодування, компресія і т.п.) прикладного рівня в потік інформації для транспортного рівня. Відповідає за можливість діалогу між додатками на різних машинах. Протоколи: HTTP, XML–RPC, FTP, TDI, XDR, SNMP, Telnet, SMTP, NCP, AFP [11].

Сеансовий рівень: додає транспортної функції зручності користування, управляє діалогом протягом встановленої сесії зв'язку.

Транспортний рівень виконує вільну від помилок, орієнтовану на роботу з повідомленнями наскрізну передачу. Ділить потоки інформації на досить малі фрагменти (пакети) для передачі їх на мережевий рівень за це відповідають

протоколи: TCP, UDP, NetBEUI, AEP, ATP, IL, NBP, RTMP, SMB, SPX, SCTP, DCCP, RTP, TFTP.

Мережевий рівень: забезпечує маршрутизацію і управління завантаженням каналу передачі, надає необроблений маршрут передачі, що складається лише з кінцевих точок. Відповідає за розподіл користувачів на групи. На цьому рівні відбувається маршрутизація пакетів на основі перетворення mac–адресу в мережеві адреси. Мережевий рівень забезпечує також прозору передачу пакетів на транспортний рівень. І за це відповідають такі протоколи як: IP, IPv6, ICMP, IGMP, IPX, NWLink, NetBEUI, DDP, IPSec, ARP, RARP, DHCP, BootP, SKIP, RIP.

Канальний рівень: здійснює вільну від помилок передачу по окремому каналу зв'язку. Забезпечує створення, передачу і прийом кадрів даних. Цей рівень обслуговує запити мережевого рівня і використовує сервіс фізичного рівня для прийому і передачі пакетів. Специфікації IEEE 802.x ділять канальний рівень на два підрівні: управління логічним каналом (LLC) і управління доступом до середовища (MAC). LLC забезпечує обслуговування мережного рівня, а підрівень MAC регулює доступ до поділюваного фізичного середовища.

Фізичний рівень: виконує реальну фізичну передачу біт даних. Отримує пакети даних від вищого канального рівня і перетворює їх в оптичні або електричні сигнали, відповідні 0 і 1 бінарного потоку. Ці сигнали посилаються через середовище передачі на прийомний вузол. Механічні і електричні/оптичні властивості середовища передачі визначаються на фізичному рівні і включають:

– тип кабелів і роз'ємів;

– розведення контактів в роз'ємах;

– схему кодування сигналів для значень 0 і 1.

Модель TCP / IP називають також моделлю DARPA (скорочення від Defense Advanced Research Projects Agency, організація, в якій свого часу розроблялися мережеві проекти, в тому числі протокол TCP/IP, і яка стояла біля

витоків мережі Інтернет) або моделлю Міністерства оборони США (модель DoD, Department of Defense, проект DARPA працював на замовлення цього відомства).

Модель TCP/IP розроблялася для опису стека протоколів TCP/IP (Transmission Control Protocol / Internet Protocol). Вона була розроблена значно раніше, ніж модель OSI – в 1970 році було розроблено необхідний набір стандартів, а до 1978 року остаточно оформився те, що сьогодні ми називаємо TCP/IP. Пізніше стек адаптували для використання в локальних мережах. На початку 1980 р протокол став складовою частиною ОС UNIX. У тому ж році з'явилася об'єднана мережа Internet.

Стек протоколів TCP/IP – набір мережевих протоколів, на яких базується інтернет. Зазвичай в стеці TCP/IP верхні 3 рівня (прикладний, представницький і сеансовий) моделі OSI об'єднують в один – прикладний. Оскільки в такому стеку не передбачено уніфікований протокол передачі даних, функції з визначення типу даних передаються з додатком.

На відміну від еталонної моделі OSI, модель TCP/IP більшою мірою орієнтується на забезпечення мережевих взаємодій, ніж на жорстке розділення функціональних рівнів. Для цієї мети вона визнає важливість ієрархічної структури функцій, але надає проектувальникам протоколів достатню гнучкість в реалізації. Відповідно, еталонна модель OSI набагато краще підходить для пояснення механіки між комп’ютерних взаємодій, але протокол TCP/IP став основним між мережевим протоколом.

1.5 Raspberry Pi

Raspberry Pi – одно платний комп’ютер розмірами приблизно з банківську карту, на початку створений як економна система для вивчення інформатики, після чого отримав набагато більше і широке застосування і популярність, ніж очікували його творці (рисунок 1.6).

Рисунок 1.6 – Зразок Raspberry Pi

Одно платний комп'ютер – самодостатній комп'ютер, зібраний на одній друкованій платі, на якій поставлені процесор, оперативна пам'ять, системи введення–виведення і інші модулі, необхідні для функціонування мікрокомп'ютера. Одно платні ПК виробляються в якості демо систем, систем для програмістів або освіти, або ж для застосування в ролі промислових або ж вбудованих комп'ютерів.

На відміну від класичних ПК форм–фактора «desktop» (стандарти AT, ATX, і т.п.), Одно платні комп'ютери нерідко просять установки якихось додаткових периферійних плат. Деякі одно платні системи представлені у вигляді малогабаритної плати з мікропроцесором і пам'яттю, що підключаються до backplane для розширення можливостей, наприклад, для нарощування числа вільних роз'ємів.

Найчастіше ці роз'єми повинні бути захищені або ж досить компактні, в наслідок цього всі складові повинні перебуває на одній платі.

Ця ощадливість з одного боку готує весь прилад більше малогабаритним, з іншого боку, зміна мікропроцесора або ж пам'яті – складне, тому що як правило всі ці складові напаяні на плату. Raspberry існує в таких моделях:

«A» (Процесор ARM1176JZ–F 700MHz, 256 Мб ОЗП, 26 пінів GPIO, 1 USB порт), вартістю $20.

«A+» (Процесор ARM1176JZ–F 700MHz, 256 Мб ОЗП, 40 пінів GPIO, 1 USB порт), вартістю $25.

«B» (Процесор ARM1176JZ–F 700MHz, 512 Мб ОЗП, 26 пінів GPIO, 2 USB порта, с ethernet), вартістю $35.

«B+» (Процесор ARM1176JZ–F 700MHz, 512 Мб ОЗП, 40 пінів GPIO, 4 USB порта, с ethernet), вартістю $30.

«2B» (Процесор ARM Cortex–A7 900MHz 4–ядерний, 1 Гб ОЗП, 40 пінів GPIO, 4 USB порта, с ethernet) вартістю $35.

«Zero» (Процесор ARM1176JZ–F 1GHz, 512 МБ ОЗП, 40 пінів GPIO, 1 USB порт) вартістю $5.

«3» (Процесор ARM Cortex–A53 1,2GHz 4–ядерний 64–розрядний , 1 Гб ОЗП, 40 пінів GPIO, 4 USB порта, ethernet, Wi–Fi 802.11n и Bluetooth 4.1) вартістю $35.

«4B» (Процесор ARM Cortex–A72 1,5GHz 4–ядерний 64–розрядний , 4 Гб ОЗП, 40 пінів GPIO, 4 USB порта, ethernet, Wi–Fi 802.11n и Bluetooth 4.1) вартістю $100.

1.5.1 Технічні характеристики Raspberry.

Комп'ютер поширюється повністю зібраними на чотиришаровій друкованій платі розміром з банківську карту. У звичайний набір поставки входить лише тільки сама плата. Корпус, блок живлення, флеш–карту потрібно замовляти додатково (рисунок 1.8).

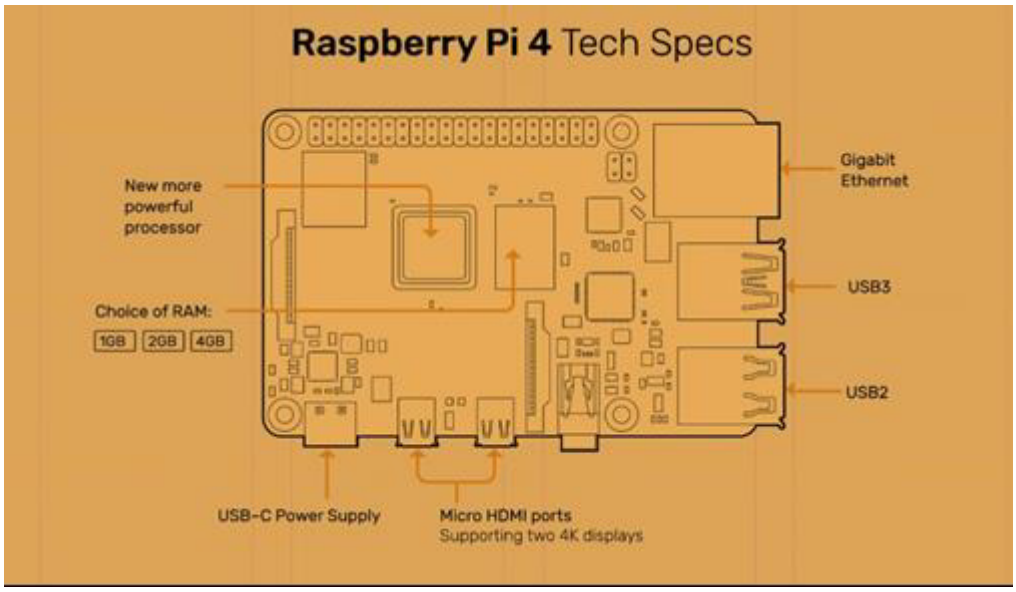


Рисунок 1.8 – Технічні характеристики Raspberry PI 4 Model B

Raspberry Pi випускається в декількох комплектаціях: модель «A», модель «B», модель «B +» і модель «2B». 1–і 3 версії комплектувати ARM11 мікропроцесором Broadcom BCM2835 з тактовою частотою 700МГц і модулем оперативної пам'яті на 256МБ / 512МБ, розміщеними за технологією «package–on–package» саме на процесорі. Модель «2B» оснащується процесором з 4 ядрами Cortex–A7 з частотою 1 ГГц і оперативною пам'яттю об'ємом 1 ГБ. Модель «A» оснащується одним USB 2.0 портом, модель «B» 2–мя, а моделі «B +» і «2 B» – 4. Ще в моделях «B», «B +» і «2B» наявний порт Ethernet. Крім головного ядра, BCM2835 підключає в себе графічне ядро за допомогою OpenGL ES 2.0, апаратного прискорення і FullHD–відео і DSP–ядро. Однією з особливостей вважається відсутність годин реального часу.

Однією з найбільш індивідуальностей Raspberry Pi вважається присутність портів GPIO (general purpose input / output). За допомогою даного raspberry комп'ютера його можна застосувати для управління різними приладами. У моделі «B» плати комплектувати 26 портами, а в моделі «B +» і «B2» – 40 портів GPIO.

Raspberry Pi функціонує в основному на операційних системах, заснованих на Linux ядрі. Запуск Windows можливий завдяки засобам

віртуалізації таким, як XenDesktop. ARM11 реалізований на 6 версії ARM, на якому кілька відомих версій Linux не запускається. Для установки операційних систем є інструмент NOOBS.

Вбудований прилад для зчитування карт пам'яті гарантовано функціонує з більшістю SD–карт об'ємом до 32 Гбайт. Завантажуватися Raspberry Pi лише тільки з карток SD. У разі якщо сама ОС знаходиться на USB–накопичувачі, але ось завантажувач зобов'язаний бути на SD. Кнопок підключення і скидання немає – прилад сам включається при подачі живлення. Живиться Raspberry Pi від порту micro–USB або ж з пари виділених портів GPIO. Для Model A рекомендовано живлення на 5В і 500–700 мА, а для Model B на 5 В і 700–1200 мА. Підведення живлення можливе за допомогою USB 3.0 або ж зарядного пристрою для телефонної апаратури, ніж від будь–якого іншого більш стабільний джерела живлення.

Самі плати споживають трохи менше, але частка енергії буде потрібно для роботи приєднаних до USB–портів приладів.

2 МЕТОДОЛОГІЯ РОЗРАХУНКІВ ПРОДУКТИВНОСТІ МЕРЕЖ

2.1 Опис проекту та типи запитів

Наприклад, є велика компанія в якої спрямован бізнес на прокат електротранспорту, в даній роботі буде розглядатися вид електротранспорту такий як електросамокат. Цей електронний вид засобу пересування розташований в центрі міста або на його околицях, для реалізації прокату не потребує персоналу або встановлених місць прокату даного виду обслуговування, але має чіткі межі пересування по місту, які відстежуються за допомогою GPS датчиків (рисунок 2.1).

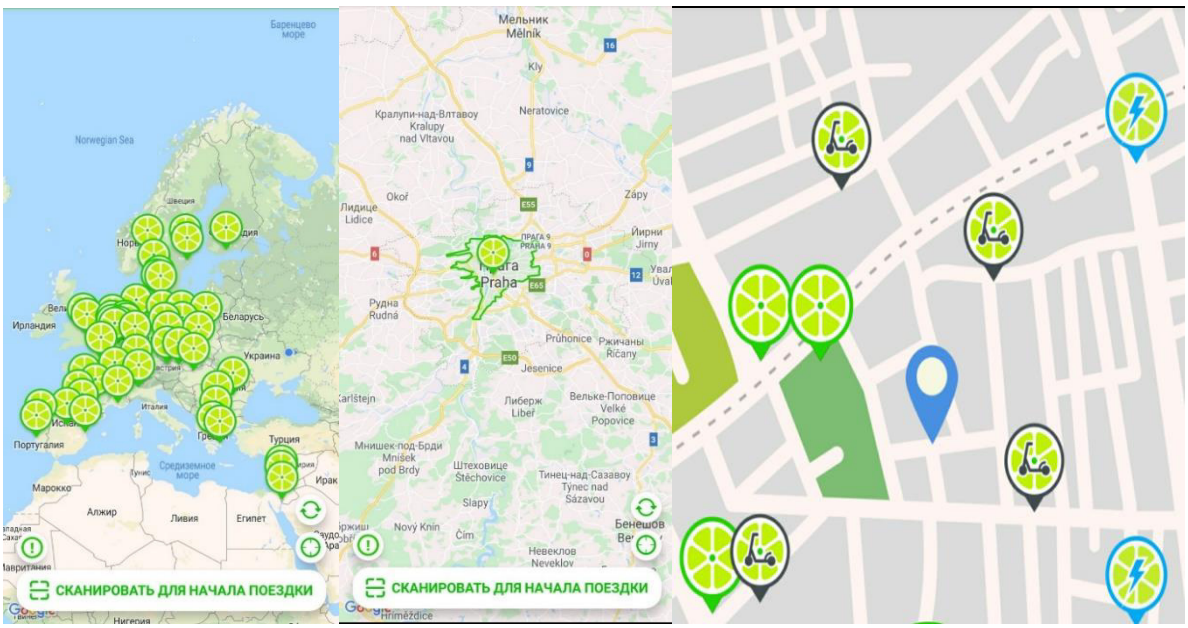


Рисунок 2.1 – Реалізація пошуку електросамокату

Для використання даного виду транспорту цієї компанії, користувачеві потрібно встановити на свій смартфон відповідний додаток, після чого користувач підтверджує свої данні та ознайомлюються з правилами

використовування даної продукції, відкриває свій рахунок для оплати послуг чи згоджується з оплатою після закінчення поїздки (рисунок 2.2).

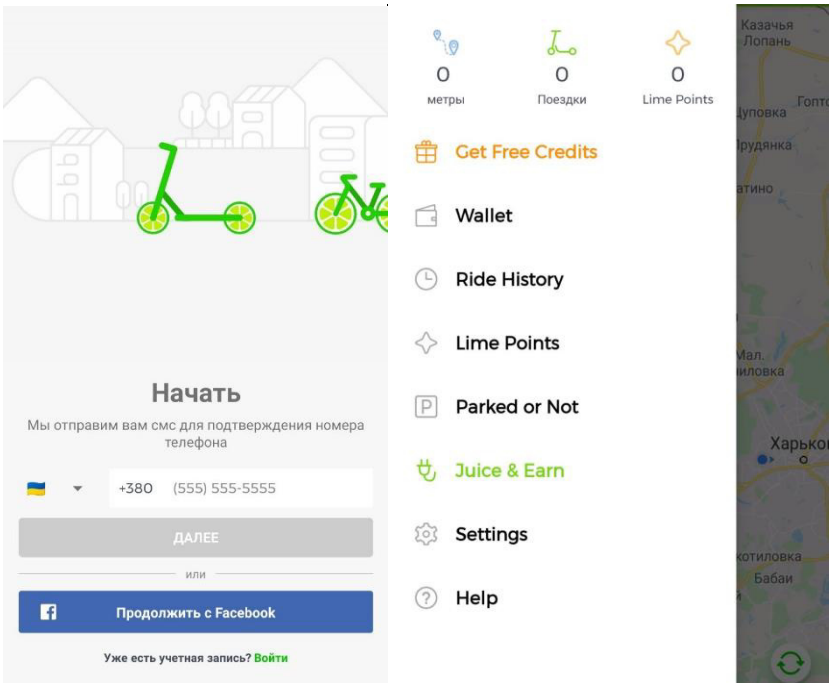


Рисунок 2.2 – Реалізація автентифікації користувача та інтерфейс програми

Далі користувач сканує через даний додаток QR штрих код електросамокату (рисунок 2.3), де відображується стан транспорту (заряд батареї та приблизна відстань яку зможе проїхати клієнт), після всіх цих операцій клієнт може пересуватися куди захоче та залишити транспорт де захоче в відповідної зоні міста, якщо клієнт виїхав за відповідну зону обслуговування то спрацьовує блокування транспорту, яка дає клієнту відповідні сигнали та поступово блокує колеса, клієнт не зможе пересуватися за відповідну зону дії, йому потрібно буде повернути транспорт в відповідну зону дії бо кошти будуть далі списуватися з картки власника, гроші списуються тоді коли користувач підтвердить що він хоче закінчити поїздку, після чого кошти списуються с банківської картки вже без підтвердження власника, бо власник картки заздалегідь заповнив всі поля регістрації та згодився з правилами даної програми.

Запити обробляється сервером, та відправляється на пристрій електротранспорту за розподіленою моделі (клієнт – сервер). Аналіз було проведено на реалізації сервісу IoT, що розгорнутий на інфраструктурі DigitalOcean, який може бути реалізований на транспортному рівні протоколом MQTT і HTTP.

Рисунок 2.3 – Реалізація бронювання електросамокату

Нижче наведено типи запитів кожен з наведених типів може мати різний об’єм даних) що приймають участь у повному циклі від запуску додатку на смартфоні, до закінчення поїздки на електросамокаті:

1 тип – багатофакторна автентифікація користувача;

2 тип – пошук вільного електросамокату на карті;

3 тип – сканування даного електросамокату та зчитування його параметрів батареї;

4 тип – підтвердження платіжної спосібності клієнта;

5 тип – час використання пристрою;

6 тип – відстеження та запис маршруту пристрою за допомогою GPS–датчиків;

7 тип – закінчення поїздки та оплата.

2.2 Імітаційне моделювання для вирішення адміністративних завдань систем масового обслуговування (СМО)

Імітаційне моделювання – це розробка і виконання на комп'ютері програмної системи, що відображує структуру і функціонування об'єкта, що моделюється або явища в часі. Таку програму називають імітаційної моделлю цього об'єкта або явища. Об'єкти і сутності імітаційної моделі представляють об'єкти і сутності реального світу, а зв'язку структурних одиниць об'єкта моделювання відображаються в інтерфейсних зв'язках відповідних об'єктів моделі. Таким чином, імітаційна модель – це спрощене подобу реальної системи, або існуючої, або тієї яку передбачається створити в майбутньому. Імітаційна модель зазвичай представляється комп'ютерною програмою, виконання програми можна вважати імітацією поведінки вихідної системи в часі [13].

Імітаційне моделювання вимагає проведення серії обчислювальних експериментів і їх статистичної обробки.

У даній роботі, наводиться опис декількох інформаційних систем імітаційного моделювання і порівняння їх можливостей, також було проведено порівняння результатів імітаційного моделювання отриманих в різних системах. У підсумку вийшло, що освоєння систем імітаційного моделювання систем масового обслуговування не складно, їх застосування дозволяє значно підвищити ефективність проведених досліджень за рахунок зниження трудомісткості виконуваних робіт і підвищення достовірності результатів моделювання.

В роботі, розглянута імітаційна середовище AnyLogic 8.5, її можливості для побудови імітаційних моделей СМО. У даній роботі була обрана система AnyLogic для вирішення постановленої завдання. Вибір даної системи заснований на ряд її переваги, а саме:

– об'єктно підхід моделювання;

– інтеграція і вбудована Java–середовище програмування;

– зручний і зрозумілий графічний інтерфейс;

– багатий і зручний довідковий матеріал.

Постановка завдання полягає в тому щоб побудувати імітаційну модель файлового сервера організації, в котрому мережевий трафік має пульсуючій характер (пульсуючій) відносіться до того факту, що дані передаються випадкове чином, з піковими швидкостями, що перевищують середні швидкості в 7–10 разів. Також вважається, що мережевий трафік пульсує в декількох масштабах часу. Припустимо, що фірма обслуговує 1 млн клієнтів. Навантаження має піковий характер бо найбільша кількість користування послугами даної компанії відбуваються з 8:00 до 9:00 та з 11:00 до 16:00. З цього слідує, що пікове (пульсуюче) навантаження кожного дня складає 7 годин.

Пропускна здатність сервера, виражена в запитах/с, складатиме за формулою (2.1–2.2):

$$\frac{\text{максимальна кількість запитів}}{\text{днів} \cdot \text{кількість пікових годин} \cdot \text{переведення хвилин в секунди}}, \qquad (2.1)$$

$$\frac{1000000}{7 \cdot 7 \cdot 3600} = 5{,}66 \left(\frac{\text{запитів}}{c}\right). \qquad (2.2)$$

Однак, цей показник не дає ніякого представлення щодо пропускної здатності мережі, що використовується в період спостереження і не дає представлення про розміри типів запитів. Таким чином, для оцінки пропускної здатності мережі чи мережевого адаптера потрібно розрахувати пропускну здатність в біт/с. Для цього потрібно віртуально імітувати модель передачі даних на сервер.

Обробляє запити надходять з пристроїв цієї організації, інтервали між перебуванням запитів розподілені за законом Пуассона (формула 2.3) з різною інтенсивністю запитів за часом.

$$f(x) = \begin{cases} 0, x < 0 \\ e^{-\lambda} \sum_{i=0}^{|x|} \frac{\lambda^i}{i!}, \ x \geq 0 \end{cases}. \quad (2.3)$$

Час обслуговування запитів файловим сервером розподілене по експоненціальному закону з середнім значення 2 секунди. Ємність вхідного буфера файлового сервера не обмежений. Дана система представляється собою багатоканальну систему масового обслуговування з обмеженою чергою (з відмовами). При заповненої черги запити залишають чергу через вихід outPreempted, тобто витісненим іншим вхідним запитом, потім повертаються користувачу з найменуванням помилки системи і повторної операцією. Структурна схема даної системи показана на (рисунке 2.5):

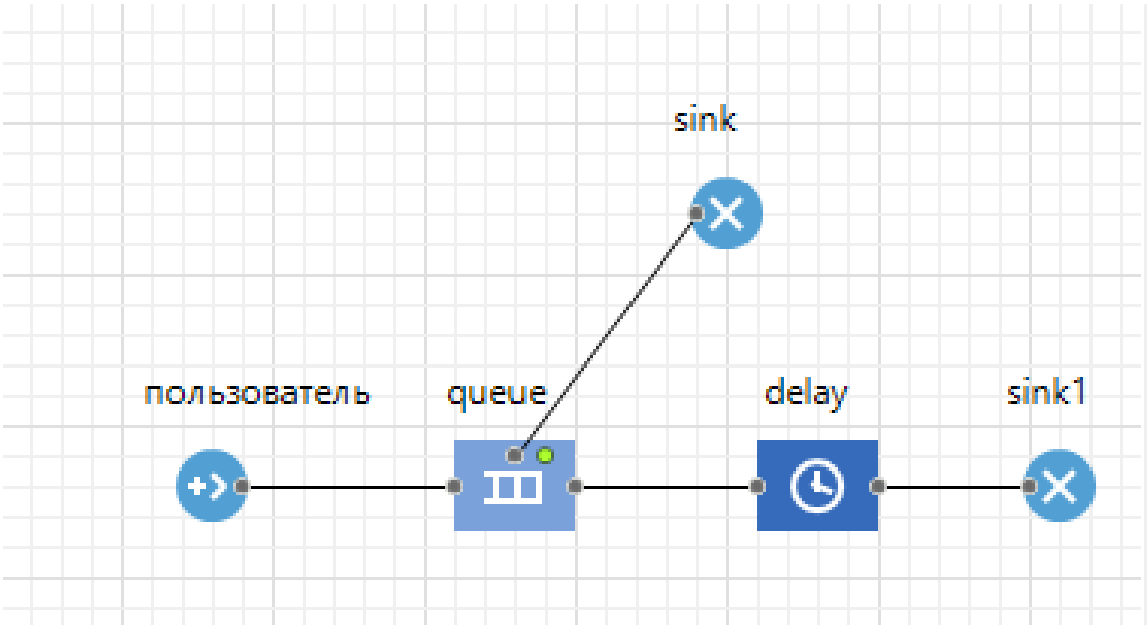


Рисунок 2.5 – Структурная схема системи построенная в программе AnyLogic

Розглянемо логіку функціонування імітаційної моделі. Сформовано клас користувач який подає запити на клас queue, який виконує модельне обчислення навантаження потоків і розподіляє пропускати пакет інформації далі або зберегти його в кеші і дочекатися поки клас delay, який обробляє

запити, звільниться від навантаження і в нормальному режимі зможе обробити такий запит, клас sink1 виконує роль вихідного потоку, який передає оброблену інформацію. Знаходження заявок в черзі реалізується за допомогою формування динамічної структури – колекція, де знову надійшла заявка додається в колекцію. При звільненні обслуговуючий пристрій (об'єкт delay), заявка починає обслуговуватися і елемент з порядковим номером видаляється з колекції, за моделлю FIFO.

Основним моментом при розробки імітаційних моделей є встановлення ефективності цієї моделі поведінки. В даному випадку, для встановлення ефективності використовуємо той факт для СМО з обмеженим часом очікування, рівним (const), для стаціонарного процесу існує аналітичний вираз інтенсивності виходить потоку (оброблених заявок) розрахована за (формулою Баррера 2.4):

$$\lambda_0 = p_0 \frac{\lambda^2}{\mu} e^{\tau(\lambda-\mu)}, \qquad (2.4)$$

де $P_0^{-1} = \begin{cases} 1 + \frac{\lambda}{\mu}\frac{\lambda e^{\tau(\lambda-\mu)}-\mu}{\lambda-\mu}, & \lambda \neq \mu \\ 2 + \lambda\tau, & \lambda = \mu \end{cases}$,

τ– обмеження часу очікування,

λ– інтенсивність найпростішого вхідного потоку,

μ– інтенсивність експоненціального закону, що описує час обслуговування.

В таблицях 3.1 та 3.2 наведено приблизні розміри повідомлень для різного типу запитів, які будуть використовуватися, в даній дипломній роботі для розрахунку най вдалого вибору протоколу. Використовуючи дані масиви з різними протоколами розраховуємо їх в системі СМО за формулою Баррера і відображаємо отримані результати графічним шляхом для наочності (рисунок 2.6), результати розрахунків для протоколу HTTP складають: а) $\lambda = \mu = \tau =$

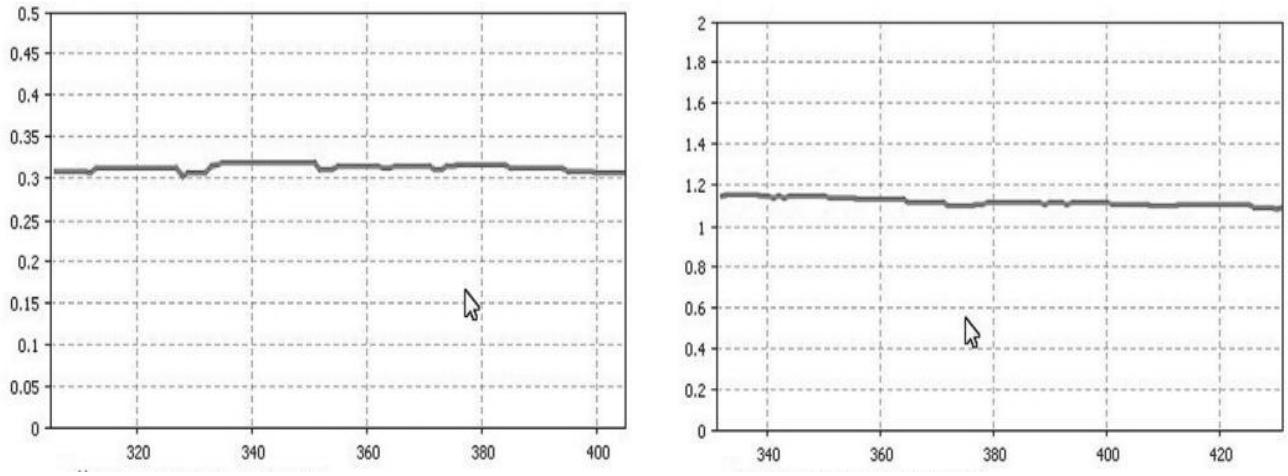
$1 - \lambda_0 = 0{,}35$, а для протоколу MQTT становить: б) $\lambda = 2,\ \mu = \tau = 1 - \lambda_0 = 1{,}09$

Таблиця 2.1 – Варіації розміру повідомлень при використанні HTTP

	Розміри повідомлень, Біт									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	1290	2290	6290	8290	9290	11290	16290	20290	3620	40290
2	5290	7290	12290	15290	18290	20290	25290	30290	5020	77290
3	10290	12290	16290	17290	19290	22290	24290	26290	30290	33290
4	3290	6290	10290	12290	14290	17290	20290	28290	33290	100290
5	100290	110290	120290	140290	165290	165290	165290	165290	260290	270290

Таблиця 2.2 – Варіації розміру повідомлень при використанні MQTT

	Розміри повідомлень, Біт									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	1002	2002	6002	8002	9002	11002	16002	20002	36002	40002
2	5002	7002	12002	15002	18002	20002	25002	30002	50002	77002
3	1002	12002	16002	17002	19002	22002	24002	26002	30002	33002
4	3002	6002	10002	12002	14002	17002	20002	28002	33002	100002
5	100002	110002	120002	140002	160002	170002	200002	250002	260002	270002



а) HTTP б) MQTT

Рисунок 2.6 – Графічне представлення інтенсивності виходу потоків протоколу

На підставі аналізу отриманих імітаційних даних можна зробити висновок про ефективного розробленої імітаційної моделі і кращими показниками в опрацюванні запитів протоколу MQTT [2].

Для того щоб прорахувати середній час обробки одного запиту (математичне очікування), яке визначається як відношення сумарного часу обробки всіх запитів N до їх кількості необхідно знати час обробки і–го запиту. Час обробки одного запиту – це час з моменту виходу запиту з блоку source до моменту виходу з системи (входу в блок sink), тобто за наступним відношенням:

$$\text{Час обробки} = \text{час виходу} - \text{час входу}. \qquad (2.3)$$

Додаючи це значення в об'єкт, який запам'ятовує значення часу для кожного запиту, ми отримаємо статичні дані в тому числі середній час обробки одного запиту. Імовірність обробки запитів виражається по (формулі 2.4):

$$\text{Середній час обробки} = \frac{\text{кількість оброблених запитів}}{\text{кількість поступивших запитів}}. \quad (2.4)$$

Додаючи це значення в об'єкт, ми отримаємо середню ймовірність обробки запитів [3]. Для того щоб було зрозуміло, як оптимально використовувати систему, були проведені декілька експериментів змінюючи при цьому довжину черги (QL) і середній час обслуговування запиту (ST), результати імітаційного моделювання наведені в (таблице 2.3).

Таблиця 2.3 – Результати імітаційних моделей спроектована програмою AnyLogic 8.5

Тип запиту №		Кількість оброблених запитів	Імовірність обробки запитів	Середнє час обробки одного запиту	Кількість втрачених запитів
1	QL=100 ST=5	100	0,998	0,476	0
2	QL=100 ST=20	80	0,802	0,933	17
3	QL=5 ST=15	90	0,897	0,568	1
4	QL=20 ST=10	98	0,979	0,470	0
5	QL=100 ST=10	98	0,982	0,480	0

З результатів робимо висновок, що при зменшенні середнього часу обслуговування (ST) і збільшенні довжини черги (QL – пам'ять сервера) збільшується середня ймовірність обробки запитів і кількість оброблених запитів, зменшується середній час обробки запиту і практично відсутні втрачені запити. Помічаємо, що при збільшенні довжини черги (QL) в два або в десять разів, більше не змінюючи при цьому середній час обслуговування (ST), результати майже однакові (експерименти №4 та №5), тобто збільшення

довжини черги на великих відсотках не впливає на результати. При збільшенні середнього часу обслуговування запиту, значно збільшуються середній час обробки і кількість втрачених запитів, а кількість оброблених запитів значно зменшується (експерименти №1 і №2). Виходячи з результатів моделювання, представлений в експерименті №1 режимом роботи сервера, є найоптимальнішим. Для експериментів №1 по №5 на (малюнку 2.6) показана ймовірність обробки запитів. В результаті видно, що кількість оброблених запитів і середній час обробки сильно залежать від середнього часу обслуговування в сервері (ST), тобто від продуктивності сервера.

На (рисунку 2.6) наведено ймовірності того, що кожен з типів запитів відбудеться, дані взяті з (таблиці 2.3):

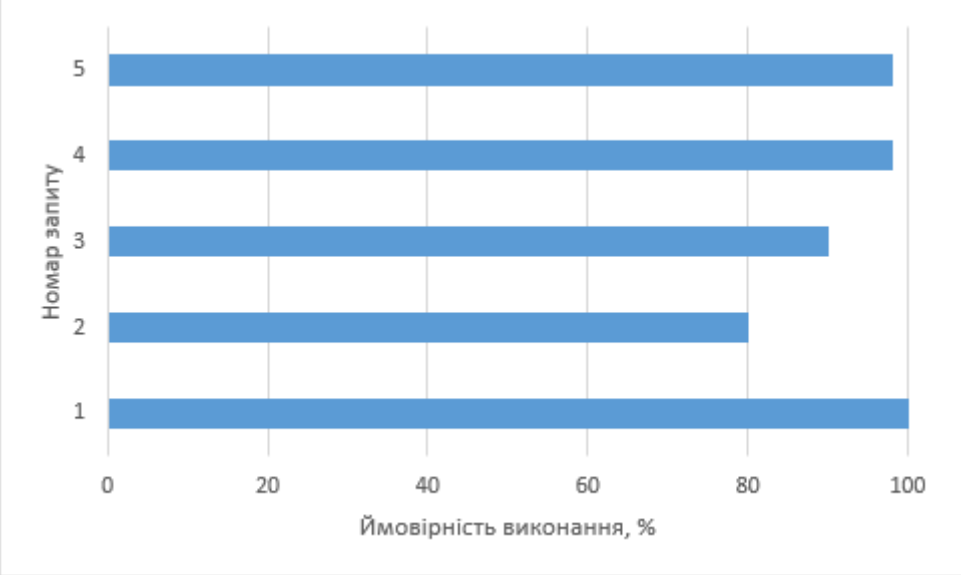


Рисунок 2.6 – Ймовірності виконання запитів

Щоб визначити, яку частку складає кожен з типів, який наведений в (таблиці 2.3) в усій передачі даними, проведемо наступні розрахунки за формулою повної ймовірності (2.5) та запишемо розрахунки до (таблиці 2.4):

$$\frac{B_n}{B_1 + B_2 + (B_2 \cdot B_3) + (B_2 \cdot B_3 \cdot B_4) + (B_2 \cdot B_3 \cdot B_4 \cdot B_5)}. \qquad (2.5)$$

Таблиця 2.4 – Розрахунок частки кожного з типу запитів

№ запиту	1	2	3	4	5
Частка запиту	0,255	0,204	0,184	0,180	0,177

Розрахуємо пропускну здатність Q для кожного типу запиту за (формулою 2.6) та запишемо розрахунки до (таблиці 2.5):

$$Q = \frac{\text{всього запитів} \cdot \text{частка кожного з типів} \cdot \text{середній розмір}}{\text{час спостереження}}. \quad (2.6)$$

Таблиця 2.5 – Розрахунки пропускної здатності для всіх типів першого варіанту

№ запиту	1	2	3	4	5
Пропускна здатність Q, біт/с	14918	48941	85866	26857	805050

Загальна пропускна здатність – це сума пропускних здатностей для кожного з типів розраховуємо за (формулою 2.7):

$$\text{Загальна пропускна здатність} = 14918{,}37 + 48941{,}5 + 85866{,}67 + \\ + 26856{,}14 + 805050 = 981632{,}7 \left(\frac{\text{Біт}}{\text{с}}\right). \quad (2.7)$$

Отже пропускна здатність для першого варіанту складає 981,633 Кбіт/с;

Розрахунки для інших варіантів виконані за аналогією в Excel, результати наведені в (таблицях 2.6–2.7):

Таблиця 2.6 – Пропускна здатність при використанні HTTP

	Пропускна здатність, Біт/с HTTP									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	14918,37	26482,99	72741,5	95870,75	107435,4	130564,6	188387,8	234646,3	419680,3	465938,8
2	48941,5	67444,9	113703,4	141458,5	169213,6	187717	233975,5	280234	465268	715063,9
3	85866,67	102556	135934,7	144279,4	160968,7	186002,7	202692,1	219381,4	252760,1	277794,1
4	26857,14	51346,94	84000	100326,5	116653,1	141142,9	165632,7	230938,8	271755,1	818693,9
5	805049	885321,1	965593,2	1126137	1326818	1366954	1607770	2009131	2089403	2169675
∑	981632,7	1133152	1371973	1608072	1881089	2012381	2398458	2974332	3498867	4447166

Таблиця 2.7 – Пропускна здатність при використанні MQTT

	Пропускна здатність, Біт/с MQTT									
Тип запиту	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
1	11587,76	23152,38	69410,88	92540,14	104104,8	127234	185057,1	231315,6	416349,7	462608,2
2	46277,01	64780,41	111038,9	138794	166549,1	185052,5	231311	277569,5	462603,5	712399,5
3	83463,4	100152,7	133531,4	141876,1	158565,4	183599,5	200288,8	216978,1	250356,8	275390,8
4	24506,12	48995,92	81648,98	97975,51	114302	138791,8	163281,6	228587,8	269404,1	816342,9
5	802737,1	883009,3	963281,4	1123826	1324506	1364642	1605458	2006819	2087091	2167363
∑	968571,4	1120091	1358912	1595012	1868027	1999320	2385397	2961270	3485805	4434104

Таблиця 2.8 – Загальна пропускна здатність при використанні MQTT

	Пропускна здатність, Біт/с									
Тип протоколу	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
MQTT	968571,4	1120091	1358912	1595012	1868027	1999320	2385397	2961270	3485805	4434104
HTTP	981632,7	1133152	1371973	1608072	1881089	2012381	2398458	2974332	3498867	4447166
Δ	1,013485	1,011661	1,009611	1,008188	1,006992	1,006533	1,005475	1,004411	1,003747	1,002946

Використовуючи дані отримані з (таблиці 2.8) будуємо графік відношення різниці пропускної здатності протоколів MQTT і HTTP (рисунок 2.7):

Рисунок 2.7 – Відношення загальної пропускної здатності

З отриманих результатів можна зробити висновок, що мережеве з'єднання повинно мати мінімальну пропускну здатність 5 Мбіт/с, рекомендовано 10 Мбіт/с. А при збільшенні розмірів запитів, різниці між протоколами майже немає. Тому при великих файлах доцільніше використовувати HTTP, бо він не потребує додаткових налаштувань при розгортанні. При передачі запитів, та відповідей невеликих розмірів, доцільніше використовувати MQTT.

2.3 Розрахунок середнього часу відгуку

Визначення затримок, та «вузьких місць» відіграє важливу роль для того, щоб робити зміни в програмному забезпеченні, модернізації обладнання або

установки, використовувати більш швидкісні ліній передачі даних [7]. Затримки можна поділити на 3 масштабні складові:

– центральний процесор відповідальний за обробку даних з пристроїв, а також за ініціалізацію зв'язку;

– мережа породжує затримки під час доставки інформації від клієнта до сервера і назад від сервера до клієнта. Ці затримки є функціями різних компонентів на шляху між клієнтом і сервером, таких як модеми, маршрутизатори, лінії передачі, мости і комутатори;

– час перебування на сервері, R_{server}– це час, що витрачається на виконання запиту. Воно включає в себе час обробки і час очікування на різних компонентах сервера, таких як процесор, диск і мережевий адаптер.

Коли запитуваний документ не перебуває в кеші клієнта, час відгуку R на запит являє собою суму часу перебування запитів на всіх ресурсах:

$$R_{miss} = R_{CPU} + R_{network} + R_{server}, \quad (2.8)$$

де R_{CPU} – час обробки процесором,

$R_{network}$ – час передачі в мережі,

R_{server} – час перебування на сервері.

Розглянемо середній час відгуку для запропонованих протоколів. Припустимо, що документи не кешуються. Середній час перебування на сервері R_{server} – 3,6 с., а час, затрачений процесором на обробку отриманих даних R_{CPU} – 0,3с. Пропускна здатність мережі складає 100 Мбіт/с.

Час передачі в мережі $R_{network}$ дорівнює розміру даних в бітах, поділеному на пропускну здатність мережі за (формулою 2.9–2.11). Продемонструємо розрахунки для першого варіанта. Розрахунки для інших варіантів було проведено в програмі Excel, результати наведено в (таблиці 2.6)

$$R_{network}(\text{HTTP}, MQTT) = \frac{\text{розмір даних протоколів}}{\text{пропускна здатність мережи}}, \quad (2.9)$$

$$R_{network}(\text{HTTP}) = \frac{120450 \cdot 8}{100000000} = 0{,}009639 \text{ (сек)}, \quad (2.10)$$

$$R_{network}(\text{MQTT}) = \frac{119010 \cdot 8}{100000000} = 0{,}009521 \text{ (сек)}, \quad (2.11)$$

$$\text{R}_{\text{miss}}(\text{HTTP}) = 0{,}009639 + 0{,}3 + 3{,}6 = 3{,}909636 \text{ (сек)}, \quad (2.12)$$

$$\text{R}_{\text{miss}}(\text{MQTT}) = 0{,}009521 + 0{,}3 + 3{,}6 = 3{,}909521 \text{(сек)}, \quad (2.13)$$

$$\Delta = \frac{\text{R}_{\text{miss}}(\text{HTTP})}{\text{R}_{\text{miss}}(\text{MQTT})}, \quad (2.14)$$

$$\Delta = \frac{3{,}909636}{3{,}909521} = 1{,}000029467. \quad (2.15)$$

Таблиця 2.9 – Середній час відгуку

Тип протоколу	Варіант1	Варіант2	Варіант3	Варіант4	Варіант5	Варіант6	Варіант7	Варіант8	Варіант9	Варіант10
HTTP	3,9096	3,9110	3,9132	3,9154	3,9181	3,9193	3,9229	3,9284	3,9328	3,9417
MQTT	3,9095	3,9109	3,9131	3,9153	3,9180	3,9192	3,92280	3,92832	3,9327	3,94160
Δ	1,000029467	1,00002945 6	1,00002943 9	1,00002942 3	1,00002940 3	1,000029394	1,00002936 7	1,00002932 6	1,00002929 3	1,00002922 7

Використовуючи дані з (таблиці 2.9) будуємо гістограму середнього часу відгуку для протоколів MQTT і HTTP (рисунок 2.8):

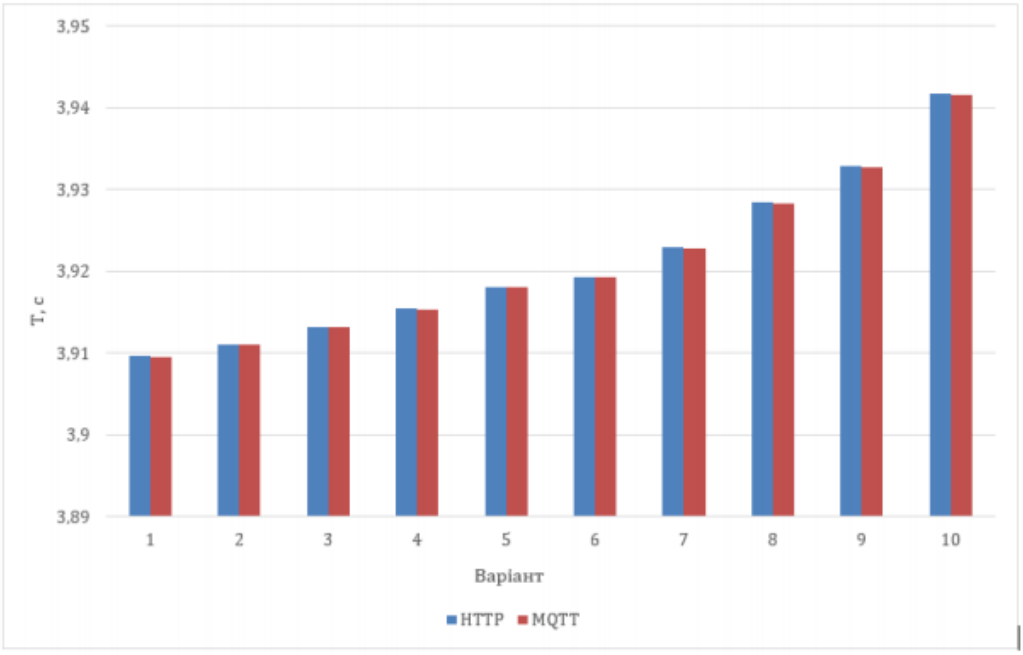


Рисунок 2.8 – Порівняння середнього часу відгуку для протоколів

Використовуючи дані з (таблиці 2.9) будуємо гістограму різниці для протоколів MQTT і HTTP (рисунок 2.8):

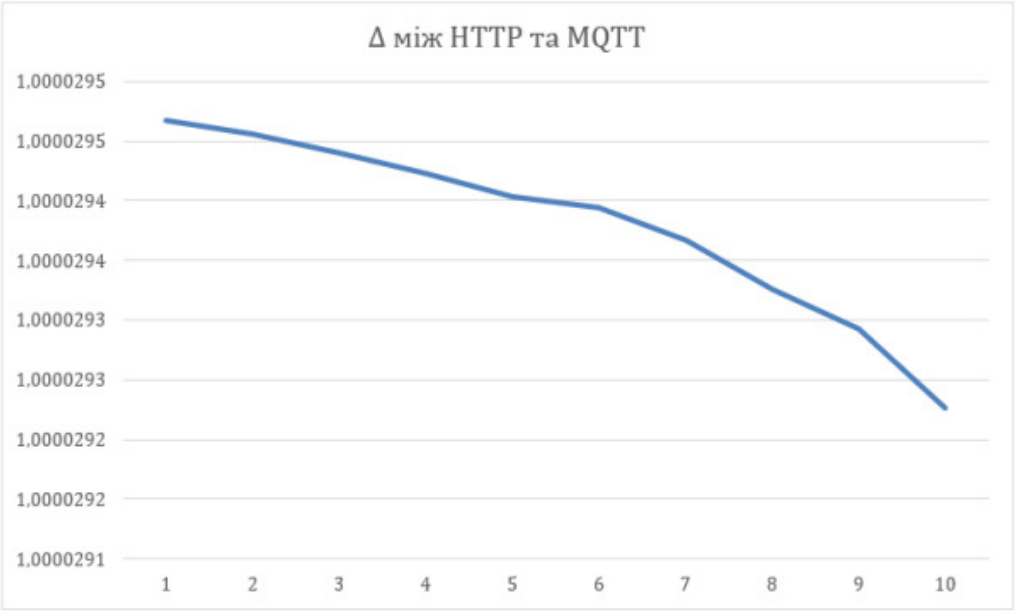


Рисунок 2.9 – Відношення середнього часу відгуку

2.4 Пропускна здатність при використанні кешуючого проксі–сервера

Розрахунок, яке значення пропускної здатності можна економити, використовуючи кешуючий проксі–сервер розраховуємо за (формулою 2.16).

$$\text{Економія ПЗ} = \text{ПЗ} \cdot \text{результативність} \cdot (\%\ \text{кешування}), \qquad (2.16)$$

де ПЗ – пропускна здатність.

Протокол HTTP передбачає кешування даних, що дозволяє зекономити частину пропускної здатності мережі. В цілях аналізу показника кешування, розглянемо випадок, коли результативність складає 60% і 42% всього трафіку можна зекономити, використовуючи кешування. Пропускна здатність HTTP складає 4,4471 Мбіт/с, а MQTT – 4,434 Мбіт/с.

$$\text{Економія ПЗ} = 4{,}4471 \cdot 0{,}42 \cdot 0{,}60 = 1{,}12 \left(\frac{\text{Мбіт}}{c}\right), \qquad (2.17)$$

$$\text{ПЗ} = 4{,}4471 - 1{,}12 = 3{,}327 \left(\frac{\text{Мбіт}}{c}\right). \qquad (2.18)$$

Так, як протокол MQTT не використовує кешування, то використання HTTP буде давати виграш в розмірі (формула 2.19):

$$4{,}434 - 3{,}327 = 1{,}107 \left(\frac{\text{Мбіт}}{c}\right). \qquad (2.19)$$

Таблиця 2.10 – Порівняння пропускної здатності протоколів при використані кеш–серверу

Протоколи	Пропускна здатність при кешуванні, Мбіт/с
HTTP	3,327
MQTT	4,434

В (таблиці 2.10) порівняно пропускну здатність протоколів з використанням кешування, розраховані за (формулою 2.19) дані розрахунки нам показують що протокол HTTP швидший чим протокол MQTT на $1{,}107\left(\frac{\text{Мбіт}}{\text{с}}\right)$ при кешуванні.

3 РОБОЧІ СХЕМИ ПРОЕКТУ З ОПИСОМ ВИКОРИСТОВУВАНОГО ОБЛАДНАННЯ

3.1 Типи підключень до мережі

В даній дипломній роботі запропонована корисна модель яка відноситься до області керування потоками протокольних типів даних в мережах. Дана модель реалізуються на серверах, які обміняться інформацією чи збирають дані в базу даних, це не тільки інформація соціальні мережі та системи масового обслуговування, а і інформація різних приладів побуту (розумний дім, статистичні прилади для відстеження змін клімату, малі різні датчики та великі верстати, GPS–датчики автомобілів, системи охорони).

Всім цим приладам чи датчикам, якщо вони використовуються організацією чи потребують онлайн керування або відстеження дій, потрібно збирати та обробляти запити с пристроїв та надавати відповідь, для цього потрібний зв'язок, для того щоб приймати сигнали чи відправляти їх, даний зв'язок можна реалізувати різними з’єднаннями наприклад різноманітні типи відображені на (рисунок 3.1):

Рисунок 3.1 – Різноманітний типи з’єднань

Залежно від того який тип з’єднань вибрати для передачі інформації з пристрою на сервер, залежить швидкість та коефіцієнт загублення пакетів які не дійшли до сервера. Вибираючи під’єднання оптоволоконне чи Ethernet, використовується транспортний протокол TCP котрий керує передачею даних, дані потрапляють на сервер без втрат та набагато швидше оскільки даний протокол має пряме підключення від пристрою до сервера. Але не завжди можна підключити пристрій на пряму через кабель, бо це не зручно і пристрій не зможе пересуватися тому використовують для цього технологію бездротового зв’язку (WiFi, Bluetooth, мобільний зв’язок) де діє інший протокол UDP, інформація не завжди потрапляє на сервер від пристрою за це відповідають багато чинників котрі впливають на загублення пакетів даних, якщо дані загубилися та мають не значний пріоритет то протокол UDP може знехтувати цими даними, а для тої інформації що має високий пріоритет і ця інформація не потрапила на сервер, то даний пристрій не отримує відповідь від сервера, що пакет інформації дійшов та заново відправляє цю інформацію поки не отримає відповідну відповідь.

3.2 Модель передачі даних в мережі

Вхідний сигнал являє собою послідовність (біт рівний тривалості), в якому за такт передається один біт інформації, реалізація прикладу як сигнал відсилається від відправника показана на (рисунке 3.2) і на (рисунке 3.3) показано як інформація приходить на приймач одержувачу у вигляді сигналу.

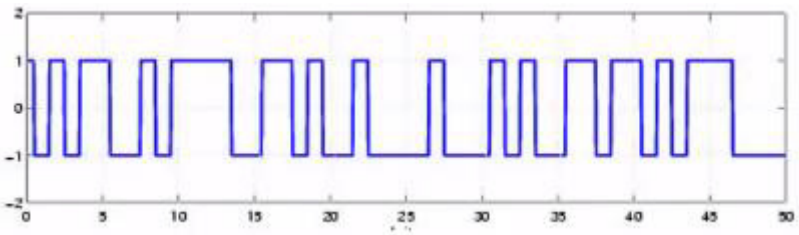


Рисунок 3.2 – Біт інформації у вигляді сигналів від відправника

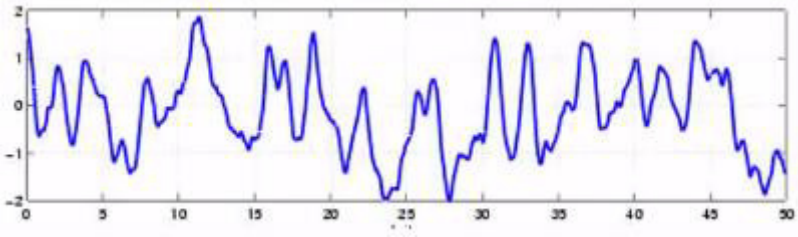


Рисунок 3.3 – Біт інформації яка прийшла на приймач одержувачу

За рахунок того, що в середовищі відбувається спотворення сигналів то одержувач приймає не такий сигнал який був відправлений, а змінений сигнал, де приймач одержувача розбирає і відновлює цей сигнал.

Розглянемо найпростішу модель передачі даних із застосуванням завадостійкого кодування, для уникнення помилок в отриманій інформації (рисунок 3.4), виділені помічені блоки що обробляють інформацію з застосуванням завадостійких кодів при якому вводиться надмірність при кодуванні, що в свою чергу зменшує пропускну здатність.

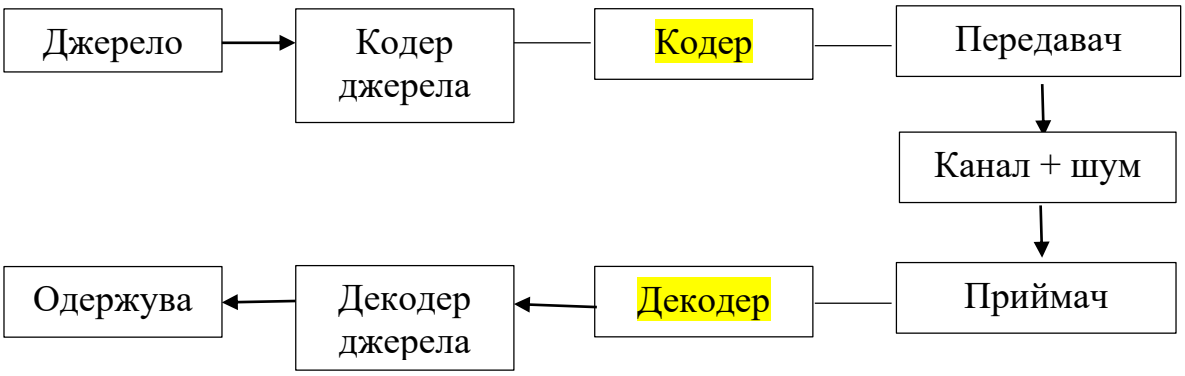


Рисунок 3.4 – Модель системи передачі даних

3.3 Радіоелектронні перешкоди в мережі

Радіоелектронні перешкоди – це деякі перешкоди, електромагнітні випромінювання, які погіршують якість функціонування радіоелектронних засобів (РЕЗ), перешкоди імітують або спотворюють спостерігаючі і

реєструючі кінцевої апаратурою сигнали, ускладнюють або виключають виділення корисної інформації, знижують їх дальність дії і точність роботи автоматичних систем управління. Під впливом перешкод РЕЗ та системи можуть перестати бути джерелами інформації незважаючи на їх повну справність і працездатність. Так як прибрати різноманітні РЕЗ перешкодами одного виду неможливо, то застосовують спеціальні види, що відповідають тим пристроям котра отримала пакет даних с перешкодою.

За ефектом (характеру) впливу на РЕЗ розрізняють маскуючі і імітуючі перешкоди.

Маскуючі перешкоди погіршують характеристики приймального пристрою РЕЗ, що збільшує кількість прийнятих символів, та знижують інформативність повідомлення, створюють фон, на якому утруднюється або повністю виключається виявлення, розпізнавання, виділення корисних сигналів або відміток цілей. При збільшенні потужності перешкод їх маскувальна дія зростає.

Імітуючі (дезінформуючі) перешкоди – це сигнали, що випромінюються станцією перешкод для внесення неправдивої інформації. За структурою вони близькі до корисних сигналів і тому створюють в крайовому пристрої РЕЗ сигнали помилкових змістів, подібні реальним, знижують пропускну здатність системи, призводять до помилкових дій. При впливі імітують перешкод характеристики приймального пристрою не погіршуються. Окремим випадком імітує перешкоди є ретрансляції перешкода, повністю повторює корисний сигнал.

Ефект впливу перешкод погіршує якість оброблюваної інформації в результаті її руйнування або старіння, що збільшує ступінь невизначеності при прийнятті рішень. Залежно від способу наведення перешкод, співвідношення ширини спектрів перешкод і корисних сигналів (рисунок 3.2 а) маскуючі перешкоди поділяють на загороджувальні (рисунок 3.2 в) і прицільні (рисунок 3.2 б; 1 – перешкода збігається за частотою з сигналом; 2 – перешкода не збігається за частотою з сигналом РЕЗ).

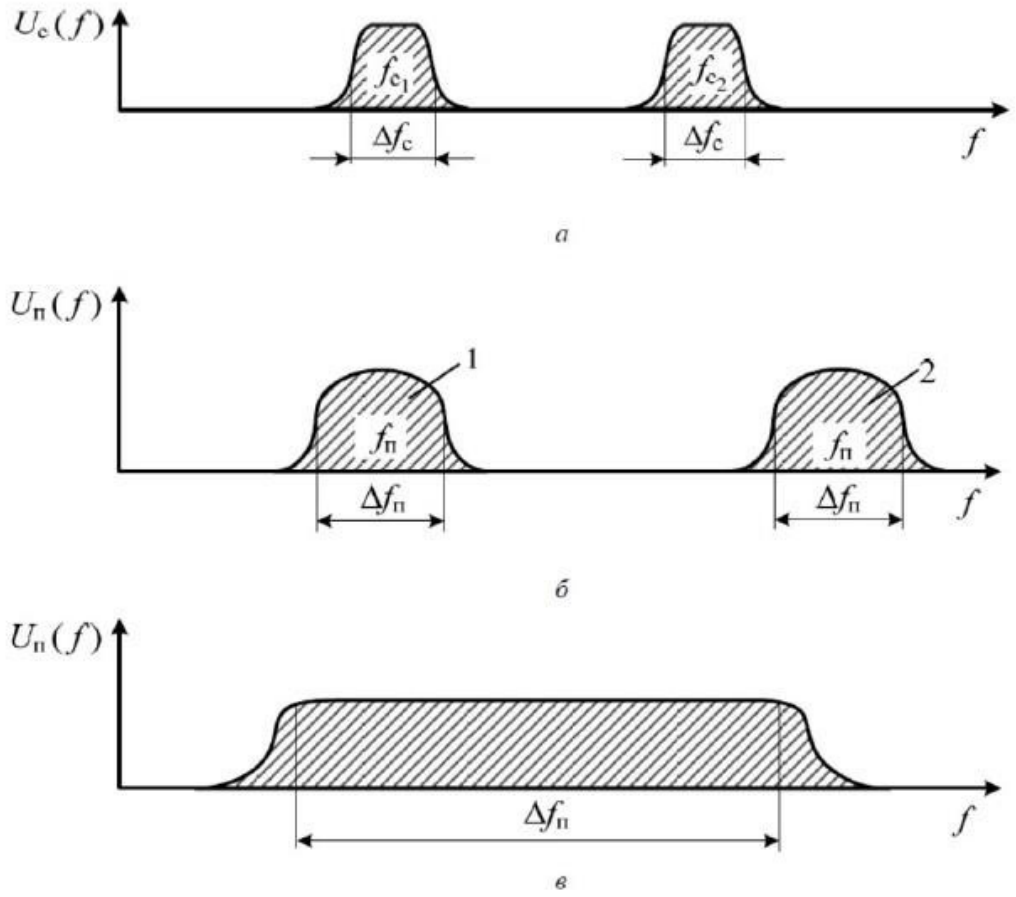


Рисунок 3.2 – Види перешкод в залежності від ширини спектра сигнал

Загороджувальні перешкоди мають ширину спектра частот, що значно перевищує смугу, займану корисним сигналом, що дозволяє пригнічувати одночасно кілька РЕЗ без точного наведення передавача перешкод (ПП) за частотою. Їх можна створювати, не маючи повних даних про параметри сигналів подавляемих РЕЗ.

Особливістю загороджувальних перешкод є те, що при незмінній потужності ПП їх спектральна щільність потужності $G_п$(Вт/МГц) зменшується в міру розширення спектра випромінювання. При рівномірному спектрі вона являє собою відношення енергетичного потенціалу передавача перешкод $P_{пп}G_{пп}$ до ширини спектра частот перешкоди $\Delta f_п$.

Для суцільної загороджувальної перешкоди, використовуємо (формулу 3.1):

$$G_п = \frac{P_{пп} G_{пп}}{\Delta f_п}. \quad (3.1)$$

Наприклад, якщо ПП, який має еквівалентну потужність 5 000 Вт, створює загороджувальні перешкоди в діапазоні частот від f_1 = 9 500 МГц до f_2 = 10 000 МГц ($\Delta f_п$ = 500 МГц), то $G_п$ = 5 000/500 = 10 Вт/МГц.

Прицільні перешкоди мають ширину спектра, порівнянну (що дорівнює або в 1,5–2 рази перевищує) з шириною спектра сигналу, що придушується РЕЗ. Ефективність їх впливу залежить від точності суміщення по частоті з корисним сигналом. Прицільні перешкоди характеризуються високою спектральною щільністю потужності. Оскільки вони випромінюються в вузькій смузі частот, то можуть бути реалізовані малопотужними ПП.

Призначення будь–якого каналу зв'язку – це передача тієї чи іншої інформації. В даному випадку розглядаються широкосмугові канали зв'язку, призначені для передачі сигналів (коду, файлів, аудіо, відео). З теорії зв'язку відомо, що існують дві основні причини зниження вірогідності передачі. Перша – зниження відношення сигнал/шум (SNR – Signal Noise Ratio або S/N – Signal to Noise). Друга причина – спотворення сигналу. Сигналом може бути інформаційний сигнал, модульований несуча або відеоімпульс. Стосовно до аналогових сигналів використовуються поняття інтермодуляционних спотворень (наприклад, добре всім відомі СТВ, CSO і канальні спотворення). У цифрових же системах зв'язку здебільшого користуються поняттям між символьної інтерференції. У даній роботі розглядається розрахунок ймовірності помилки (BER – Bit Error Rate) в залежності від реалізованого значення S/N.

З теорії передачі аналогових сигналів відомо, що одним з критеріїв якості сигналу є S/N, яке визначається, як відношення середньої потужності сигналу (S) до середньої потужності шуму (N). У цифрових системах зв'язку частіше використовується нормована версія S/N, що позначається як E_b/N_0, де E_b –

енергія біта. Її можна описати, як потужність сигналу S, помножену на час передачі біта інформації T_b. N_0 – це спектральна щільність потужності шуму, і її можна виразити як потужність шуму N, поділену на ширину смуги W. Оскільки час передачі біта і швидкість передачі бітів взаємно протилежні , T_b можна замінити на 1/R: (де R – бітова швидкість).

$$\frac{E_b}{N_0} = \frac{ST_b}{N/W} = \frac{\frac{S}{R}}{\frac{N}{W}}. \tag{3.2}$$

Перепишемо вираз (3.2) так, щоб було явно видно, що відношення E_b/N_0 є відношенням S/N, нормоване на ширину смуги і швидкість передачі бітів:

$$\frac{E_b}{N_0} = \frac{S}{N}\left(\frac{W}{R}\right). \tag{3.3}$$

Однією з найважливіших метрик якості в системах цифрового зв'язку є графік залежності ймовірності появи помилкового біта PB (BEP – Bit Error Probability) від E_b/N_0. На (рисунке 3.3) показаний вид більшості подібних кривих залежностей Pb від E_b/N_0.

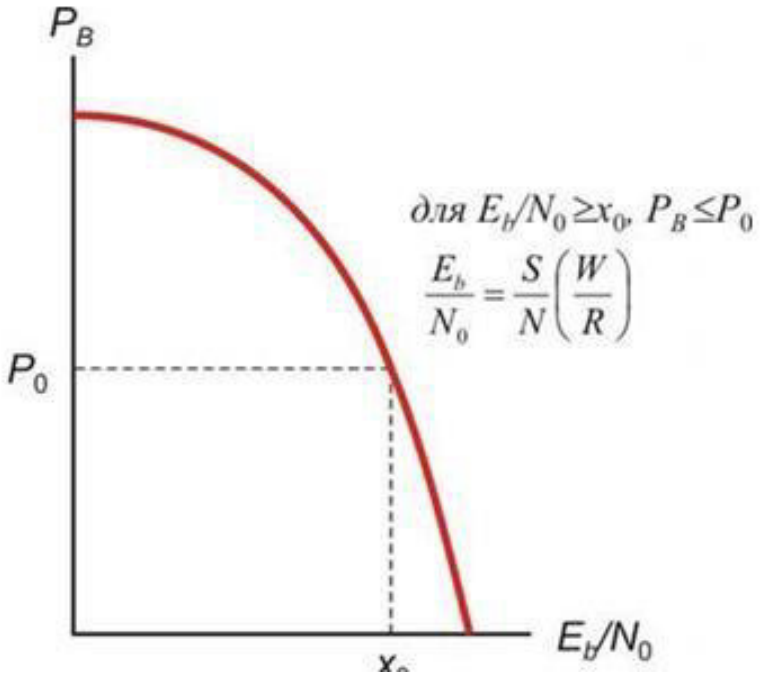


Рисунок 3.3 – Залежність Pb от E_b/N_0

При $E_b/N_0 \geq x_0$, $P_b \leq P_0$. безрозмірне відношення E_b/N_0 – це стандартна якісна міра продуктивності систем цифрового зв'язку. Отже, необхідне відношення E_b/N_0 можна розглядати як метрику, що дозволяє порівнювати якість різних систем: чим менше необхідне відношення E_b/N_0, тим ефективніші процес детектування при даній ймовірності помилки.

Ставлення S/N – зручний і звичний критерій якості: числитель (S) являє міру потужності сигналу (в узгодженому режимі вимірюється вольтметром або аналізатором спектру), яку бажано зберегти, а знаменник (N) – погіршення шумової потужності, під якою найчастіше мають на увазі теплову шумову потужність і потужність перешкод» (легко вимірюється ваттметром або аналізатором спектру в обмовляється смузі частот). Ставлення S/N інтуїтивно сприймається як міра якості. Розглянемо, навіщо для цифрових систем необхідна інша метрика – відношення енергії біта до спектральної щільності потужності шуму.

Стосовно до аналогових сигналів зручно користуватися поняттям потужності, тому що сигнал даного виду безперервний в часі. Аналоговий

сигнал представляється нескінченним по тривалості, який не потрібно розмежування в часі. Необмежено тривалий аналоговий сигнал містить нескінченну енергію, отже, використання енергії – не найзручніший спосіб опису характеристик такого сигналу. Як зазначалося вище, потужність легко вимірюється (при фіксованому опорі лінії передачі досить виміряти напругу сигналу) і зручна для практичного використання.

У цифрових же системах зв'язку ми передаємо (і приймаємо) символи шляхом передачі деякого сигналу протягом кінцевого проміжку часу передачі символу – Ts. Стосовно до одного інформаційного символу потужність (усереднена за часом) залежить від швидкості передачі. Для сигналів з дискретною структурою потрібна «досить добра» метрика в межах кінцевого проміжку часу. Набагато більш зручним параметром опису цифрових сигналів є енергія, тобто потужність, про інтегрована за часом.

Таким чином, саме нормований параметр E_b/N_0 є найзручнішою метрикою для цифрових систем. Цифровий символ – це транспортний засіб, що передає цифрове повідомлення. Повідомлення може містити 1 біт (двійкове повідомлення), два (четверичной), 10 біт (1024–ковий). В аналогових системах немає нічого подібного такої дискретної структури повідомлення. Аналоговий інформаційне джерело – це нескінченно квантова хвиля. Для цифрових систем необхідно, щоб критерій якості дозволяв порівнювати одну систему з іншого саме на бітовому рівні. Отже, описувати цифрові сигнали в термінах S/N практично марно, тому що символ може переносити різну кількість біт. Для конкретики міркувань візьмемо, наприклад, що для встановленої ймовірності виникнення помилки (BER) в цифровому двійковому сигналі необхідне відношення S/N дорівнює 20. Оскільки двійковий сигнал має однобітове значення, необхідне відношення S/N на біт має 20 одиницям. Тепер припустимо, що наш сигнал вже є 1024–річним, з тим же самим відповідним S/N = 20. Тепер, оскільки сигнал має 10–бітове значення, необхідне відношення S/N на один біт одно всього 2. Даний приклад міркувань показує, що для

цифрових систем зв'язку необхідно використовувати саме параметр E_b/N_0, а не S/N.

Поняття «Шум». Серед усіх джерел шуму найбільш поширеним на практиці і найбільш широко використовуваним в якості моделі випадкового (хаотичного) процесу є шум, що описується нормальним (гауссовским) розподілом. Він виникає в результаті одночасного впливу багатьох незалежних випадкових джерел.

Типовим прикладом шуму з нормальною щільністю, тобто рівномірним, є тепловий шум, обумовлений броунівським рухом електронів в провіднику. Шум подібного типу прийнято називати білим. Для фахівців з цифровій техніці ідеальний білий шум простіше уявити у вигляді послідовності нескінченно коротких імпульсів з випадковою амплітудою і наступних через випадкові проміжки часу. Така послідовність імпульсів буде володіти необмеженим однорідним спектром. (Спектр нескінченно короткого імпульсу нескінченний).

Іншими словами, реальний білий шум відповідає ідеальному білого шуму, який пройшов через фільтр. Він уже має обмежений спектр (еквівалент імпульсів з кінцевою тривалістю), а при обмеженій ширині спектра його потужність в кінцевій смузі частот також конечна. Зазвичай при розрахунках потужності N реального білого шуму в смузі частот W (Гц) використовують спектральну щільність потужності шуму $N_0 = N/W$ (Вт/Гц) і абсолютну температуру джерела шуму T (K^0), де $K^0 = C^0 + 2730$. При цьому найбільша потужність шуму, яку можна отримати від теплового джерела (тобто в узгодженому режимі роботи) дорівнює:

$$N = kTW, \tag{3.4}$$

де $k=1{,}138 \cdot 10^{-23}$ (Дж/К) – постійна Больцмана.

На практиці багато зручніше працювати з децибельними рівнями (тільки складаються і віднімаються):

$$N = -228{,}6 + 10\lg(T) + 10\lg(\mathrm{W})\ (\partial\mathrm{Б} \cdot \mathrm{Вт}), \tag{3.5}$$

$$\mathrm{N} = -228{,}6 + 10\lg(\mathrm{T})\ (\partial\mathrm{Б} \cdot \frac{\mathrm{Вт}}{\mathrm{Гц}}). \tag{3.6}$$

Взаємозв'язок потужностей і енергії сигналу для цифрових систем взаємозв'язок потужностей і енерії сигналу грунтується за вираженням (3.7):

$$\frac{\mathrm{E}}{\mathrm{N_0}} = \frac{\mathrm{SWT_0}}{\mathrm{N}} = \frac{\mathrm{WT_0S}}{\mathrm{N}}, \tag{3.7}$$

де $\mathrm{E} = \mathrm{ST_0}$ – за визначенням енергія сигналу,

$\mathrm{N} = \mathrm{N_0W}$ – потужність шуму,

$\mathrm{T_0}$ – час передачі сигналу.

Величина $\mathrm{WT_0}$ (тобто, час передачі одного символу, помножене на ширину смуги) іноді іменується базою сигналу і в даному випадку є коефіцієнтом перерахунку відносини енергій сигналу і шуму в відношення їх середніх потужностей.

При передачі цифрового сигналу з форматом модуляції M–QAM (M – формат модуляції або число елементів простору сигналів при цифрової модуляції) число рівнів амплітуд L визначається як

$$L = \sqrt{\mathrm{M}}. \tag{3.8}$$

Енергія символу сигналу визначиться за (формулою 3.9):

$$\mathrm{E_s} = \mathrm{E_b}\log_2\mathrm{L}. \tag{3.9}$$

Очевидно, що при передачі двійкових сигналів $E_s = E_b$, а при передачі багаторівневих імпульсів в основній смузі, що збігається зі смугою Найквиста

W_N = 0,5T_b, потужність символу S = (E_b/T_b) $\log_2 L$ і потужність шуму одно N=N_0(0,5Tb).

$$\frac{S}{N} = 2(\log_2 L)\frac{E_b}{N_0}. \quad (3.10)$$

Переведемо (формулу 3.10) в логарифмічну форму:

$$\frac{S}{N} = \frac{E_b}{N_0} + 10\lg(m). \quad (3.11)$$

Серед показників, що характеризують відношення потужностей, широко використовується також ставлення несуча/шум (C/N), яке показує, у скільки разів потужність C прийнятої модульованої високочастотної (ВЧ) несучої на виході приймального фільтра з смугою більше потужності шуму N, що породжується спільною дією всіх джерел шуму даного тракту. Ставлення C/N є зручним параметром при розрахунках енергетики на вході приймача. Наведемо корисну залежність:

$$\frac{E_b}{N_0} = \frac{C}{N} + 10\frac{W}{f_s \cdot m}, \quad (3.12)$$

де f_s – символьна швидкість.

Імовірність помилки при прийомі цифрових сигналів є дуже важливим параметром, за яким ведуть оцінку можливості його передачі з того чи іншого каналу зв'язку. Відразу обмовимося, що ймовірність помилки (Bit Error Probability – BEP) і швидкість виникнення бітової помилки (Bit Error Rate – BER) – це дещо різні поняття. Проте, їх чисельні значення дуже близькі і, ведучи мову про BEP (PB), завжди мають на увазі BER, тому що це фізична величина, що реєструється вимірювальними приладами. Точно також ми

будемо поступати і в даному випадку. Імовірність помилки в загальному випадку дорівнює сумі ймовірностей всіх можливостей її появи. Ми ж, як і раніше, будемо розглядати вплив тільки основного джерела появи помилки – адитивного білого гауссовского шуму (Additive White GaussianNoise – AWGN).

Вирази, досить повно описують ймовірність помилки P_b, дуже громіздкі. Проте, з дуже малою похибкою (близько 0,1 дБ) вони можуть бути спрощені. Наприклад, найбільш короткою і зручною формулою є функція:

$$P_b\left(\frac{E_b}{N_0}\right) \approx 2 \cdot \left(1 - \frac{1}{\sqrt{M}}\right) \cdot \operatorname{erfc}\left[\sqrt{\frac{3\log_2(M)}{2(M-1)} \cdot \frac{E_b}{N_0}}\right]. \qquad (3.13)$$

Для прямокутного безлічі сигналів гауссова каналу і прийому за допомогою узгоджених фільтрів, ймовірність появи бітової помилки при модуляції M –QAM, де M = 2^k і k – парне число, за (формулою 3.13) може бути записано в розрахунковому вигляді:

$$P_b \approx \frac{2 \cdot (1 - L^{-1})}{\log_2 L} \cdot Q\left[\sqrt{\frac{3\log_2 L}{(L^2 - 1)} \cdot \frac{2E_b}{N_0}}\right]. \qquad (3.14)$$

Тут, $L = \sqrt{M}$ як і раніше – кількість рівневих відліків, а Q(x) являє собою гаусів інтеграл помилок і часто використовується при описі ймовірності з гаусом щільністю розподілу. Визначається ця функція наступним чином:

$$Q(x) \approx \frac{1}{\sqrt{2\pi}} \int_x^{\infty} \exp\left(-\frac{u^2}{2}\right) du. \qquad (3.15)$$

Відзначимо, що гаусів інтеграл помилок може визначатися кількома способами. При цьому всі визначення однаково придатні для опису ймовірності помилки при Гауссові шумі. Q(x) безпосередньо не обчислюється в

аналітичному вигляді і зазвичай наводиться у вигляді довідкових таблиць. Ця обставина певною мірою гальмує розвиток машинних методів розрахунку цифрових каналів зв'язку. Проте, при певних обмеженнях, функція Q(x) апроксимується більш простими виразами. Найбільш вдалою апроксимацію для x>3 є досить проста функція, придатна для подальших розрахунків:

$$\mathrm{Q(x)} \approx \frac{1}{x\sqrt{2\pi}} \exp\left(-\frac{\mathrm{u}^2}{2}\right). \qquad (3.16)$$

3.4 Розрахунок залежності ймовірності бітової помилки від SNR

Потрібно розрахувати ймовірність помилки BER для 64QAM за (формулою 3.10) знайдемо різницю між $\mathrm{S/N}$ і $\mathrm{Eb/N0}$ в логарифмічній формі:

$$\frac{S}{N} = \frac{E_b}{N_0} + 10\lg(6) = \frac{E_b}{N_0} + 7{,}78,$$

$$\frac{\mathrm{S}}{\mathrm{N}} \text{ і } \frac{\mathrm{E_b}}{\mathrm{N_0}} = 7{,}8\ дБ.$$

Використовуючи формулу (3.8) отримаємо параметр L:

$$\mathrm{L} = \sqrt{64} = 8.$$

Підставивши L отримаємо ймовірність появи бітової помилки при модуляції 64–QAM в розрахунковому вигляді:

$$\mathrm{P_b} \approx \frac{2\cdot(1-8^{-1})}{\log_2 8}\cdot \mathrm{Q}\left[\sqrt{\frac{3\log_2 8}{(8^2-1)}\cdot\frac{2\mathrm{E_b}}{\mathrm{N_0}}}\right] = \frac{7}{\mathrm{S/N}}\mathrm{Q}\left[\sqrt{\frac{2}{7}\cdot\frac{\mathrm{E_b}}{\mathrm{N_0}}}\right]. \qquad (3.17)$$

Функцію P_b розрахуємо за (формулою 3.16), підставляючи разні значення S/N відповідно (до таблиці 3.1). Отримані дані заносимо в (таблицю 3.1)

Таблица 3.1 – Залежність ймовірності бітової помилки від SNR

S/N, дБ	P_b
12	0,18
14	0,11
16	0,066
18	0,03
20	0,0099
22	0,00175
24	0,000175
26	0,00000274

Розрахувавши ймовірність для деяких значень SNR побудуємо залежність (рисунок 3.4).

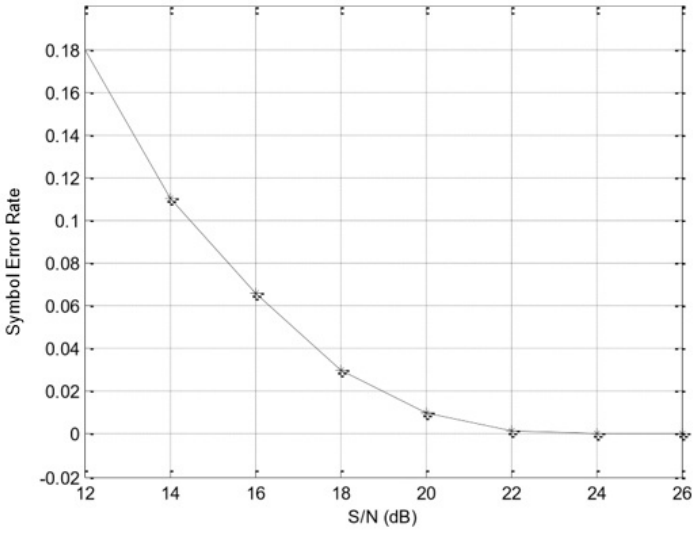


Рисунок 3.4 – Залежність P_b от SNR

3.5 Різні формати пристроїв для серверів і їх тести на продуктивність

Прикладні багато користувачеві комерційні та бізнес–системи, що включають системи управління базами даних і обробки транзакцій, великі видавничі системи, мережеві додатки і системи обслуговування комунікацій, розробку програмного забезпечення та обробку зображень все більш наполегливо вимагають переходу до моделі обчислень «клієнт–сервер» і розподіленої обробки. У розподіленої моделі «клієнт–сервер» частину роботи виконує сервер, а частина призначений для користувача комп'ютер (в загальному випадку клієнтська і призначена для користувача частини можуть працювати і на одному комп'ютері). Існує кілька типів серверів, орієнтованих на різні застосування: файл–сервер, сервер бази даних, принт–сервер, обчислювальний сервер, сервер додатків. Таким чином, тип сервера визначається видом ресурсу, яким він володіє (файлова система, база даних, принтери, процесори або прикладні пакети програм).

Є певна класифікація серверів, яка визначається масштабом мережі, в котрій вони використовуються: сервер відділу або сервер масштабу підприємства чи сервер робочої групи. Ця класифікація є досить умовна. Наприклад, розмір групи може змінюватися в діапазоні від декількох чоловік до декількох сотень чоловік. Очевидно в залежності від числа користувачів і характеру розв'язуваних ними завдань вимоги до складу обладнання та програмного забезпечення сервера, до його надійності і продуктивності сильно варіюються. Параметри різних серверів наведені в (таблиці 3.2).

Таблиця 3.2 – Порівняння обладнання для серверів

Тип	Одноплатний мікрокомп'ютер	Тонкий клієнт	Desktop Server
Модель	Raspberry Pi 4 Model B	HP t630	ProLiant ML30 Gen10
Тип процессора	Broadcom BCM2711 ARM Cortex–A72 Quad Core	AMD GX–420GI	Intel Xeon E–2124 (з можливістю заміни)
Кіл–ть ядер	4	4	4
Частота, ГГц	1,5	2	3,3 – 4,3
Об'єм оперативної пам'яті, ГБ	4	4	4 (з можливістю розширення)
ОЗП	LPDDR4	DDR4	DDR4
Графічний чіпсет	VideoCore IV	AMD Radeon R7E	Matrox G200
Обсяг SSD, ГБ	64	64	64
Розміри, см	8×6×2	17x19x4	36,8 x 17,5 x 47,5
Споживання, Вт	15	65	350
Ціна, грн	2500	11000	19000

В (табл. 3.2) представлені три пристрої, які виконують роль web–сервера, за параметрами ці пристрої вибиралися приблизно однакові що б більш точніше порівняти наскільки ефективніше схожі параметри комплектуючих на різних розмірах пристроїв (маленький 8х6см; середній 17х4см; великий 36х17см).

3.5.1 Кластер та встановлення пакету Docker.

В даній дипломній роботі використовується міні комп’ютер Raspberry Pi 4 Model B на 4гб ОЗУ, але один цей міні комп’ютер котрий буде використовуватися як сервер годиться тільки до маленьких проектів, тому пропонується зв'язати декілька цих пристроїв в загальну мережу щоб вони могли розподілити між собою навантаження та функціонувати між маленькими–середніми–великими розрахунками, ця об’єднання міні ПК називається кластером.

Кластер має гнучкі параметрами котрі можна регулювати не затрачаючи багато фінансів, є надійним, багатофункціональним, доступністю швидкого терміналу і перезавантаження, хорошою обчислювальною потужністю, низьким споживанням електроенергії і вартістю пристрою.

Для реалізації поставленої задачі буде використовуватися 3 міні ПК, на котрі потім буде встановлене програмне забезпечення Docker. Для підключення використаємо наступне обладнання:

– комутатор TP–LINK TL–SF1008D;

– 3 міні ПК – Raspberry Pi 4 Model B на 4гб ОЗУ;

– блок живлення 5 Вольт 2 Ампера на 4 лінії.

Зібраний кластер в прозорому корпусі відображено на (рисунке 3.5).

Рисунок 3.5 – Кластер на Raspberry Pi 4 Model B

Комутатор відповідає за підключення до мережі інтернет, та за комутацію трьох міні ПК, ну а міні ПК власне і будуть вузлами нашого кластера.

Для тестування кластеру спочатку встановлюємо програмне устаткування Docker – це програмне забезпечення для автоматизації розгортання і управління додатками в середовищах з підтримкою контейнеризації. Дозволяє «упакувати» додаток з усім його оточенням і залежностями в контейнер, який може бути перенесений на будь–яку Linux–систему з підтримкою cgroups в ядрі, а також надає середовище з управління контейнерами. Docker встановимо на Raspberry Pi останню версію Raspbian ядром 4.19 [12] з оновленням до 8.3.

Для встановлення Docker скористаємося інструкціє з офіційного сайту Raspberry [14]. Для встановлення Docker Compose скористаємося інструкцією [20].

В результаті отримаємо Docker версії 17.05.0–ce та Docker Compose версії 1.13.0.

Встановити Docker та Docker Compose вдалося на всі три тестові міні ПК Raspberry Pi 4 B.

Тут слід зауважити, що чим потужніший міні ПК тим швидше на нього встановлюються пакети та розгортаються образи. Що стосується образів, то для міні ПК вони мають бути спеціальні, тобто підготовлені для роботи з ARM процесорами. У всьому іншому робота з docker ідентична роботі на звичайній Linux системі.

Підтримка Docker на міні ПК дозволяє спростити розгортання образів зі сервісами а також процес їх оновлення, потужності багатоядерних версій цілком достатньо для роботи сервісів, необхідних для побудови локальної системи для IoT пристроїв.

Docker swarm це спосіб управляти безліччю вузлів докера об'єднаних в одну логічну сутність. Тобто, побудувавши свій кластер ми можемо виконувати розпаралелення обчислень на різних вузлах. Перевіримо можливість побудови такого кластеру на міні ПК.

Далі нам необхідно налаштувати власне Docker swarm. Для цього скористаємося інструкцією [15]. Головним вузлом зробимо Raspberry Pi 4B,

резервним – Raspberry Pi 4B другий, Raspberry Pi 4B третій – звичайними вузлами. На (рисунке 3.6) зображено результат команди відображення вузлів, як можна побачити все налаштовано так, як і планувалося.

```
admin: @sudo doker node is
ID                          HOSTNAME   STATUS   AVALLABILITY   MANAGER STATUS
4de4jfv3gg5web66165n *      pi1        Ready    Active         Reachable
lsjhio955mnllo00f4dq        pi2        Ready    Active         Leader
gfd34ll6mv4kjklojfco        pi3        Ready    Active
```

Рисунок 3.6 – Вузли Docker swarm

Також у відповідності до інструкції для більш наочної візуалізації було встановлено сервіс для візуалізації роботи, доступ до якого відбувається через веб інтерфейс, зовнішній вигляд якого наведено на (рисунке 3.7):

Рисунок 3.7 – Веб інтерфейс сервісу візуалізації

Для тестування роботи запустимо у нашому кластері контейнер, який відображає веб сторінку та виконаємо його масштабування до 7 екземплярів. Після цього ми маємо 7 контейнерів, доступ до яких відбувається за кожним з

3–ма IP адресів міні ПК. Далі за допомогою балансерів можна реалізувати розподілення навантаження на всі вузли. Візуалізацію масштабування 7 веб контейнерів наведено на (рисунке 3.8):

Рисунок 3.8 – Візуалізація масштабування веб контейнерів

Отже, якщо підсумувати, то використання міні ПК дає нам можливість розгорнути Docker swarm, що надає нам чудову можливість виконання паралельних обчислень, адже він дає змогу легко масштабувати контейнери по всім вузлам. Також слід зазначити, що бажаним є використання вузлів з однаковими характеристиками з декількома ядрами.

3.5.2 Тест на продуктивність пристроїв.

Порівнюємо на ефективність три пристрої з (таблиці 3.2) і так–же кластер, який описаний в (пункті 3.4), тести проводилися в різних програмах віртуально.

Тест на швидкість роботи центрального процесора (CPU). Суть цього тесту полягає не тільки в оцінці синтетичної потужності CPU, але і в бойових умовах. Для цього вимірюємо продуктивність CPU в реальних умовах по навантаженню, використовуючи службову програму p7zip Benchmark: total rating результати представлені в (MIPS), (Більше – краще).

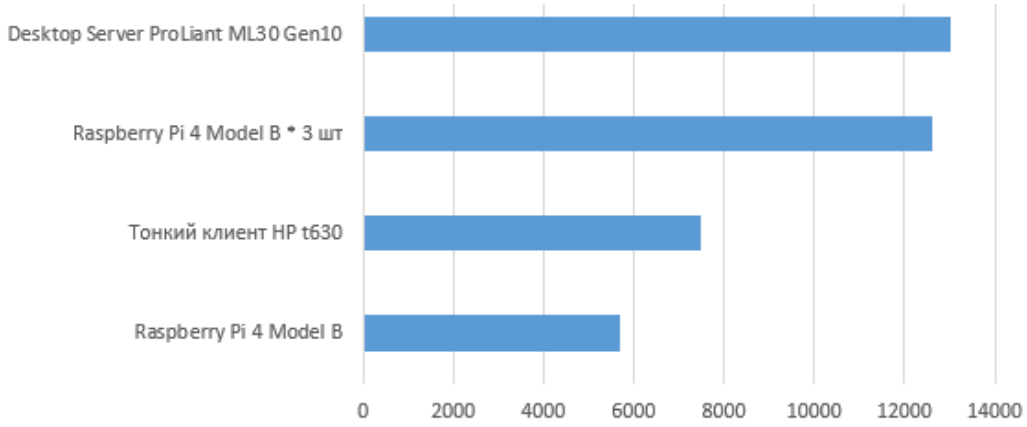


Рисунок 3.9 – Тест на швидкість роботи CPU, MIPS

Наступний тест називається кеш–бентч – це тест швидкості передачі швидкості між кешем процесора і оперативної пам'яті, результати (Мб/сек) (Больше – лучше).

Рисунок 3.10 – Тест кеш–бентч, Мб/сек

Далі вимірюємо швидкість роботи з оперативною пам'яттю виміряно тестом програми Phoronix Test Suite ramspeed–1.4.1 (Мб/сек) (Больше – лучше).

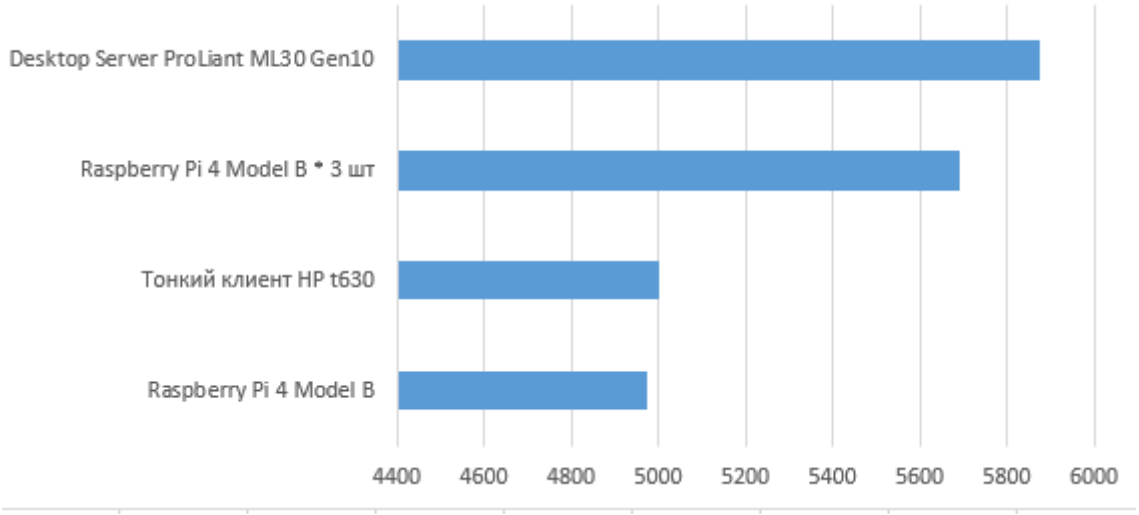


Рисунок 3.11 – Тест швидкості роботи ОЗП, Мб/сек

Тест на швидкість роботи з БД MySQL, тест називається OLTP програмного пакета Sysbench на 1млн записів (транзакцій в сек) (Більше – краще).

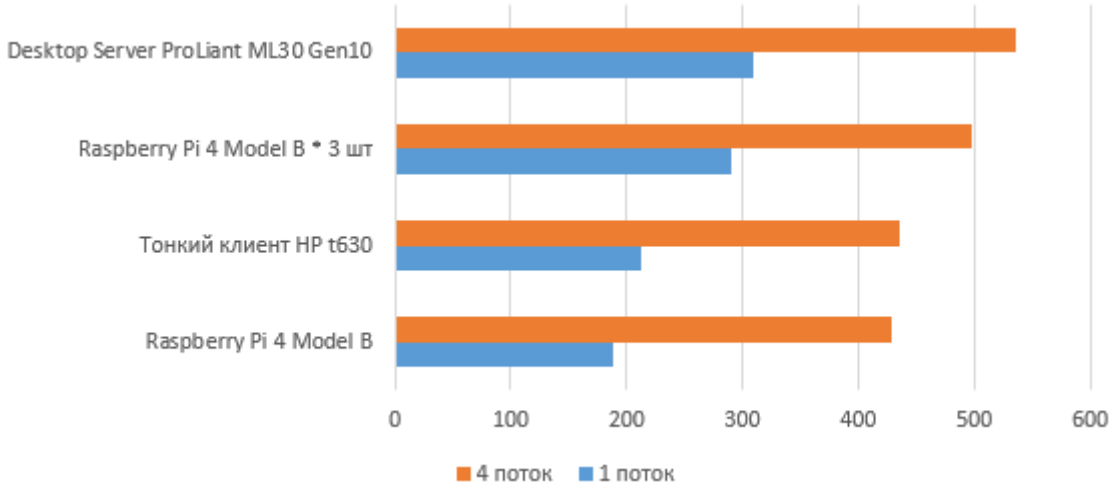


Рисунок 3.12 – Тест на швидкість роботи з БД, транзакцій в сек

Наступний тест проводився на швидкість роботи мережевих інтерфейсів, тест проводився за допомогою утиліти Iperf Client – пропускна здатність Ethernet (Мбіт/сек), (більше–краще)

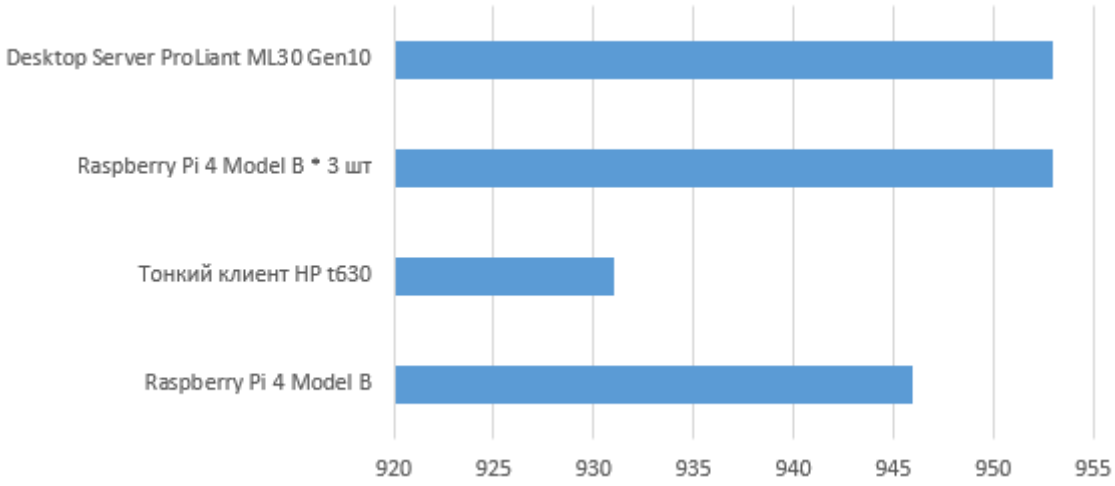


Рисунок 3.13 – Швидкість роботи мережевих інтерфейсів, Мбіт/сек

Наступний тест на споживану потужність результати в (Вт), (менше – краще).

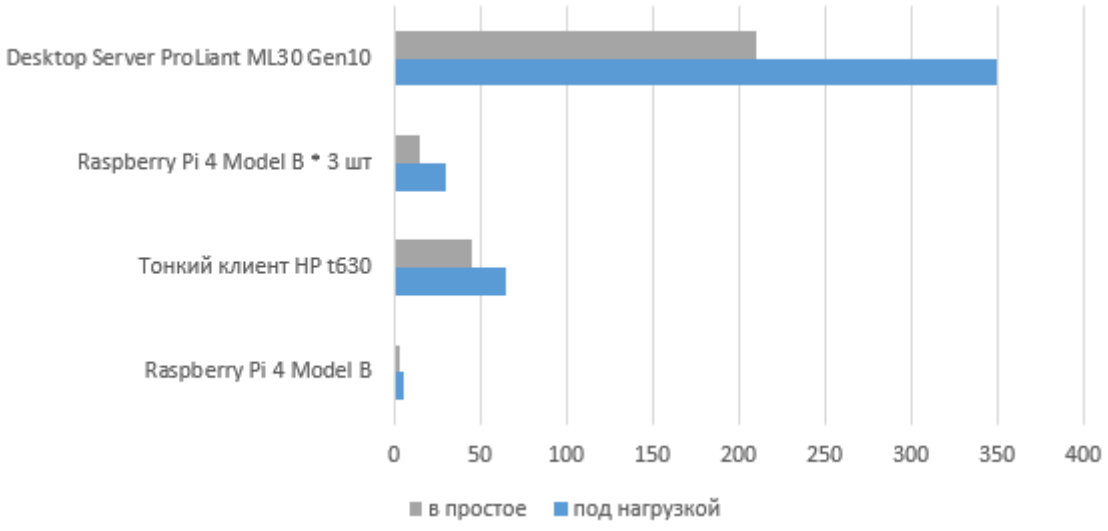


Рисунок 3.14 – Споживання електроенергії, Вт

По тестам, в котрих брали участь 4 пристрої, видно що лідирує desktop рішення із–за своїх параметрів і частоти потужності процесора, слідом наближене за йде кластер з 3–х Raspberry 4B, який є більш раціональним для використання для різних обсягів завдань, тому що його можна розширювати аж до тисяч пристроїв в залежності який стоїть комутатор, який об'єднує ці пристрої в єдину мережу, а за допомогою різних програм розподіляє рівномірно завдання на ці пристрої. Тонкий клієнт показує значення тестів трохи більше одного Raspberry але не значно, це пристрої не так раціонально в застосуванні, а desktop server споживає в десятки разів більше електроенергії і стоїть в три рази більше ніж кластер з 3–х штук Raspberry 4B який схожий по ха– ра з desktop server представленим в цій роботі [23].

Грунтуючись на виведення тестових показників раціональніше використовувати кластери – оскільки вони є багато функціональними, економічними і не дорогими.

Далі в наступному пункті буде представлена модель зроблена на основі даного кластера, в якій буде відображено спосіб управління пульсуючими потоками даних.

3.6 Спосіб управління пульсуючими потоками блоків даних

За пропонована корисна модель відноситься до області керування потоками протокольних типів даних, в мережах які обмінюються інформацією з пакетною комутацією та зокрема, до системи управління трафіком, що проходить через порти пристроїв комутації пакетів.

Поширене застосування корисна модель найбільш матиме в різноманітних варіантах засобів реалізації управління процесами обробки в умовах пульсуючого трафіку, тобто коли трафік перевищений на портах пакетних комутаторів, мережних шлюзів та маршрутизаторів.

При розрахунках навантаження на мережу застосовують формули теорії телетрафіку, які не у повній мірі враховують пульсуючий характер реального трафіку. На практиці значення інтенсивності потоків на деяких портах можуть значно перевищувати свою оптимальну пропускну спроможність в окремі проміжки часу за умовою пульсуючого характеру трафіку, що призводить до перенавантаження обладнання трафіком.

Для вирішення цієї ситуації розроблено багато методик локального управління потоками. Які можуть накладати обмеження на загальну кількість пакетів яка може зберігатися в буферній пам'яті, а далі розмір пам’яті що передається до місця призначення коли інтенсивність пульсуючого трафіку має максимум завантаження щонайменше одного із портів то виділяє цьому порту від загальної пропускної здатності телекомунікаційного пристрою за рахунок відповідного зменшення пропускної спроможності щонайменше одного порту, на якому в цей період спостерігається зменшення або незмінність інтенсивності потоку протокольних блоків даних. Але ця методика має свої суттєві недостати такі як, якщо виділена пам'ять буде заповнена на якомусь з портів комутатора, то подальше прийняття інформації блокується а всі зайві пакети будуть стерті, все це призводить до повторного передавання інформації що не є ефективним методом.

Один із методів запобігання перенавантаженню в пакетній мережі описаний в патенті України №42903 (МПК9 : H04L12/50, 2009.03.03) [20]. Згідно цього методу в основу корисної моделі поставлена задача підвищення ефективності способу управління пульсуючими потоками протокольних блоків даних шляхом оптимізації умов розподілу пропускної спроможності яка полягає в тому що коли є перенавантаження на один із портів то застосовується інші порти які можуть обробляти дану інформацію, та включає ситуацію коли всі порти перенавантажені чи застосовується специфічний порт який може приймати конкретну інформацію, то інформація на комутаторі який вимірює проміжну спосібність дає сигнал що не може прийняти інформацію дана інформація вертається клієнту відповідно у зв’язку (клієнт–сервер), та через

деякий проміжок часу інформація без втручання клієнта знову надходить коли сервер більш–менш не являється таким завантаженим.

Приклад здійснення способу відображено на (рисунку 3.15):

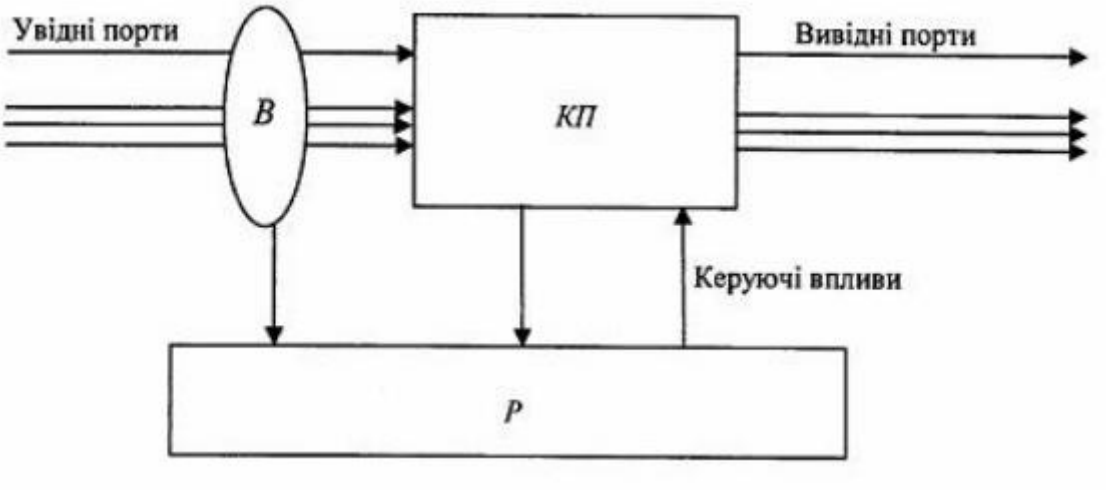


Рисунок 3.15 – Схема чотирьохпортового комутаційного пристрою

На рисунку 3.15 відображення схема чотирьохпортового комутаційного пристрою (КП), до складу компонентів КП додається вимірювач (В), який вимірює швидкість потоку пакетів на кожному із 4–х вхідних портів комутатора, діапазон котрих сягає 10^5пакетів/с, з довжиною пакета – 1500байт. Вимірювання здійснюється з періодичністю 100мс. Після чого результати вимірювання подається на інший елемент КП, на регулятор пропускної здатності (Р), який виконує роль визначення величини та напрямку необхідних змін ширини смуг пропускання портів комутатора.

Якщо значення другої похідної щодо якогось порту виявиться більше, ніж нуль, то регулятор Р подає на виконавчий механізм КП команду на збільшення ширини смуги цього порту. Якщо значення другої похідної щодо якогось порту виявиться менше, ніж нуль, то регулятор Р на виконавчий механізм КП подає команду на зменшення ширини смуги цього порту.

Недоліком даного методу є те що коли всі порти перенавантажені чи застосовується специфічний порт який може приймати конкретну інформацію, то інформація на комутаторі який вимірює проміжну спосібність дає сигнал що не може прийняти інформацію, після чого дана інформація вертається клієнту відповідно у зв'язку (клієнт–сервер), та через деякий проміжок часу інформація без втручання клієнта знову надходить коли сервер більш–менш не являється таким завантаженим – цей метод не включає те що інформація може не повернутися із–за пошкодження чи втручання дій клієнта який відправляв запит.

Таким чином, після огляду інших методів та патентів на дану тему, були зроблені висновки щодо модифікації патенту який розглядався до цього №42903 (МПК9 : H04L12/50, 2009.03.03), модифікація даного метода полягає, в тому що усуває попередні недоліки, а саме втручання клієнта, яке призводить до втрачання запиту на сервер, та економить час відгуку системи. Дані властивості були здобуті за допомогою:

– порівняння найбільш оптимального протоколу для різних розмірів масиву;

– порівняння різних типів серверів та їх тестування;

– зібрання кластеру та встановлення на нього програмного забезпечення;

– розробка нової системи управління пульсуючими потоками даних.

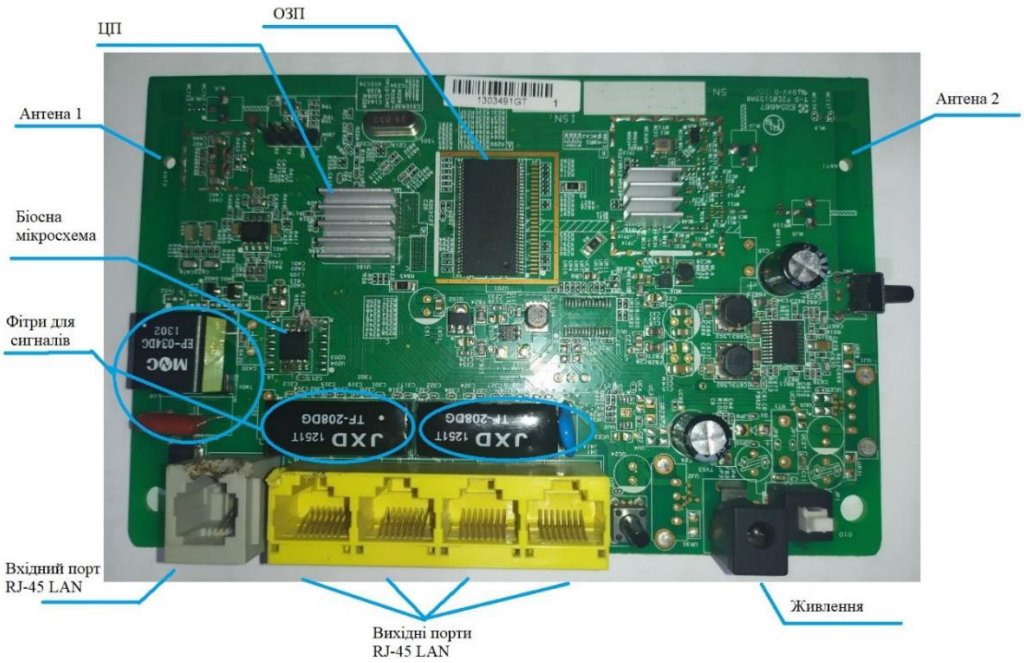


Рисунок 3.16 – Розміщення компонентів на платі роутера

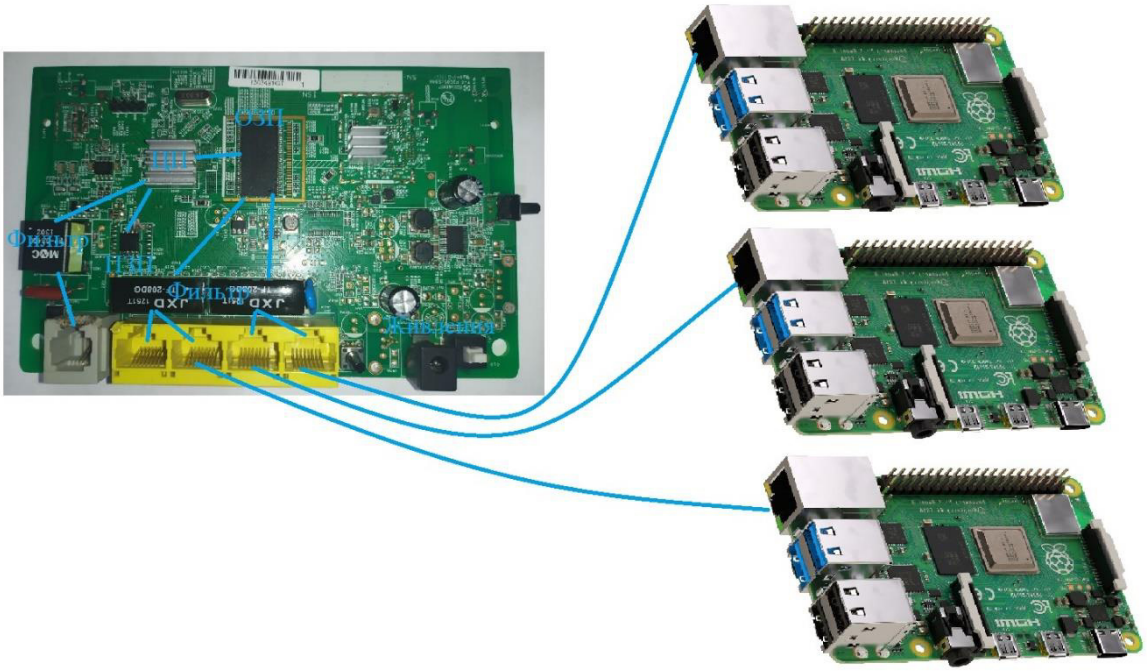


Рисунок 3.17 – Покращена схема багато портового комутаційного пристрою

На (рисунке 3.16) відображена схема роутеру, використовуючи даний роутер та з’єднавши його з кластером (рисунок 3.17) і встановши твердотільний накопичувач на який записується інформація, якщо регулятор пропускної здатності (ЦП) не має змогу перенаправити ширину пропускної смуги, бо всі порти перенавантажені чи застосовується специфічний порт який може приймати конкретну інформацію, то даний запит записується на твердотільний накопичувач, котрий взаємодіє с вимірювачем програмного забезпеченням, що вимірює загруженість ЦП на кластері, коли загруженість ЦП не така інтенсивна як до цього, то данні запити подаються через регулятор пропускної здатності на комунікаційний пристрій який обробляє запит [19].

ВИСНОВКИ

На сьогоднішній день IoT напрям розвитку інформаційних технологій є надзвичайно популярний. З кожним роком все більше і більше компаній виходять на ринок з різноманітними рішеннями у цій сфері. Впровадження IoT технологій у навколишнє середовище може суттєво покращити стан використання невідновлювальних енергоносіїв, спростити життя людини та підвищити комфорт її комфорт. Усе це створює чудові передумови до росту цієї галузі

В даній роботі було розроблено методологію розрахунку продуктивності мережі з метою використання їх для побудови IoT систем. Також було поставлено задачу дослідити можливості подібних пристроїв для організації паралельних обчислень і проаналізувати використання оптимальних протоколів для її взаємодії.

Для реалізації поставленої задачі було обрано міні комп'ютери Raspberry 4B котрі були об'єднані в одну мережу, що позволило додати функціональності даному пристрою, для розмірів потужності мереж. Порівняння обчислювальних можливостей виконувалося завдяки програмі AnyLogic та багатьма програмам тестування.

Перший тест полягав у складанні елементів масиву різного розміру, що дозволив оцінити швидкодію двох широко використовуємих протоколів. Аналіз протоколів передачі даних показав, що найбільш доцільними є протоколу MQTT та HTTP. Але так як, MQTT вимагає менших обчислень на боці клієнта, тобто датчика, то його використання є більш бажаним. Що до безпеки передачі даних, було досліджено, що при належній організацій систем захисту (використання складних паролів) Wi–Fi підключення можна вважати безпечним, а побудову географічно віддалених систем доцільно виконувати з застосуванням технологій VPN та SSH.

Друге порівняння та тестування серед обладнання серверних пристроїв відобразило що десктоп варіант пристрою краще за тестами але значно дорожче та споживає забагато електроенергії, тому було обраний варіант об’єднання міні ПК для створення кластеру який має достатню функціональність, потужність та досить не високу вартість.

В ході дослідження завадостійкості різних каналів даної моделі мережі було виявлено, що при однакових значеннях відносини сигнал/шум, в прямому каналі ймовірність бітової помилки мінімальна, Pb = 0,08. Якщо сигнал проходить через канали, з однаковим ставленням сигнал/шум (в нашому випадку 18дБ) ймовірність бітової помилки незначно вище, в порівнянні з прямим каналом, Pb = 0,14. У випадках, коли на шляху проходження сигналу сигнал/шум в каналах різний, ймовірність бітової помилки вище більш ніж в 2 рази Pb = 0,34.

Дослідження паралельних обчислень продемонструвало, що вони можливі на багатоядерних міні ПК. Також було протестована можливість побудови Docker кластерів на міні ПК, що дозволяє побудувати системи розподілених паралельних обчислень.

В якості демонстрації можливостей розглянутих систем була побудована система з хабом у вигляді кластеру, розгортанням MQTT брокера та Node–Red візуалізатора як Docker контейнерів.

Результати даних досліджень можуть бути використані для побудови власних IoT систем, або паралельних обчислень на міні ПК.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kuzianin O. MQTT- the protocol for internet of things// Innovations in Science and Technology: the XVI All-Ukrainian Students R&D Conference Proceedings, (Kyiv, April 18, 2016)/ National Technical University of Ukraine ‘Kyiv Polytechnic Institute’. Part II.- Kyiv, 2016. pp. 48-49.

2. Кузянін О.С. MQTT- протокол технології «інтернет речей»/ Міжнародна науково- практична конференція “Перспективи розвитку сучасної науки”: матеріали конференції 6-7 травня 2016 року Чернігів, Україна, с. 67-70.

3. Кузянин А.С. Шаги повышения безопасности технологии Internet of Things, IoT/ XXII Международная научная конференция “Актуальные вопросы современной техники и технологии”: 22 апреля 2016 г., г. Липецк, с. 9-11.

4. Industry 4.0 and the digital twin 2017. URL: https://www2.deloitte.com/insights/us/en/focus/industry–4–0/digital–twin–technology–smart–factory.html#endnote–11 (дата звернення 14.11.2019)

5. Архитектура безопасности "Интернета вещей" 2018. URL: https://docs.microsoft.com/ru–ru/azure/iot–suite/iotsecurity–architecture (дата звернення 17.11.2019)

6. Росляков А. «Интернет вещей» : Учебное пособие / А. В. Росляков, С. В. Ваняшин, А. Ю. Гребешков. – Самара : ПГУТИ, 2015 – 136 с.

7. Лісковський І. «Телекомунікаційні та інформаційні технології» / Аналітичний метод визначення параметрів якості для системи черг у мережному вузлі 2014 с. 61–66.

8. Лісковський І. «Дослідження надійності функціонування мережі тактової синхронізації» "Вісник ДУІКТ" т. №1 – 2013, с.111–116.

9. Історія інтернету речей / URL: https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot (дата звернення 11.11.2019)

10. Каштанов В.А. Теория надежности сложных систем : (Теория и практика) / В.А. Каштанов, А.И. Медведев. - М. : Европ. центр по качеству, 2002. - 469 с.

11. MQTT и Modbus: сравнение протоколов, используемых в шлюзах для IoT – URL: https://habrahabr.ru/company/intel/blog/304228/ (дата звернення 09.11.2019)

12. Download Raspbian for Raspberry Pi – URL: https://www.raspberrypi.org/downloads/raspbian/ – Дата доступу : 22.11.2019.

13. Муршед Ф. А. Использование имитационного моделирования для администрирования систем массового обслуживания // Вестник технологического университета. 2017. Т.20, №1. С. 125-127

14. How to get docker–compose (fig) up and running on Raspberry Pi 2 – URL: https://gist.github.com/oysteinjakobsen/e59cdd38a688ee8a418a (дата звернення 11.11.2019)

15. How to run a Raspberry Pi cluster with Docker Swarm – URL: https://howchoo.com/g/njy4zdm3mwy/how–to–run–a–raspberry–pi–cluster withdocker–swarm (дата звернення 11.11.2019)

16. Ляхов А.И. Многоканальные mesh–сети: анализ подходов и оценка производительности / А.И. Ляхов, И.А. Пустогаров, С.А. Шпилев // Информационные процессы. – 2008. – Том 8, № 3. – С. 173–192.

17. Гусс С.В. Самоорганизующиеся mesh–сети для частного использования // Математические структуры и моделирование.: Омск, 2016. — Вып. 4.– С. 102–115.

18. Кравченко В.И. Радиоэлектронные средства и мощные электромагнитные помехи / В.И.Кравченко, Е.А.Болотов, Н.И.Латунова. – М.: Радио и связь, 1987. – 256 с.

19. Нагай В.В. «Спосіб управління пульсуючими потоками протокольних блоків даних» // Технічні науки: зб. науково-практичних конференцій. Серія «Інноватика в сучасній освіті та науці». Херсон, 2019. с.103-105.

20. Ю.А.Кочергін, «Спосіб управління пульсуючими потоками протокольних блоків даних», МПК : H04L12/50. №42903, 03.03.2009.

21. Р.Резайіфар, «Протоколи радіозв'язку для багатоканальних систем зв'язку», МПК H04L 29/06. №95229, 25.07.2011.

22. А.Мантраваді, «Спосіб і пристрій для передачі інформації в системі, яка використовує різні протоколи передачі», МПК H04L 25/02. №88026, 25.01.2019.

23. Ляхов А.И. Многоканальные mesh-сети: анализ подходов и оценка производительности / А.И. Ляхов, И.А. Пустогаров, С.А. Шпилев // Информационные процессы. – 2008. – Том 8, № 3. – С. 173-192.

24. Гук М. Аппаратные средства локальных сетей. Энциклопедия – Спб.: Питер, 2005. – 573 с.