НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Навчально-науковий комплекс "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004.03

«До захисту допущено»
Завідувач кафедри
_____ А.І. Петренко
(підпис) (ініціали, прізвище)
“___”_____2017 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 8.05010103 Системне проектування
(код і назва)

на тему: Дослідження мікроконтролерів та міні комп’ютерів у IoT пристроях

Виконав: студент 6 курсу, групи ДА-52м
(шифр групи)

Чернацький Артем Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., Харченко К.В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту доцент, к.т.н., Харченко К.В. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент Завідувач кафедри обчислювальної техніки, д.т.н., професор, Стіренко С.Г. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»

Інститут (факультет) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І. Петренко
(підпис) (ініціали, прізвище)
«___»_____2017 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Чернацькому Артему Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Дослідження мікроконтролерів та міні комп'ютерів у IoT пристроях,
науковий керівник дисертації Харченко Костянтин Васильович, к.т.н., доц. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «20» березня 2017 р. № 32-ст

2. Термін подання студентом дисертації _____

3. Об'єкт дослідження Мікроконтролери та міні комп'ютери у IoT пристроях

4. Предмет дослідження Методи реалізації та протоколи взаємодій мікроконтролерів та міні комп'ютері як IoT пристроїв

5. Перелік завдань, які потрібно розробити
   1. Провести систематизацію, порівняння поширених мікроконтролерів та міні ПК, а також виконати їх навантажувальне тестування
   2. Провести аналіз протоколів передачі даних, а також засобів захисту інформації для IoT пристроїв
   3. Дослідити можливість паралельних обчислень на IoT пристроях
   4. Розробити зразок IoT системи

5. Надати практичні рекомендації щодо реалізації систем IoT

6. Орієнтовний перелік ілюстративного матеріалу презентація на тему: « Дослідження мікроконтролерів та міні комп'ютерів у IoT пристроях»

7. Орієнтовний перелік публікацій

Чернацкий А.Ю. Система автоматичного моніторингу параметрів приміщення на базі Arduino Mega 2560 / Перевертайло В.В., Майкут С.О., Чернацький А.Ю. // Системний аналіз та інформаційні технології: материалы 17-й Міжнародної науково-технічної конференції SAIT 2015, Київ, 22-25 червня 2015 р. – 2015. – с.269

Чернацкий А.Ю. Система автоматичного сканування параметрів приміщення / Перевертайло В.В., Майкут С.О., Чернацький А.Ю. // Тези доповідей XII Всеукраїнської наукової конференції студентів і молодих вчених (м. Київ, 21–22 квітня 2015 р.) – 2015. – с.362

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка стартап-проекту	Харченко К.В., к.т.н., доцент		

9. Дата видачі завдання 06.02.2017

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	2	3	4
1	Отримання завдання	06.02.2017	
2	Збір інформації	20.02.2017	
3	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	05.03.2017	
4	Порівняння характеристик мікроконтролерів та міні ПК	15.03.2017	
5	Навантажувальне тестування мікроконтролерів та міні ПК	01.04.2017	
6	Дослідження можливостей протоколів даних для взаємодії IoT пристроїв	08.04.2017	
7	Аналіз засобів захисту інформації при роботі з IoT пристроями	15.04.2017	
8	Дослідження можливостей паралельних обчислень мікроконтролерів та міні ПК	01.05.2017	
9	Розробка зразка IoT системи	15.05.2017	

1	2	3	4
10	Оформлення дипломної роботи	31.05.2017	
11	Отримання допуску до захисту та подача роботи в ДЕК	09.06.2017	

Студент _____ А.Ю. Чернацький
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ К.В. Харченко
(підпис) (ініціали, прізвище)

РЕФЕРАТ

на магістерську дисертацію

виконану на тему: Дослідження мікроконтролерів та міні комп'ютерів у IoT пристроях

студентом: Чернацьким Артемом Юрійовичем

Робота виконана на 107 сторінках, містить 48 ілюстрацій, 62 таблиці. При підготовці використовувалася література з 27 різних джерел.

Актуальність теми

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 та підвищити комфорт життя. Усе це створює чудові передумови до росту цієї галузі.

На даний момент на ринку є безліч IoT пристроїв, протоколів передачі даних, мікроконтролерів та міні комп'ютерів для побудови таких систем. На жаль, інформація про пристрої, які можуть бути базою, а також різноманітні технології і способи взаємодії між такими пристроями є досить розрізненою.

Мета і задачі дослідження

Метою магістерської дисертації є проведення систематизації, аналізу, порівняння та навантажувального тестування існуючих на ринку популярних мікроконтролерів та міні комп'ютерів, дослідження поширених відкритих протоколів взаємодій IoT пристроїв та Middleware, аналізу захисту інформації у них, а також розробка зразка IoT системи та надання практичних рекомендацій, що до розробки подібних систем.

Рішення поставлених завдань та досягнуті результати

В даній роботі було проаналізовано можливості мікроконтролерів та міні ПК від таких виробників як Arduino, Espressif, Raspberry та Orange Pi. У роботі проведено дослідження можливостей різних пристроїв на мовах програмування C++, Java та Python, з метою їх використання для побудови IoT систем. В якості технологій взаємодії було проаналізовано такі технологій як REST, Socket, MQTT та SNMP. Також було виконано навантажувальне тестування пристроїв і дослідження їх можливостей для паралельних обчислень, зокрема у вигляді кластера Docker Swarm та з допомогою потоків.

В ході виконання роботи було зроблено висновки та надано рекомендації що до оптимальних можливостей використання міні ПК та мікроконтролерів, а також розроблено зразок IoT системи з використанням Node-Red для візуалізації та Docker Compose для розгортання серверної частини, в якості протоколу взаємодії було обрано MQTT, а датчики були побудовані на базі ESP8266.

Об’єкт дослідження

Об’єктом дослідження є мікроконтролери та міні комп’ютери у IoT пристроях.

Предмет дослідження

Предметом дослідження є методи реалізації та протоколи взаємодій мікроконтролерів та міні комп’ютері як IoT пристроїв.

Наукова новизна

Наукова новизна роботи полягає в систематизації і порівняльній характеристиці мікроконтролерів та міні комп’ютерів, технологій їх взаємодії як IoT пристроїв та наданні рекомендацій по їх вибору у відповідності до обраних критеріїв.

Ключові слова

IoT, Arduino, ESP, Docker, Raspberry, Orange Pi, мікроконтролер, міні ПК, паралельні обчислення, Node-Red

ABSTRACT

on master's thesis

on topic: Research of microcontrollers and mini computers in IoT devices

student: Artem Chernatsky

Work carried out on 107 pages containing 48 figures, 62 tables. The paper was written with references to 27 different sources.

Topicality

Today IoT development trend of information technology is extremely popular. Every year more and more companies are entering the market with various solutions in this area. The introduction of IoT technologies in the environment can significantly improve the use of non-renewable energy resources, simplify life and increase the comfort of life. All this creates excellent preconditions for the growth of this industry.

Currently on the market there are many IoT devices, data transfer protocols, microcontrollers and mini computers to build such systems. Unfortunately, information about devices that can be a base, and a variety of technologies and methods of interaction between these devices is rather fragmented.

Aims and objectives

The purpose of the master's thesis is organizing, analyzing, comparing and load testing of existing market popular microcontrollers and mini computers, common research protocols open interactions IoT devices and Middleware, information security analysis of them, and the development model IoT systems and provide practical recommendations that the development of such systems.

The solution of the tasks and results

This paper analyzed the possibility of microcontrollers and mini PCs from manufacturers such as Arduino, Espressif, Raspberry and Orange Pi. In this paper, the research capabilities of different devices on the programming languages C ++, Java and Python, to be used for the construction of IoT. As technology interaction analyzed such technologies as REST, Socket, MQTT and SNMP. It was also made handling devices and test their research capabilities for parallel computing, in particular in the form of cluster Docker Swarm and through streams.

During the job was done conclusions and recommendations regarding the best possibilities of using mini PC and microcontrollers, and designed like the IoT system using Node-Red for visualization and Docker Compose to deploy the server side, as the protocol interaction was chosen MQTT and sensors have been built on the basis ESP8266.

The object of research

Object is a microcontroller and mini computers in IoT devices.

Subject of research

The subject of research is the implementation methods and protocols microcontroller interactions and mini computer as IoT devices.

Scientific novelty

Scientific novelty consists in organizing and comparative characterization of microcontrollers and mini computer technology as they interact IoT devices and advice on their choice according to selected criteria.

Keywords

IoT, Arduino, ESP, Docker, Raspberry, Orange Pi, microcontroller, mini PC, parallel computing, Node-Red

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IoT	Internet of Things, Інтернет речей
framework	готовий до використання комплекс програмних рішень
ПЗ	програмне забезпечення
БД	база даних
скетч	блок коду для Arduino
IT	інформаційні технології
ОС	операційна система
SNMP	Simple Network Management Protocol
MQTT	Message Queue Telemetry Transport
REST	Representational State Transfer
SSH	Secure Shell
IP	Internet Protocol

ВСТУП

У наші дні спостерігається бурхливий ріст та розвиток інформаційних технологій. Те, що здавалось фантастикою декілька десятиліть назад, зараз є буденною звичайністю. Швидкість зростання обсягів даних та обчислювальних потужностей надзвичайно стрімкий. Сучасні мобільні пристрої зазвичай мають технічні характеристики в рази кращі, ніж комп'ютери минулого, а їхня собівартість зменшується з кожним роком.

За такого рівня технічного прогресу у більшості звичайних побутових пристроїв вже є власний процесор, що керує ним та дозволяє автоматизувати роботу. В результаті таких досягнень виникає питання, а чи можливо створити єдину систему контролю пристроїв, що нас оточують? Чи можна автоматизувати більшу частину побутових процесів?

Відповіддю на ці питання стали розробки IoT пристроїв та систем. Багато різноманітних відомих та не дуже компаній займаються розробкою та створенням таких систем.

Основними перевагами системи "розумних" пристроїв є можливість автоматизації їх роботи та віддаленого контролю. Це дозволяє автоматично контролювати параметри мікроклімату у приміщенні, вмикати та вимикати різного роду пристрої, надсилати повідомлення користувачам при зміні параметрів та навіть автоматично розміщувати пости у соціальних мережах.

Іншою корисною функцією даної системи є реалізація можливості енергозбереження. Якщо система контролює охолоджувальні та нагрівальні установки, то вона здатна ефективно витрачати енергію, адже очевидним є той факт, що немає ніякої необхідності підтримувати комфортну температуру у приміщені за умови відсутності господаря. Для нашої країни енергозбереження є дуже актуальним питанням, адже ціни на енергоносії суттєво зросли за останній

час.

Так що ж собою представляє IoT? Найкраще суть концепції IoT пристроїв характеризують слова Роба Ван Краненбурга: «IOT - концепція простору, в якому все з аналогового і цифрового світів може бути поєднане - це перевизначить наші стосунки з об'єктами, а також властивості і суть самих об'єктів.». Загалом під реалізацією концепції IoT розуміють деяку мережу, у якій фізичні предмети, які оснащені вбудованими засобами взаємодії, взаємодіють між собою та з оточуючим середовищем. Концепція включає наступні 4 напрями технологій:

- ідентифікації
- засоби вимірювання
- засоби передачі даних
- протоколи взаємодії пристроїв

З ростом популярності IoT напрямку у інформаційних технологіях багато виробників почало розробляти та представляти свої рішення пристроїв, протоколів та технологій взаємодії, які можуть стати базою для IoT. Минули ті часи, коли найрозповсюдженішими міні ПК були Raspberry, а мікроконтролерами Arduino, все змінюється і зараз вже з’явилось безліч конкурентів, які пропонують безліч аналогів таких пристроїв.

На даний момент на ринку є безліч IoT пристроїв, протоколів передачі даних, мікроконтролерів та міні комп’ютерів для побудови таких систем. На жаль, інформація про пристрої, які можуть бути базою, а також різноманітні протоколи та способи взаємодії між такими пристроями є досить розрізненою.

Метою магістерської дисертації є проведення систематизації, аналізу, порівняння та навантажувального тестування існуючих на ринку популярних мікроконтролерів та міні комп’ютерів, дослідження поширених відкритих протоколів взаємодій IoT пристроїв та Middleware, аналізу захисту інформації у

них, а також розробка зразка IoT системи та надання практичних рекомендацій, що до розробки подібних систем.

Об’єктом дослідження є мікроконтролери та міні комп’ютери як основа для пристроїв IoT систем.

Предметом дослідження є методи реалізації та протоколи взаємодій мікроконтролерів та міні комп’ютері як IoT пристроїв.

1 КОНЦЕПЦІЯ IOT

Найкраще суть концепції Роба Ван Краненбурга «IOT - концепція простору, в якому все з аналогового і цифрового світів може бути поєднане - це перевизначить наші стосунки з об'єктами, а також властивості і суть самих об'єктів.».

Інтернет речі – це концепція деякої обчислювальної мережі, у якій розташовані фізичні предмети (речі), які оснащені вбудованими засобами для взаємодії між собою та з зовнішнім середовищем. Організація об’єднання таких мереж розглядається як явище, що буде здатне змінити економічні та суспільні процеси, що виключить з частини дій і операцій з необхідністю участі людини. [1]

1.1 Історія

Можна сказати, що сама ідея зародилася у далекому 1926, коли Нікола Тесла у інтерв’ю для журналу Colliers заявив: «В майбутньому радіо буде перетворено в «великий мозок», всі речі стануть частиною єдиного цілого, а інструменти, завдяки яким це стане можливим, будуть легко вміщатися в кишені.»[2].

У 1990 році з’явилася перша інтернет річ. Це був тостер Джона Ромкі, який він під’єднав ло мережі.[3]

У 1999 році вперше з’явився сам термін «Інтернет речей» (Internet of Things, IOT), він був запропонований Кевіном Ештоном.[3]

У 2008-2009 роках, за оцінками компанії Cisco, відбувся перехід від «інтернету людей», до «інтернету речей», тобто кількість підключених пристроїв до інтернету стала більша, за населення планети, дані наведено на рис. 1.1. [1]

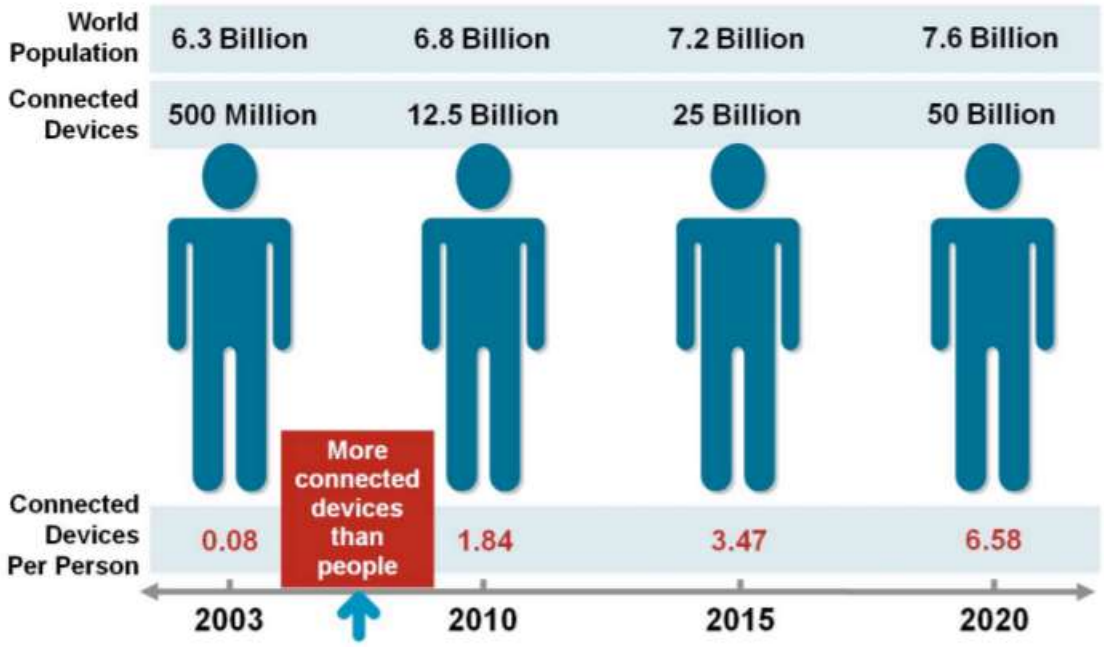

Рисунок 1.1 – Ріст кількості пристроїв мережі інтернет

1.2 Технології IoT

Концепція IoT включає наступні 4 напрями технологій:

- ідентифікації
- засоби вимірювання
- засоби передачі даних
- протоколи взаємодії пристроїв

Під технологіями ідентифікації розуміють засоби реєстрації пристроїв, інтернет мереж це MAC адреси.[1]

До засобів вимірювання відносяться різноманітні датчики та сенсори, що дозволяють перетворити аналогові дані навколишнього середовища у цифрові.[3]

Засобами передачі даних виступають протоколи підключення до мережі, такі як Wi-Fi, ZigBee, Z-wave та інші.[3]

Протоколами взаємодії можуть бути WEB, SSH, MQTT та інші. [1]

Отже, як можна побачити питань для розробки у IoT дуже багато. За прогнозами Cisco до 2020 року до мережі буде підключено близько 50 мільярдів пристроїв [1] і, очевидно, що більшість з них будуть як раз IoT, що створює широкий простір для творчості та розробки таких пристроїв.

2 АНАЛІЗ МОЖЛИВОСТЕЙ МІКРОКОНТРОЛЕРІВ ТА МІНІ КОМПЮ'ТЕРІВ

Так чи інакше можливості IoT пристроїв залежать від апаратної платформи на якій вони побудовані. Для вирішення сучасних завдань, які ставляться перед такими пристроями, доцільно використовувати мікроконтролери або міні комп'ютери. У цьому розділі розглянемо популярні рішення, які існують на ринку, а також їх можливості.

2.1 Мікроконтролери Arduino

Фактично ера доступних та простих мікроконтролерів розпочалася з появою рішень від Arduino. Вдала реалізаціє разом з нескладним використанням та численними інструкціями по роботі з ними дозволили кожному бажаючому долучитися до роботи з мікроконтролерами. Згодом з'явилися плати розширення, які дозволили підключати їх до локальних мереж чи інтернету, що суттєво спростило реалізацію концепції IoT.

На сьогоднішній день існує безліч моделей мікроконтролерів від Arduino. Так як концепція IoT потребує можливості взаємодіяти з іншими пристроями в локальній чи інтернет мережі, то ми не будемо розглядати такі моделі як Nano, Uno чи Mega, які потребують додаткових плат розширень Wi-Fi чи Ethernet, а звернемо увагу на комплексні рішення такі як:

- Arduino Ethernet
- Arduino Yun
- Arduino TIAN

2.1.1 Arduino Ethernet

Arduino Ethernet це фактично Arduino Uno з вбудованою платою розширення Ethernet. Тобто основною відмінністю є присутність RJ – 45 та microSD роз'ємів. Вартість становить приблизно $43. Його зовнішній вигляд наведено на рис 2.1, а технічні характеристики у таблиці 2.1 [4].

Рисунок 2.1 – Arduino Ethernet

Таблиця 2.1 – Характеристики Arduino Ethernet

Параметр	**Значення**
Мікроконтролер	ATmega328
Робоча напруга	5В
Вхідна напруга (рекомендований)	7-12В
Цифрові Входи / Виходи	14 (4 ШІМ-виходи)
Flash-память	32 КБ
SRAM	2 KB
Незалежна пам'ять	1 KB
Тактова частота	16 MHz
Підключення до локальної мережі	Ethernet

2.1.2 Arduino Yun

Arduino YUN - це перший представник новітньої серії плат Arduino з вбудованим WiFi, що поєднують в собі найширші можливості Linux і простоту використання Arduino. Arduino YUN є комбінацією класичного Arduino Leonardo (на базі мікроконтролера ATmega32U4) і WiFi-системи на кристалі, що працює під управлінням Linino (дистрибутив ОС GNU / Linux на основі OpenWRT для мікропроцесорів MIPS). Вартість складає близько $69. Його зовнішній вигляд наведено на рис 2.2, а технічні характеристики у таблиці 2.2 [5].

Рисунок 2.2 – Arduino YUN

Таблиця 2.2 – Характеристики Arduino YUN

Параметр	**Характеристика**
1	**2**
Мікроконтролер	ATmega32u4
Робоча напруга	5В
Вхідна напруга (рекомендований)	5В чи PoE 802.3af
Цифрові Входи / Виходи	14 (7 ШІМ-виходи)
Flash-память	32 КБ

1	2
SRAM	2.5 KB
Незалежна пам'ять	1 KB
Тактова частота	16 MHz
Підключення до локальної мережі	Ethernet, Wi-Fi
Процесор	400 МГц
RAM	64 МБ
Flash-память (Процесора)	16 МБ

2.1.3 Arduino TIAN

Найпотужніший Arduino, який побудоано на базі 32 бітного процессора з можливістю підключення по Wi-FI 2.4/5 ГГц та Ethernet. Вартість приблизно $92. Його зовнішній вигляд наведено на рис 2.3, а технічні характеристики у таблиці 2.3 [6].

Рисунок 2.3 – Arduino TITAN

Таблиця 2.3 – Характеристики Arduino TITAN

Параметр	Характеристика
Мікроконтролер	SAMD21G18
Робоча напруга	3.3 В
Вхідна напруга (рекомендований)	5В
Цифрові Входи / Виходи	14 (12 ШІМ-виходи)
Flash-память	256 КБ
SRAM	32 КВ
Незалежна пам'ять	4 КВ
Тактова частота	48 MHz
Підключення до локальної мережі	Ethernet, Wi-Fi
Процесор	560 МГц
RAM	64 МБ
Flash-память (Процесора)	16 МБ + 4 ГБ eMMC

Таким чином сучасні мікроконтролери від Arduino мають досить високу вартість та за характеристиками власне чіпів мікроконтролерів мало відрізняються від попередніх проектів, позитивним є чудова їх підтримка, яка дозволяє початківцям отримати навички з роботи з ними.

2.1.4 Оцінка обчислювальної потужності Arduino

Розглянемо обчислювальні потужності контролерів на базі чіпів Atmega328 та Atmega2560. Для оцінки будемо додавати елементи short int масиву та виконувати get запити. Для роботи будемо використовувати фреймворк Arduino та середу розробки PlatformIO. Дані по роботі з масивами наведено у таблицях 2.4 та 2.5.

Таблиця 2.4 – Робота з масивом на Atmega 328

Розмір масиву	Спроба, мкс					Середній
	1	2	3	4	5	
500	5416	5320	5408	5488	5516	5429.6
900	9712	9660	9964	9876	10116	9865.6

Таблиця 2.5 – Робота з масивом на Atmega 2560

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	5752	5800	5692	5880	5760	5776.8
1000	12184	11360	11624	12368	12176	11942.4
2500	29752	30164	30480	30056	28556	29801.6
3500	40972	41308	40728	42516	42968	41698.4

В якості тесту мережевого з’єднання перевіримо виконання запитів get сайту www.uacrr.org. Для підключення до мережі інтернет скористаємося платою розширення з чіпом w5100 та зробимо по 5 спроб по 10 запитів у кожній. Дані наведено у таблиці 2.6.

Таблиця 2.6 – Робота Arduino з get

Чіп	Спроба (10 заптів), мкс					Середній, мкс	1 запит у середньому, мкс
	1	2	3	4	5		
atmega328	11344144	13172020	13153576	11312408	13159716	12428372.80	1242837.28
atmega2560	13188420	13423952	11393332	13223156	13186344	12883040.80	1288304.08

2.2 Мікроконтролери ESP

Поява мікроконтролерів від Espressif стала наступним суттєвим кроком у розвитку IoT. Якщо продукти від Arduino дозволило кожному бажаючому розпочати програмування мікроконтролерів, то продукти від Espressif дали

можливість так само легко об’єднувати їх у локальні мережі за допомогою Wi-Fi, збільшивши при цьому їх потужність та зменшивши собівартість.

2.2.1 ESP-8266

Мікроконтролер ESP-8266 з’явився у 2014 році. Він дозволяє за допомогою вбудованого Wi-Fi підключатися до мереж та оновлювати ПЗ. Існує велика кількість модифікацій, розглянемо останню, яка побудована на чіпі ESP-12 від WEMOS Electronic. Вартість складає $3-5. Його зовнішній вигляд наведено на рис 2.4, а технічні характеристики у таблиці 2.7 [7].

Рисунок 2.4 – WEMOS D1 mini pro

Таблиця 2.7 – Характеристики ESP-8266

Параметр	Характеристика
Мікроконтролер	ESP-8266EX
Робоча напруга	3.3 В
Вхідна напруга (рекомендований)	5 В
Цифрові Входи / Виходи	11
Flash-память	4/16 MB
Тактова частота	80/120 MHz
Підключення до локальної мережі	Wi-Fi

2.2.2 ESP-32

У 2015 році компанія Espressif представила новий мікроконтролер ESP-32. У ньому було встановлено Bluetooth модуль на додачу до Wi-Fi та процесор з 2 ядрами. Орієнтовна вартість $15-30. Його зовнішній вигляд наведено на рис 2.5, а технічні характеристики у таблиці 2.8 [8].

Рисунок 2.5 – SparkFun ESP32 Thing

Таблиця 2.8 – Характеристики SparkFun ESP32 Thing

Параметр	**Характеристика**
Мікроконтролер	ESP-32
Робоча напруга	2.2 - 3.6 В
Вхідна напруга (рекомендований)	5 В
Цифрові Входи / Виходи	28
Flash-память	4 MB
Тактова частота	240 MHz 2 ядра
Підключення до локальної мережі	Wi-Fi

Таким чином, можна вважати що продукти від Espressif є чудовим поєднанням ціни та продуктивності.

2.2.3 Оцінка обчислювальної потужності ESP

Порівняємо роботу ESP з таким самим масивом як і Arduino та get запитами. Для роботи будемо використовувати фреймворк Arduino та середу розробки PlatformIO.

У таблицях 2.9, 2.10, 2.11 наведено дані по роботі з масивами для чіпів ESP8266 на частотах 80МГц та 160МГц та ESP32.

Таблиця 2.9 – Робота ESP8266 80 МГц з масивом

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	743	706	723	703	701	715.2
1000	1468	1469	1408	1441	1396	1436.4
2500	3487	3507	3647	3517	3495	3530.6
10000	14096	14191	14274	14082	14086	14145.8
20000	28002	28271	28571	28325	28293	28292.4

Таблиця 2.10 – Робота ESP8266 160 МГц з масивом

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	369	372	352	350	367	362
1000	717	700	730	703	710	712
2500	1776	1754	1762	1749	1789	1766
10000	7122	7056	7006	7039	6990	7042.6
20000	13986	14178	14095	14071	13952	14056.4

Таблиця 2.11 – Робота ESP32 з масивом

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	239	282	283	295	205	260.8
1000	431	410	439	407	408	419
2500	1061	1030	1046	1035	1054	1045.2
10000	4235	4147	4123	4079	4149	4146.6
20000	8387	8261	8196	8269	8406	8303.8
50000	20786	20864	20368	20325	20320	20532.6

Перевірку за get запитів виконаємо аналогічно Arduino, відмінністю буде підключення до мережі інтернет через вбудований модуль Wi-Fi. Дані наведено у таблиці 2.12.

Таблиця 2.12 – Робота ESP з get запитами

Чіп	Спроба (10 заптів), мкс					Середній, мкс	1 запит у середньому, мкс
	1	2	3	4	5		
esp8266 80mhz	10198784	10094795	10116844	10244791	10131784	10157399.60	1015739.96
esp8266 160mhz	10189839	10176841	10094827	10080844	10094830	10127436.20	1012743.62
esp32	10545813	10386885	10081844	10388835	10436859	10368047.20	1036804.72

2.3 Міні комп’ютери з ОС LINUX

Альтернативою мікроконтролерам при побудові IoT пристроїв можуть виступати міні комп’ютери з ОС Linux. Існує безліч варіантів їх реалізації для використання в IoT необхідна наявність GPIO та бажано компактні розміри. Керуючись з цими міркуваннями розглянемо пристрої від наступних виробників:

- Raspberry Pi
- NanoPi

- Orange Pi

2.3.1 Raspberry Pi

Перші Raspberry Pi з’явилися у 2012 році. Саме ці пристрої поклали основу міні комп’ютерам, які ентузіасти почали використовувати для побудову IoT пристроїв, потужніших за Arduino.

На даний момент найменшим є Arduino Zero. Він не має Ethernet чи Wi-Fi модулів і вимагає їх підключення через OTG. Анонсована вартість складає $5, але фактично придбати його можливо лише за набагато більшу вартість. Його зовнішній вигляд наведено на рис 2.6, а технічні характеристики у таблиці 2.13 [9].

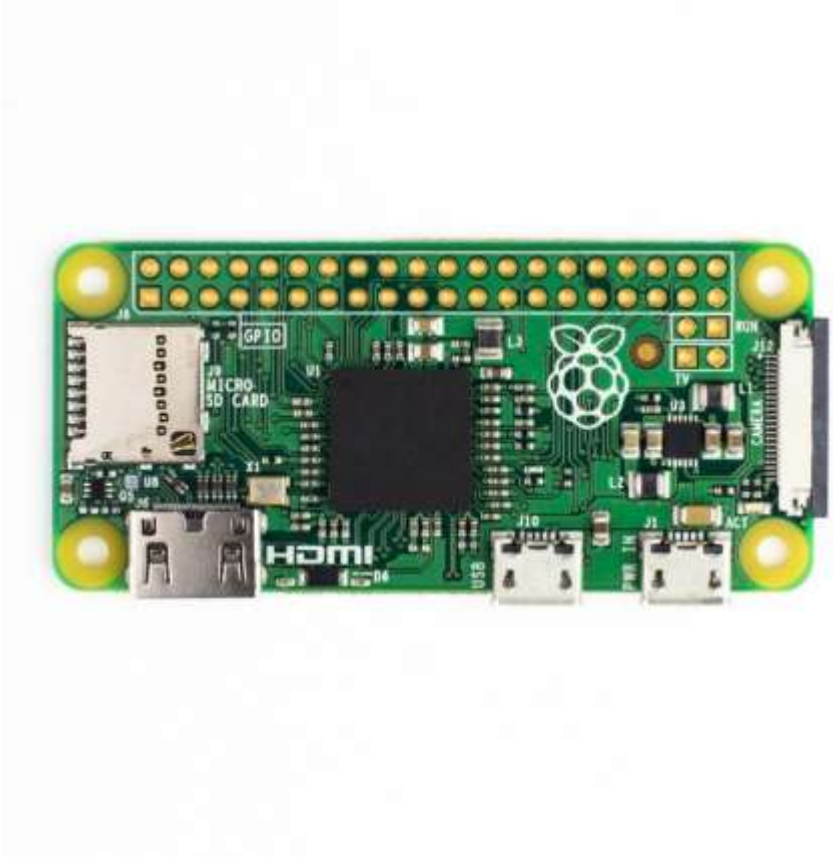

Рисунок 2.6 – Raspberry Pi Zero

Таблиця 2.13 – Характеристики Raspberry Pi Zero

Параметр	Характеристика
Процесор	1Ghz, Single-core CPU
Оперативна пам’ять	512 MB
Пам’ять	- (microSD)
Входи / Виходи	40
USB	OTG
Підключення до локальної мережі	- (зовнішні модулі)

Старшою моделлю є Raspberry Pi 3B, вона має 64 бітний процесор, вбудовані моделі Ethernet, Wi-Fi та Bluetooth, а вартість стартує від $40. Її зовнішній вигляд наведено на рис 2.7, а технічні характеристики у таблиці 2.14 [10].

Рисунок 2.7 – Raspberry Pi 3B

Таблиця 2.14 – Характеристики Raspberry 3B

Параметр	Характеристика
Процесор	1.2GHz 64-bit quad-core ARMv8 CPU
Оперативна пам’ять	1024 MB
Пам’ять	- (microSD)
Входи / Виходи	40
USB	4
Підключення до локальної мережі	Ethernet, Wi-Fi

Таким чином, Raspberry Pi Zero вимагає додаткових модулів для комунікації, що робить його вартість не дуже привабливою. Raspberry Pi 3B має досить потужний процесор та усі необхідні модулі для взаємодії, але за його вартість можна придбати 10 плат ESP-8266, до того ж його потужність у більшості випадків буде надлишковою. Доцільним мабуть було б його використання у якості центрального вузла систем розумний будинок. Також слід зауважити високий рівень підтримки ПЗ Raspberry Pi, він набагато вище ніж у інших виробників.

2.3.2 NanoPi

Після тріумфу з появи Raspberry Pi вже пройшов час і з’явилися аналоги від інших виробників. Розглянемо деякі моделі NanoPi, які є досить компактними для побудови IoT пристроїв.

NanoPi Neon дуже маленький міні комп’ютер розміром трохи більше коробки сірників. Загалом він більш потужний за Raspberry Pi Zero, та має повноцінний USB та вбудований Ethernet модуль. Вартість складає $10-12. Його зовнішній вигляд наведено на рис 2.8, а технічні характеристики у таблиці 2.15 [11].

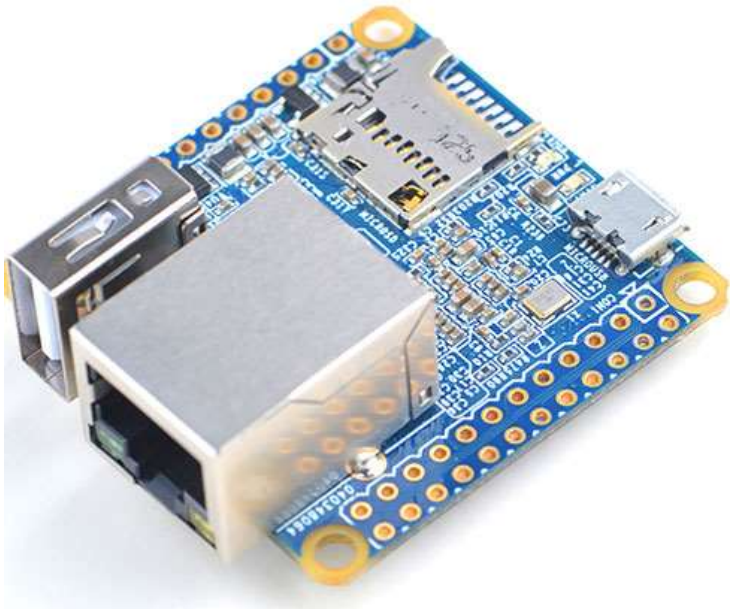

Рисунок 2.8 – NanoPi Neon

Таблиця 2.15 – Характеристики NanoPi Neon

Параметр	**Характеристика**
Процесор	Quad-core Cortex-A7 Up to 1.2GHz
Оперативна пам’ять	256/512 MB
Пам’ять	- (microSD)
Входи / Виходи	36
USB	1
Підключення до локальної мережі	Ethernet

NanoPi-NEO-Air фактично є версією NanoPi Neon з Wi-Fi, без Ethernet та USB з eMMC. Вартість складає $26. Його зовнішній вигляд наведено на рис 2.9, а технічні характеристики у таблиці 2.16 [12].

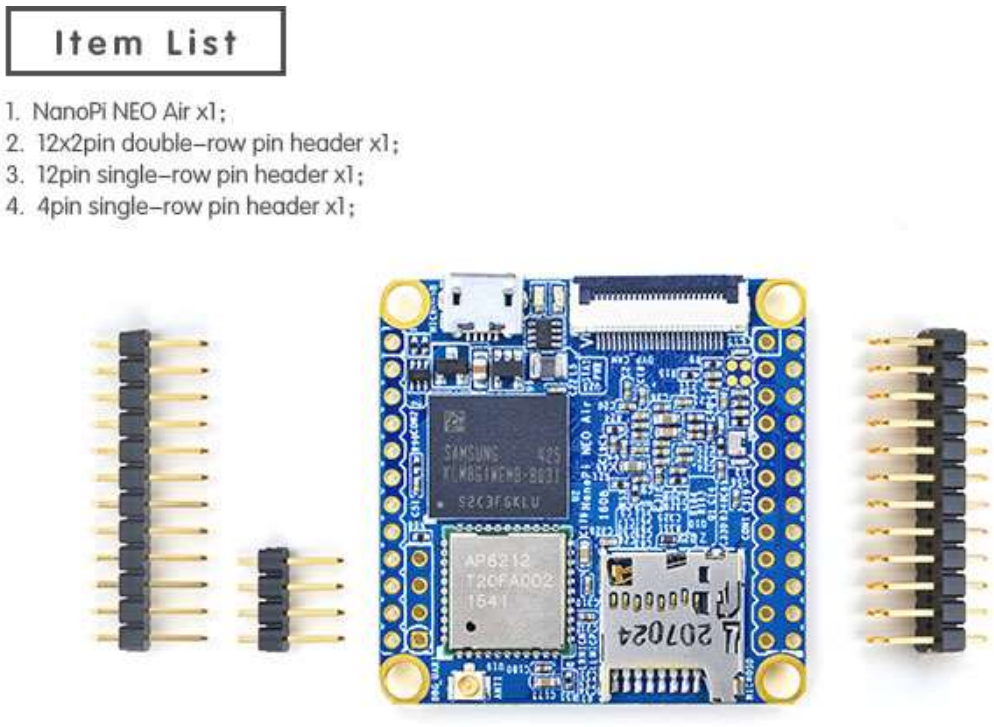

Рисунок 2.9 – NanoPi-NEO-Air

Таблиця 2.16 – Характеристики NanoPi-NEO-Air

Параметр	**Характеристика**
Процесор	Quad-core Cortex-A7 Up to 1.2GHz
Оперативна пам’ять	512 MB
Пам’ять	8GB eMMC
Входи / Виходи	36
USB	OTG
Підключення до локальної мережі	Wi-Fi

2.3.3 Orange Pi

Orange Pi це інший виробник міні комп’ютерів, які можуть виступати альтернативою Rasberry Pi. Orange Pi Zero це міні комп’ютер розміром трохи

більшим коробки сірників. Він має вбудований Ethernet та Wi-Fi модуль. Його вартість складає $7-9. Зовнішній вигляд наведено на рис 2.10, а технічні характеристики у таблиці 2.17 [13].

Рисунок 2.10 – Orange Pi Zero

Таблиця 2.17 – Характеристики Orange Pi Zero

Параметр	**Характеристика**
Процесор	Quad-core Cortex-A7 Up to 1.2GHz
Оперативна пам’ять	256/512 MB
Пам’ять	- (microSD)
Входи / Виходи	26
USB	1
Підключення до локальної мережі	Ethernet + Wi-Fi

2.3.4 Оцінка обчислювальної потужності Linux міні ПК

На міні ПК ми оцінимо роботу з масивом, аналогічним тому, що був на мікроконтролерах та get запитами на мовах програмування:

- C++
- Java
- Python

Порівняємо роботу на чіпах:

- BCM2837 (Raspberry Pi 3B)
- Alwinner H2+ (Orange pi Zero)
- BCM2835 900 МГц (Raspberry Pi B+)
- BCM2835 700 МГц (Raspberry Pi B)

Будемо використовувати Java 1.8, Python 2.7.

Розглянемо можливості Raspberry Pi B. У таблицях 2.18, 2.19, 2.20 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.18 – Робота Raspberry Pi B з масивом на мові C++

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	61	55	58	59	58	58.2
1000	104	102	100	100	96	100.4
2500	232	227	223	223	226	226.2
10000	1231	863	862	1185	1158	1059.8
20000	1793	2003	1794	1980	1715	1857
50000	4536	4700	4578	4570	4699	4616.6
100000	8846	8925	9018	8985	8973	8949.4
500000	43482	43346	43356	43373	43577	43426.8
1000000	86423	86390	86829	86462	86473	86515.4

Таблиця 2.19 – Робота Raspberry Pi B з масивом на мові Java

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	349	341	342	345	343	344
1000	672	716	666	726	667	689.4
2500	1758	1695	1681	1888	1742	1752.8
10000	7147	7019	7412	7424	7450	7290.4
20000	14288	14624	14101	13138	14269	14084
50000	36388	37181	37003	36811	36935	36863.6
100000	72972	73398	74114	73030	71561	73015
500000	393625	385094	394484	390129	392586	391183.6
1000000	1041554	1057389	1041417	1036211	1032176	1041749.4

Таблиця 2.20 – Робота Raspberry Pi B з масивом на мові Python

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	949	756	765	776	857	820.6
1000	1538	1532	1579	1682	1791	1624.4
2500	3781	4164	3877	3964	3822	3921.6
10000	16859	17043	19399	16545	18203	17609.8
20000	52954	51594	51030	41594	66952	52824.8
50000	91659	109398	91459	90147	104782	97489
100000	223719	184523	184428	190008	197432	196022
500000	927385	1169856	910268	943585	943781	978975
1000000	1758225	1793586	2007110	2016870	1820199	1879198

Графік порівняння часу роботи з масивом наведено на рис. 2.11.

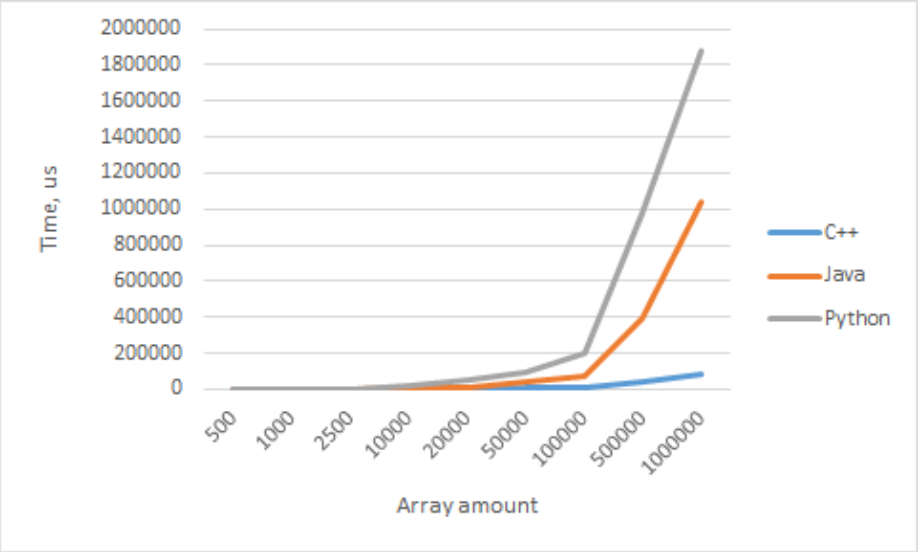

Рисунок 2.11 – Порівняння часу роботи з масивом на Raspberry Pi B

У таблиці 2.21 наведено результати роботи міні ПК Raspberry Pi B з get запитом на мовах C++, Java та Python. Тут слід зауважити, що так як у C++ відсутній стандартній метод, то виконання команді get тут і для всіх інших мікроконтролерів буде реалізовано через виклик стандартного методу wget.

Таблиця 2.20 – Робота Raspberry Pi B з get запитами

	C++	**Java**	**Python**
Спроба 1, мкс	8868	32523321	107062913
Спроба 2, мкс	8992	32138767	106868502
Спроба 3, мкс	7715	32223424	106677047
Спроба 4, мкс	8408	32514622	115084619
Спроба 5, мкс	8114	31958117	128274945
Середній, мкс	8419.4	32271650.2	112793605.2
Середній час на 1 запит, мкс	841.94	3227165.02	11279360.52

Діаграму порівняння часу, необхідного для get запиту, наведено на рис. 2.12.

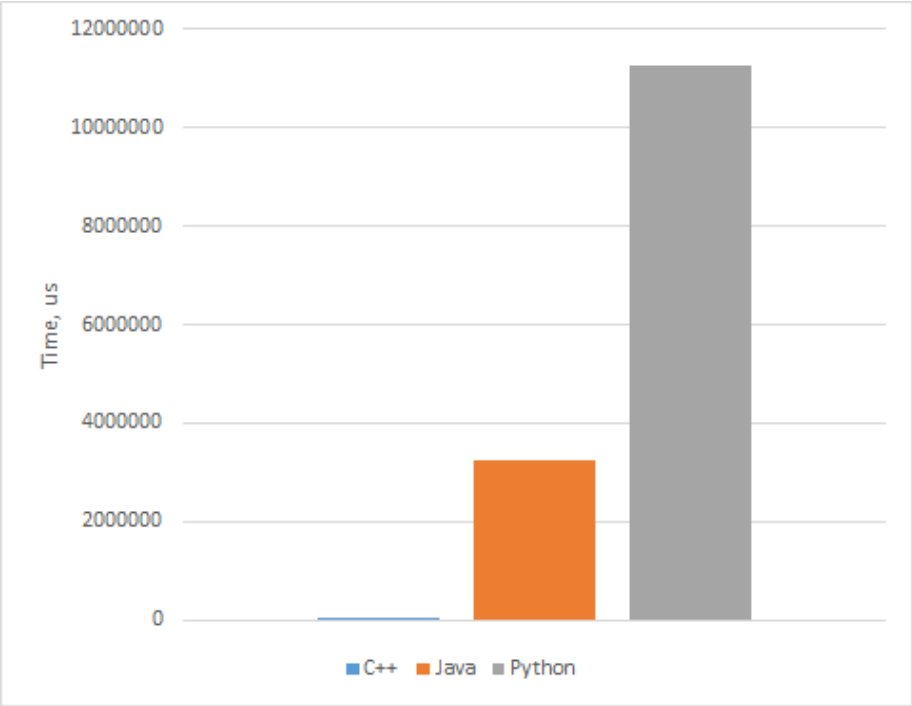

Рисунок 2.12 – Порівняння часу get запиту на Raspberry Pi B

Тепер розглянемо можливості Raspberry Pi B+ на частоті 900 МГц. У таблицях 2.21, 2.22, 2.23 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.21 – Робота Raspberry Pi B+ з масивом на мові C++

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	61	56	59	59	60	59
1000	102	99	100	144	101	109.2
2500	234	228	229	228	226	229
10000	868	862	859	863	859	862.2
20000	1710	1706	1706	1985	1992	1819.8
50000	4632	4613	4407	4245	4591	4497.6
100000	8993	8889	8896	8983	8997	8951.6
500000	43346	43379	43404	43428	43405	43392.4
1000000	86425	86528	86295	86507	86253	86401.6

Таблиця 2.22 – Робота Raspberry Pi B+ з масивом на мові Java

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	346	340	346	339	343	342.8
1000	671	667	665	673	666	668.4
2500	1710	1703	1696	1769	1931	1761.8
10000	7146	7018	7155	7054	7126	7099.8
20000	14159	14120	14281	14432	14119	14222.2
50000	36459	36311	36284	37205	36626	36577
100000	75076	76588	74725	75924	75547	75572
500000	384387	385139	383493	383468	377301	382757.6
1000000	777872	778116	777440	762203	761295	771385.2

Таблиця 2.23 – Робота Raspberry Pi B+ з масивом на мові Python

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	753	773	807	880	911	824.8
1000	1550	1505	1561	1696	1608	1584
2500	4283	3879	3911	3986	3975	4006.8
10000	17992	16464	17359	17078	17750	17328.6
20000	51051	45406	45257	63766	50159	51127.8
50000	84406	84684	86549	98244	86256	88027.8
100000	185350	176530	188459	191624	185390	185470.6
500000	924998	919485	986265	918937	1140068	977950.6
1000000	1764916	1741495	1835357	1802151	1853143	1799412.4

Графік порівняння часу роботи з масивом наведено на рис. 2.13.

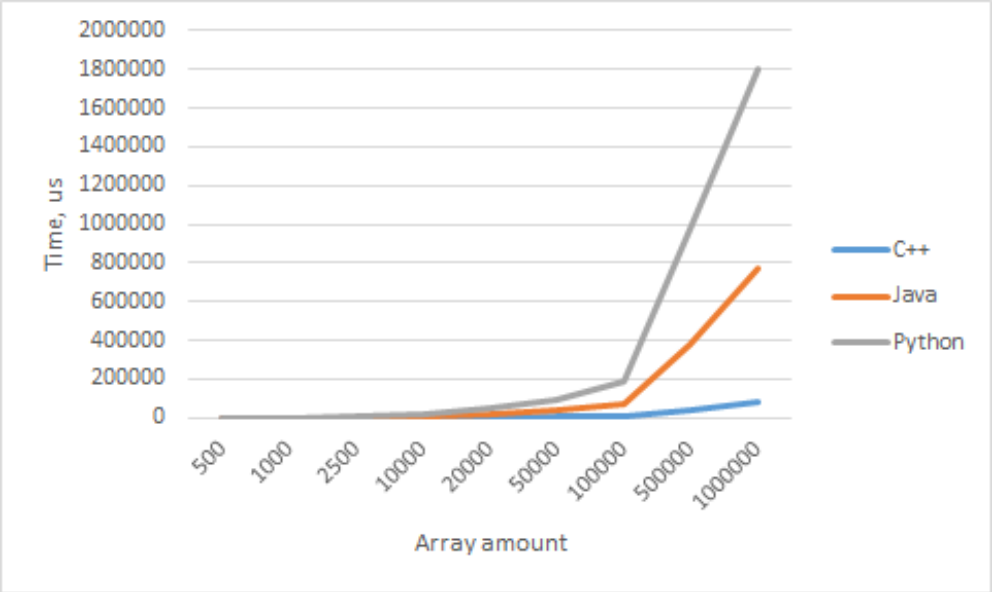

Рисунок 2.13 – Порівняння часу роботи з масивом на Raspberry Pi B+

У таблиці 2.24 наведено результати роботи міні ПК Raspberry Pi B+ з get запитом на мовах C++, Java та Python.

Таблиця 2.24 – Робота Raspberry Pi B+ з get запитами

	C++	**Java**	**Python**
Спроба 1, мкс	7415	31598980	106762354
Спроба 2, мкс	7659	31188980	106781940
Спроба 3, мкс	7179	31366150	106792260
Спроба 4, мкс	7344	31844274	106690711
Спроба 5, мкс	8266	31357745	106879304
Середній, мкс	7572.6	31471225.8	106781313.8
Середній час на 1 запит, мкс	757.26	3147122.58	10678131.38

Діаграма порівняння часу необхідного для get запиту наведено на рис. 2.14.

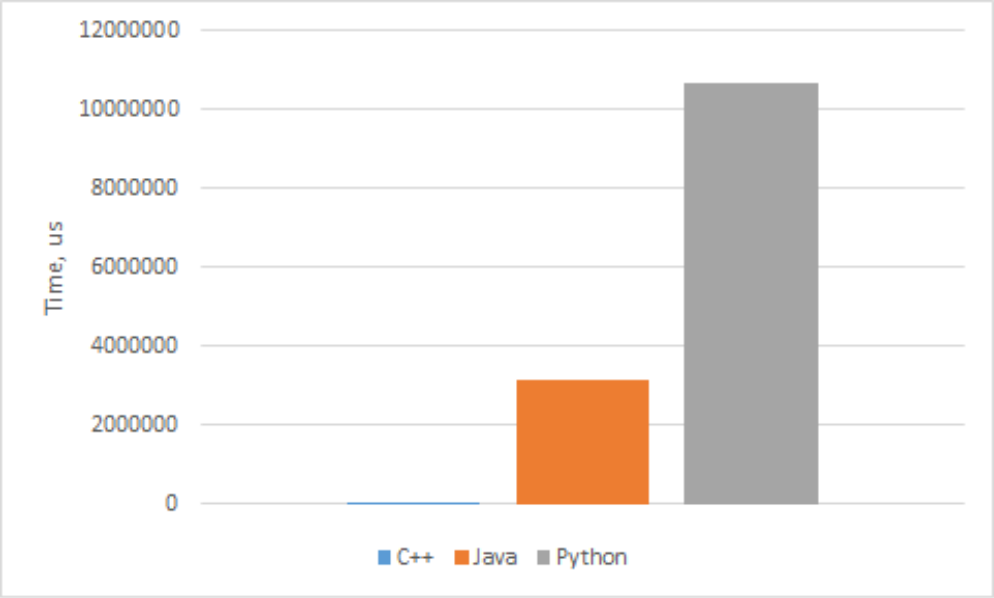

Рисунок 2.14 – Порівняння часу get запиту на Raspberry Pi B

Розглянемо можливості Orange Pi Zero. У таблицях 2.25, 2.26, 2.27 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.25 – Робота Orange Pi Zero з масивом на мові C++

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	38	41	37	38	38	38.4
1000	73	72	72	71	70	71.6
2500	172	169	168	168	168	169
10000	662	657	657	681	660	663.4
20000	1314	1347	1341	1310	1307	1323.8
50000	3321	3264	3288	3299	3264	3287.2
100000	6629	6600	6594	6553	6764	6628
500000	33580	33560	33692	33646	35336	33962.8
1000000	67263	67389	67374	67282	67390	67339.6

Таблиця 2.26 – Робота Orange Pi Zero з масивом на мові Java

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	392	392	392	392	391	391.8
1000	785	773	860	773	774	793
2500	1928	1926	1929	1927	1928	1927.6
10000	7839	7851	7775	7842	7772	7815.8
20000	15720	15686	15905	15797	15547	15731
50000	39041	39376	39107	39038	39243	39161
100000	78647	78064	78935	78632	78295	78514.6
500000	394295	393729	395513	394548	395018	394620.6
1000000	790311	789634	789417	789163	791445	789994

Таблиця 2.27 – Робота Orange Pi Zero з масивом на мові Python

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	884	926	1137	1118	943	1001.6
1000	2175	2184	2200	2235	2184	2195.6
2500	5384	5439	5349	5540	5338	5410
10000	17895	21570	18355	22212	21840	20374.4
20000	37152	44569	37860	46051	45711	42268.6
50000	106288	103782	115603	110312	115053	110207.6
100000	194632	213484	210927	217606	236230	214575.8
500000	1179930	1123158	1170509	1176582	1176763	1165388.4
1000000	1962021	2201430	2312172	2162072	2212021	2169943.2

Графік порівняння часу роботи з масивом наведено на рис. 2.15.

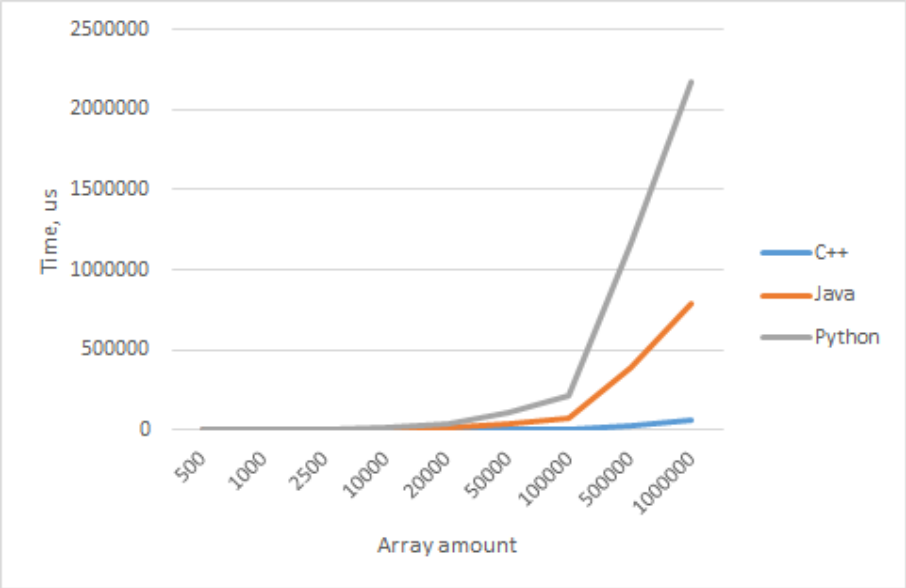

Рисунок 2.15 – Порівняння часу роботи з масивом на Orange Pi Zero

У таблиці 2.28 наведено результати роботи міні ПК Orange Pi Zero з get запитом на мовах C++, Java та Python.

Таблиця 2.28 – Робота Orange Pi Zero з get запитами

	C++	**Java**	**Python**
Спроба 1, мкс	6959	23057360	68104533
Спроба 2, мкс	6562	23107796	65714322
Спроба 3, мкс	7131	22806219	66415161
Спроба 4, мкс	6957	22813429	61833606
Спроба 5, мкс	7018	22802202	74184355
Середній, мкс	6925.4	22917401.2	67250395.4
Середній час на 1 запит, мкс	692.54	2291740.12	6725039.54

Діаграму порівняння часу необхідного для get запиту наведено на рис. 2.16.

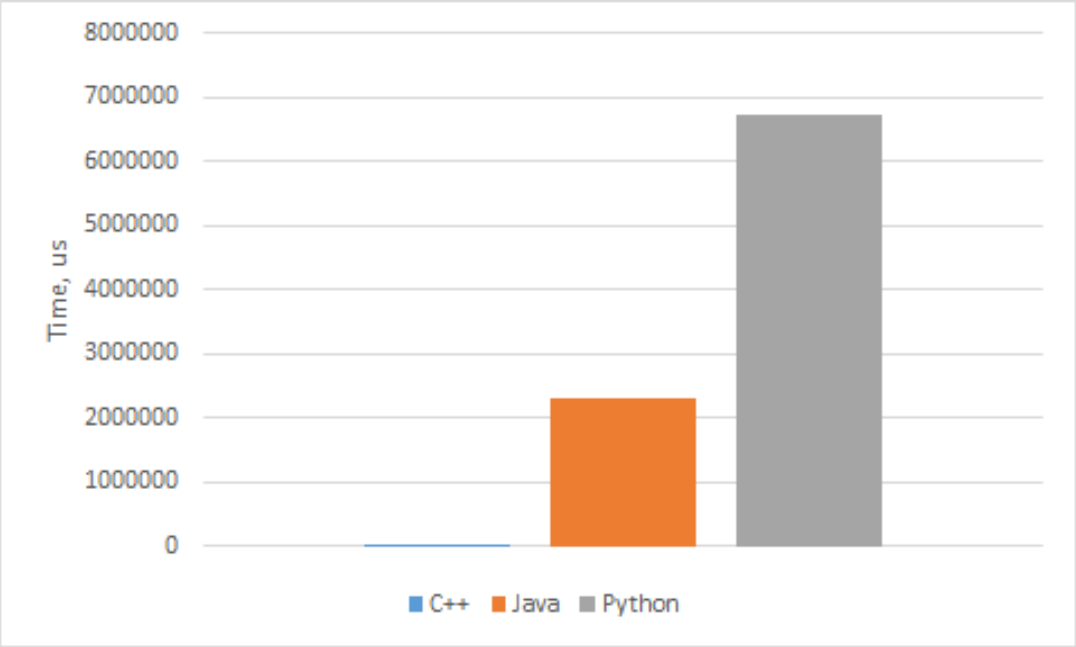

Рисунок 2.16 – Порівняння часу get запиту на Orange Pi Zero

Розглянемо можливості Raspberry Pi 3B. У таблицях 2.29, 2.30, 2.31 наведено результати роботи міні ПК з масивом на мовах C++, Java та Python.

Таблиця 2.29 – Робота Raspberry Pi 3B з масивом на мові C++

Розмір масиву	**Спроба, мкс**					**Середній, мкс**
	1	**2**	**3**	**4**	**5**	
500	31	29	28	29	28	29
1000	56	55	54	54	54	54.6
2500	133	131	129	132	131	131.2
10000	510	512	511	512	511	511.2
20000	1015	1006	1016	1005	1005	1009.4
50000	2562	2533	2551	2531	2534	2542.2
100000	5063	5052	5125	5045	5078	5072.6
500000	25302	25474	25422	25440	25427	25413
1000000	50611	50329	50251	50393	50296	50376

Таблиця 2.30 – Робота Raspberry Pi 3B з масивом на мові Java

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	296	194	295	287	248	264
1000	590	583	586	579	583	584.2
2500	744	737	735	737	749	740.4
10000	2978	2906	2969	2973	2841	2933.4
20000	5991	2790	5895	6010	5833	5303.8
50000	29715	30234	29228	24777	23352	27461.2
100000	60435	60413	60638	58489	58045	59604
500000	291306	293589	299809	291765	291213	293536.4
1000000	583388	501184	583950	598401	588926	571169.8

Таблиця 2.31 – Робота Raspberry Pi 3B з масивом на мові Python

Розмір масиву	Спроба, мкс					Середній, мкс
	1	2	3	4	5	
500	388	411	407	462	411	415.8
1000	772	463	463	894	794	677.2
2500	1928	1004	2193	2215	2199	1907.8
10000	8946	8880	7883	8918	8009	8527.2
20000	18205	18597	19148	18034	16068	18010.4
50000	42322	43337	47294	43026	46712	44538.2
100000	82474	48641	93641	48657	89272	72537
500000	214953	371900	224796	361249	230509	280681.4
1000000	831492	765522	470556	894129	941255	780590.8

Графік порівняння часу роботи з масивом наведено на рис. 2.17.

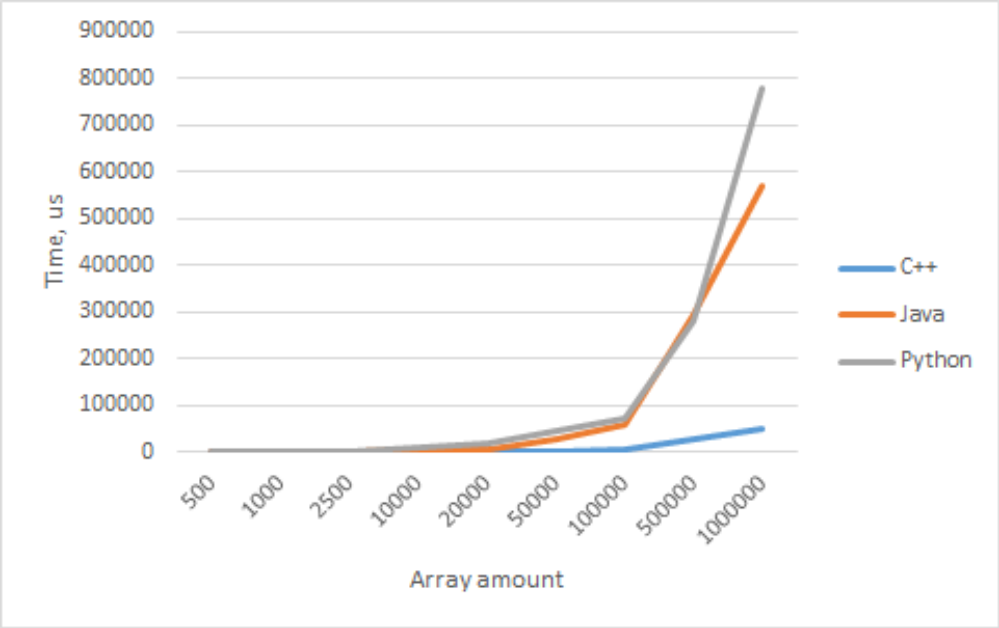

Рисунок 2.17 – Порівняння часу роботи з масивом на Raspberry Pi 3B

У таблиці 2.32 наведено результати роботи міні ПК Raspberry Pi 3B з get запитом на мовах C++, Java та Python.

Таблиця 2.32 – Робота Raspberry Pi 3B з get запитами

	C++	**Java**	**Python**
Спроба 1, мкс	3620	18259217	106498854
Спроба 2, мкс	3711	17480077	106656274
Спроба 3, мкс	3638	17428310	106523145
Спроба 4, мкс	3543	17177134	106621022
Спроба 5, мкс	3438	17625854	106525526
Середній, мкс	3590	17594118.4	106564964.2
Середній час на 1 запит, мкс	359	1759411.84	10656496.42

Діаграму порівняння часу необхідного для get запиту наведено на рис. 2.18.

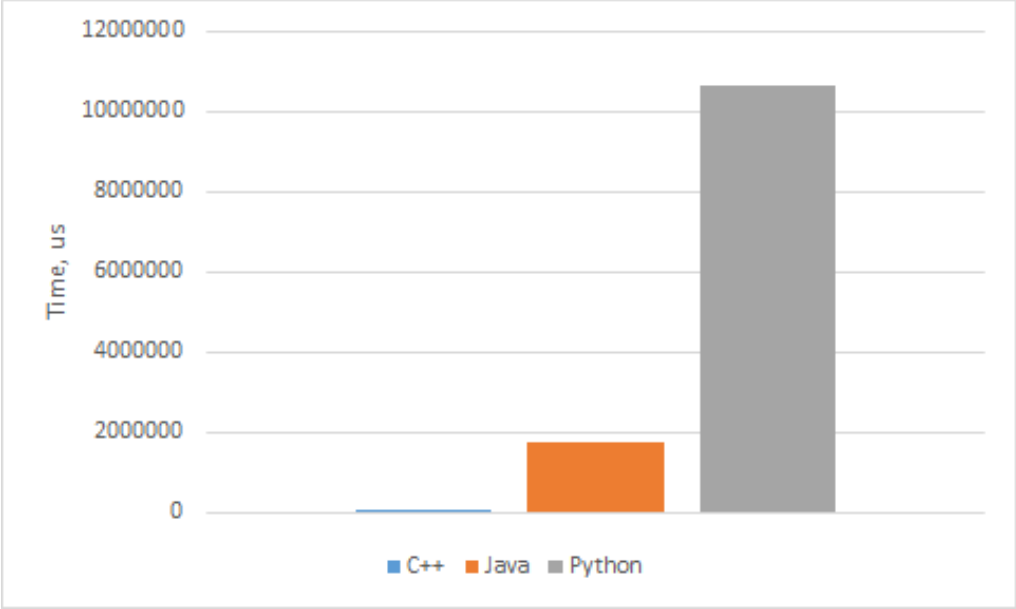

Рисунок 2.18 – Порівняння часу get запиту на Raspberry Pi 3B

2.4 Порівняння обчислювальної потужності

Порівняємо результати тестування для різник міні ПК та мікроконтролерів на мовах C++, Java та Python.

2.4.1 Мова C++

Результати порівняння часу роботи з масивом на мові C++ наведено у таблиці 2.33.

Таблиця 2.33 – Порівняння часу роботи з масивом

Розмір масиву	**Nano**	**Mega 2560**	**ESP8266 80МГц**	**ESP8266 160МГц**	**ESP32**	**Raspberry Pi 1B**	**Raspberry Pi 1B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
500	5430	5777	715	362	261	58	59	38	29
900	9866								

1	2	3	4	5	6	7	8	9	10
1000		11942	1436	712	419	100	109	72	55
2500		29802	3531	1766	1045	226	229	169	131
3500		41698							
10000			14146	7043	4147	1060	862	663	511
20000			28292	14056	8304	1857	1820	1324	1009
50000					20533	4617	4498	3287	2542
100000						8949	8952	6628	5073
500000						43427	43392	33963	25413
1000000						86515	86402	67340	50376

Графік порівняння часу роботи усіх пристроїв з масивом наведено на рис. 2.19, міні ПК – рис. 2.20.

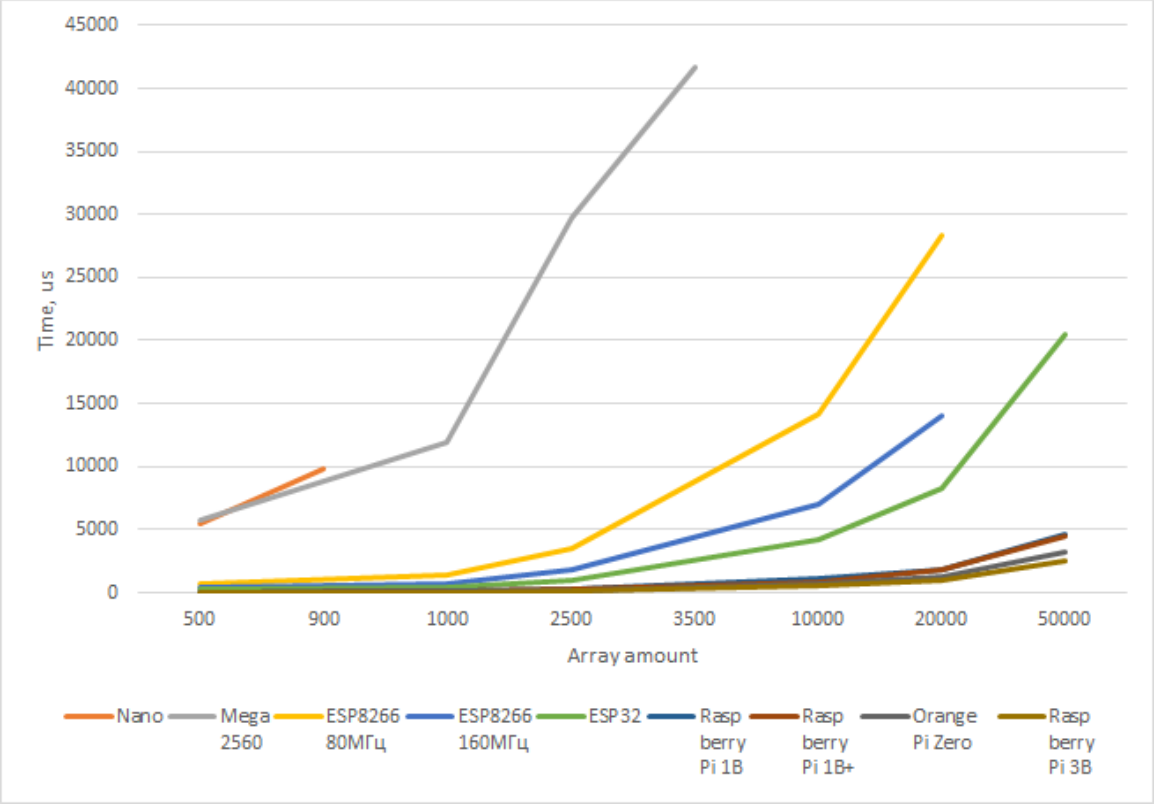

Рисунок 2.19 – Порівняння часу роботи усіх пристроїв з масивом

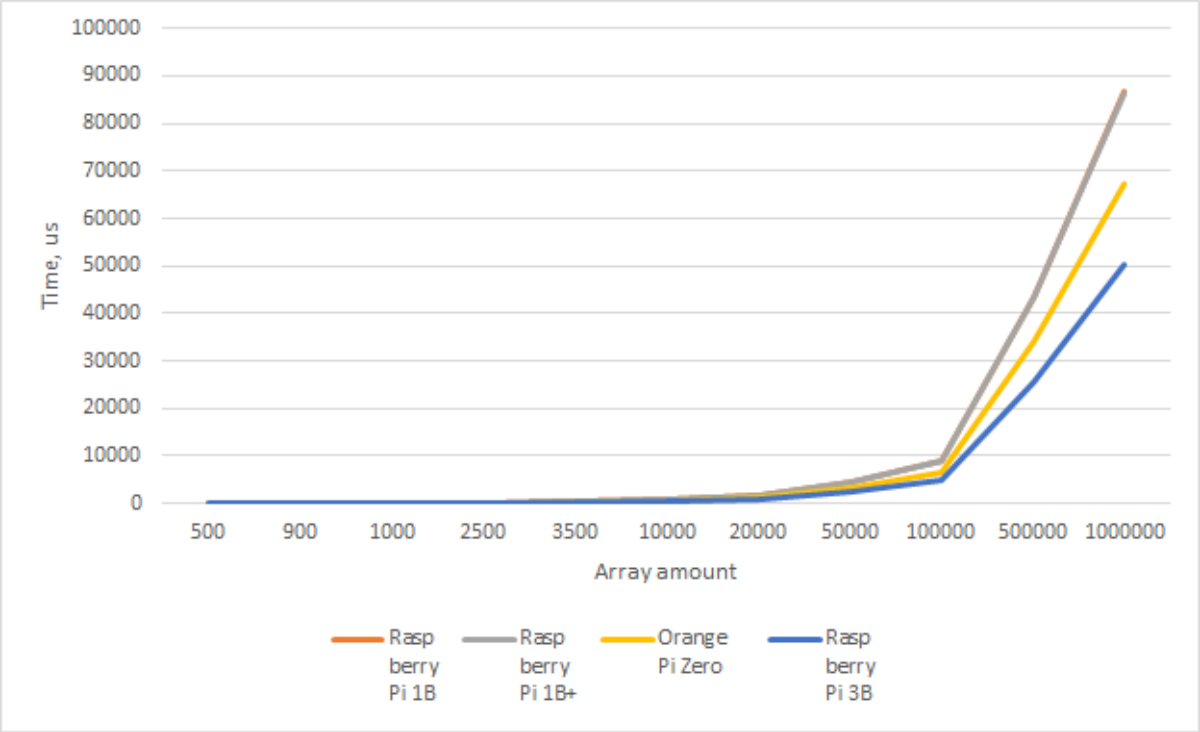

Рисунок 2.20 – Порівняння часу роботи міні ПК з масивом

Результати порівняння часу роботи з get запитом наведено у таблиці 2.34.

Таблиця 2.34 – Порівняння часу роботи з get запитом

Плата	**1 запит у середньому, мкс**
Nano	1242837.28
Mega 2560	1288304.08
ESP8266 80МГц	1015739.96
ESP8266 160МГц	1012743.62
ESP32	1036804.72
Raspberry Pi 1B	841.94
Raspberry Pi 1 B+	757.26
Orange Pi Zero	692.54
Raspberry Pi 3B	359

Діаграми порівняння часу роботи мікроконтролерів з get запитом наведено на рис. 2.21, міні ПК – рис. 2.22.

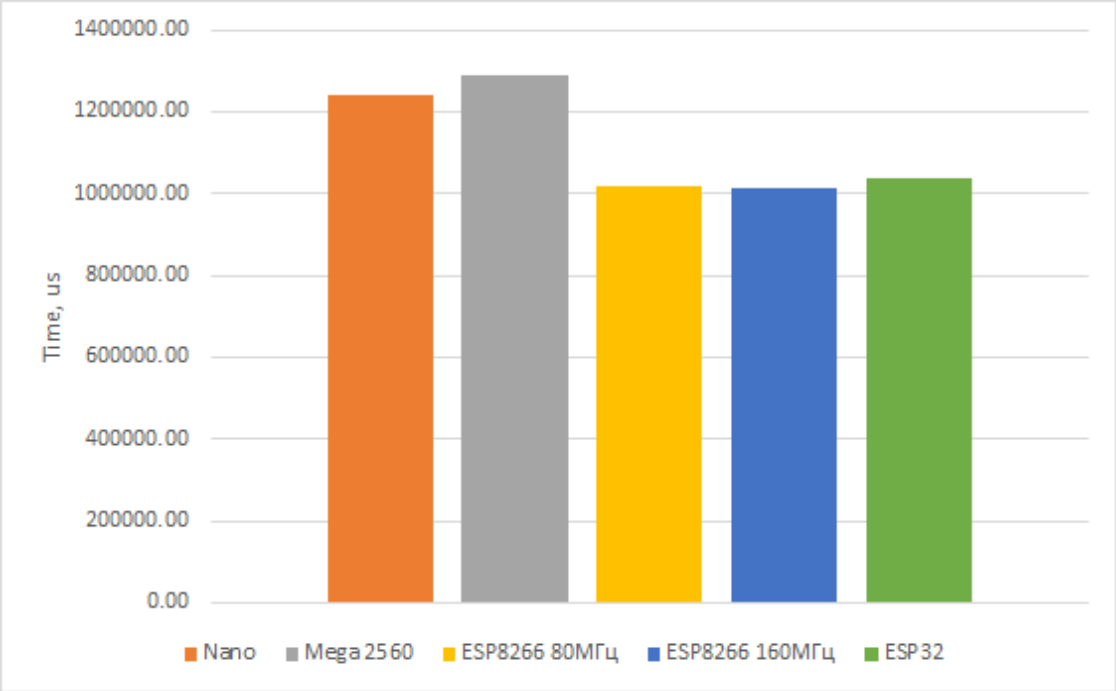

Рисунок 2.21 – Порівняння часу роботи мікроконтролерів з get запитом

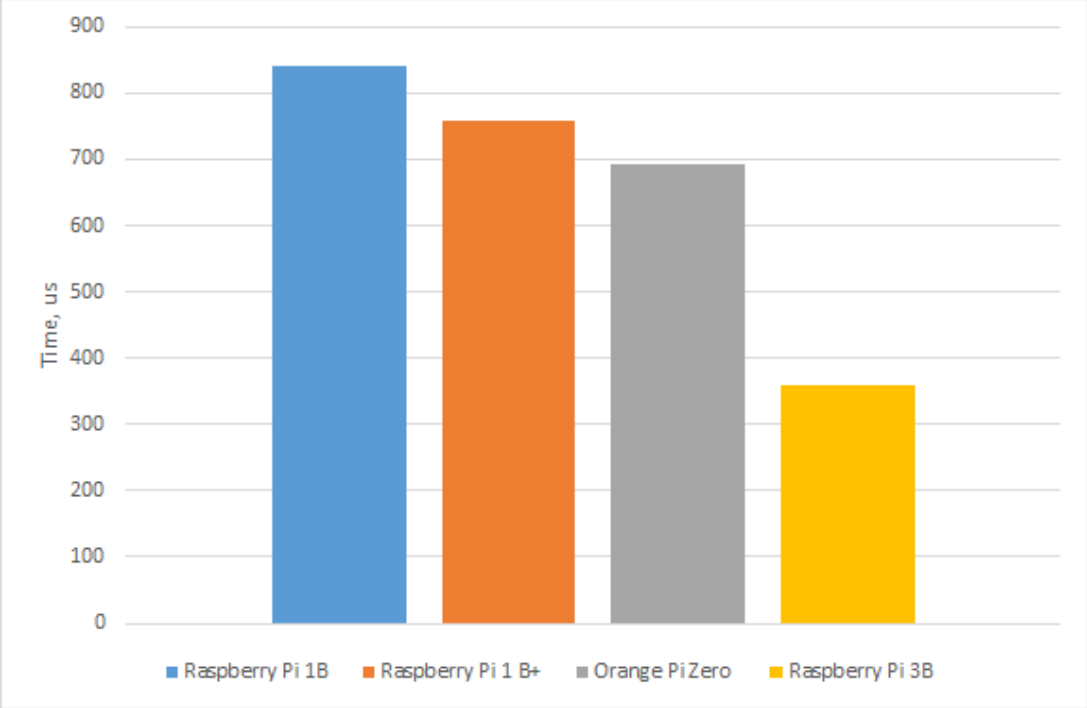

Рисунок 2.22 – Порівняння часу роботи міні ПК з get запитом

2.4.2 Мова Java

Аналогічно попередньому пункту порівняємо час для роботи з масивом та get запитів для міні ПК. У таблиці 2.35 наведено порівняння часу роботи з масивом.

Таблиця 2.35 – Порівняння часу роботи з масивом для міні ПК

Розмір масиву	**Raspberry Pi 1B**	**Raspberry Pi 1 B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
500	344	342.8	391.8	264
1000	689.4	668.4	793	584.2
2500	1752.8	1761.8	1927.6	740.4
10000	7290.4	7099.8	7815.8	2933.4
20000	14084	14222.2	15731	5303.8
50000	36863.6	36577	39161	27461.2
100000	73015	75572	78514.6	59604
500000	391183.6	382757.6	394620.6	293536.4
1000000	1041749.4	771385.2	789994	571169.8

Графік порівняння часу роботи міні ПК з масивом наведено на рис. 2.23

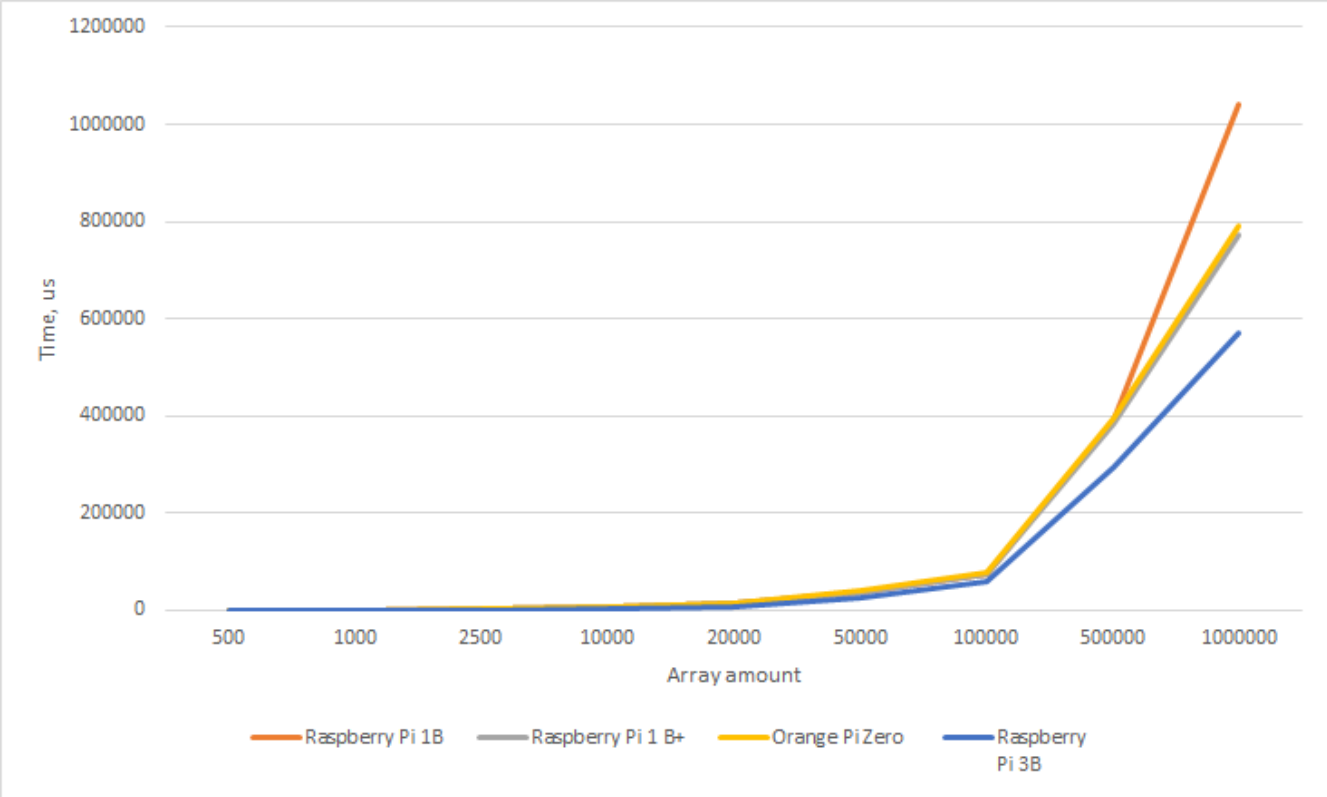

Рисунок 2.23 – Порівняння часу роботи міні ПК з масивом

Результати порівняння часу роботи з get запитом наведено у таблиці 2.36.

Таблиця 2.36 – Порівняння часу роботи з get запитом

	Raspberry Pi 1B	**Raspberry Pi 1 B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
Середній час на 1 запит, мкс	3227165.02	3147122.58	2291740.12	1759411.84

Діаграму орівняння часу роботи міні ПК з get запитом наведено на рис. 2.24

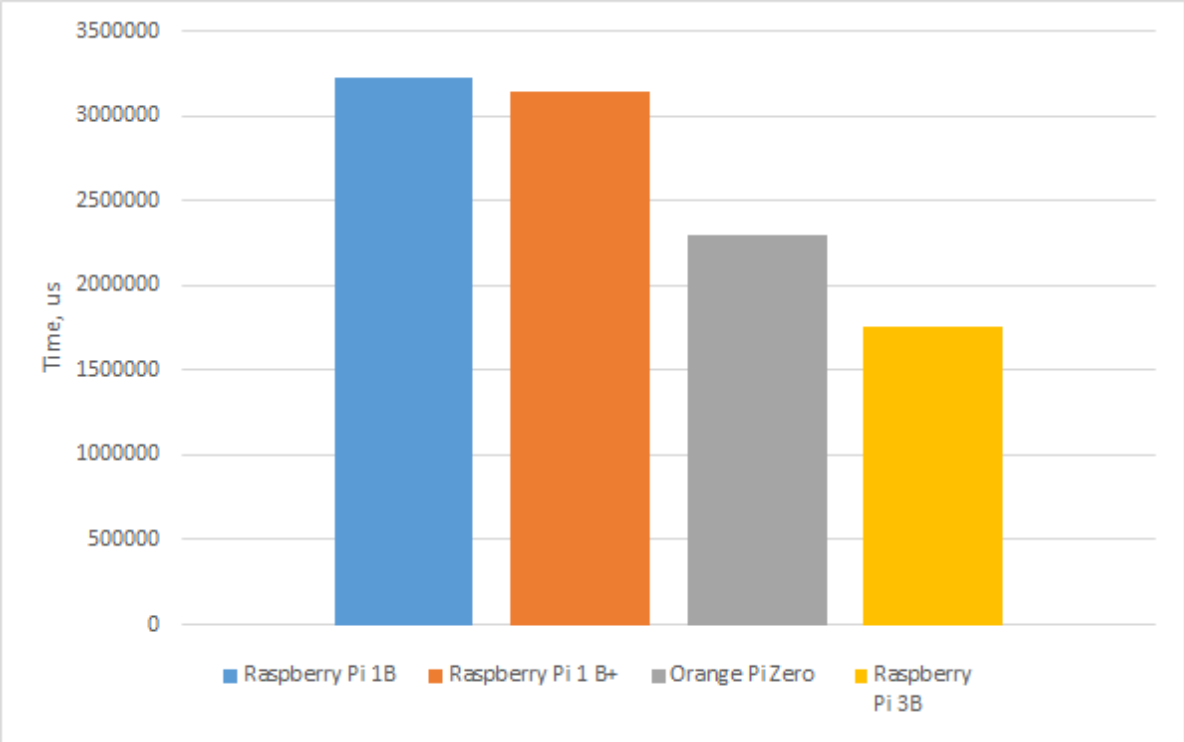

Рисунок 2.24 – Порівняння часу роботи міні ПК з get запитом

2.4.3 Мова Python

Аналогічно попередньому пункту порівняємо час для роботи з масивом та get запитів для міні ПК. У таблиці 2.37 наведено порівняння часу роботи з масивом.

Таблиця 2.37 – Порівняння часу роботи з масивом для міні ПК

Розмір масиву	Raspberry Pi 1B	Raspberry Pi 1 B+	Orange Pi Zero	Raspberry Pi 3B
500	820.6	824.8	1001.6	415.8
1000	1624.4	1584	2195.6	677.2
2500	3921.6	4006.8	5410	1907.8
10000	17609.8	17328.6	20374.4	8527.2
20000	52824.8	51127.8	42268.6	18010.4
50000	97489	88027.8	110207.6	44538.2
100000	196022	185470.6	214575.8	72537
500000	978975	977950.6	1165388	280681.4
1000000	1879198	1799412.4	2169943	780590.8

Графік порівняння часу роботи міні ПК з масивом наведено на рис. 2.25

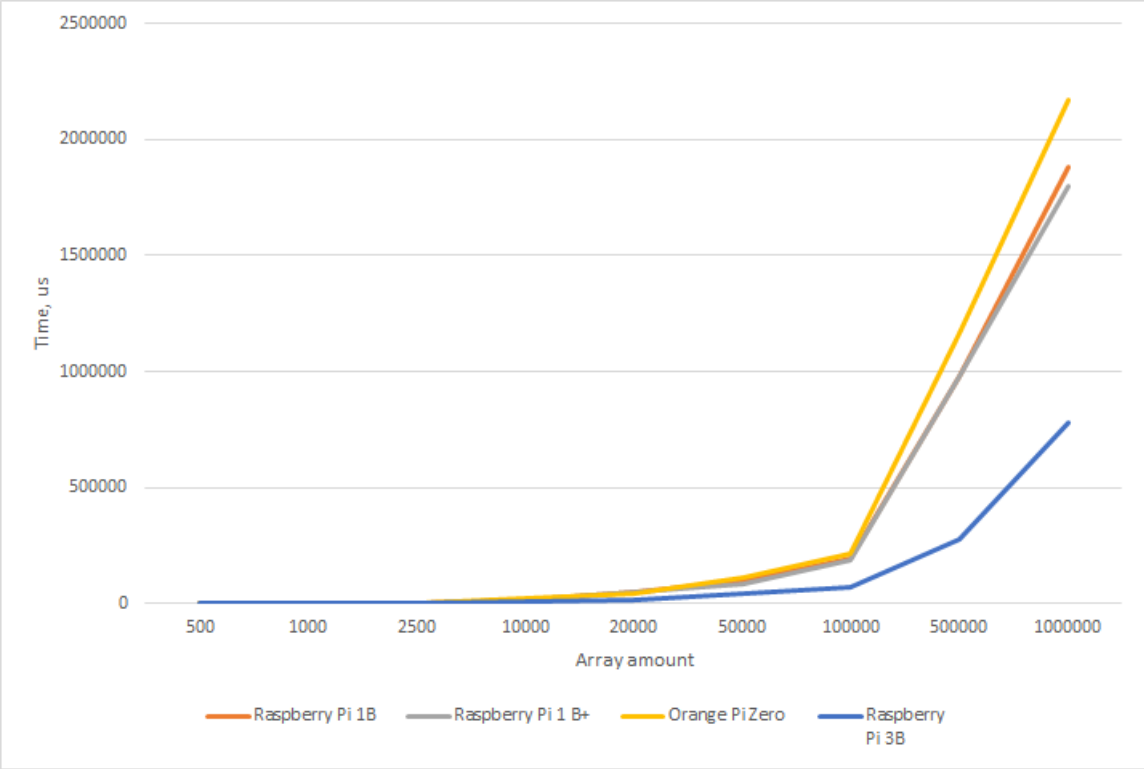

Рисунок 2.25 – Порівняння часу роботи міні ПК з масивом

Результати порівняння часу роботи з get запитом наведено у таблиці 2.38.

Таблиця 2.38 – Порівняння часу роботи з get запитом

	Raspberry Pi 1B	**Raspberry Pi 1 B+**	**Orange Pi Zero**	**Raspberry Pi 3B**
Середній час на 1 запит, мкс	11279360.5	10678131.4	6725039.54	10656496.4

Діаграму порівняння часу роботи міні ПК з get запитом наведено на рис. 2.26

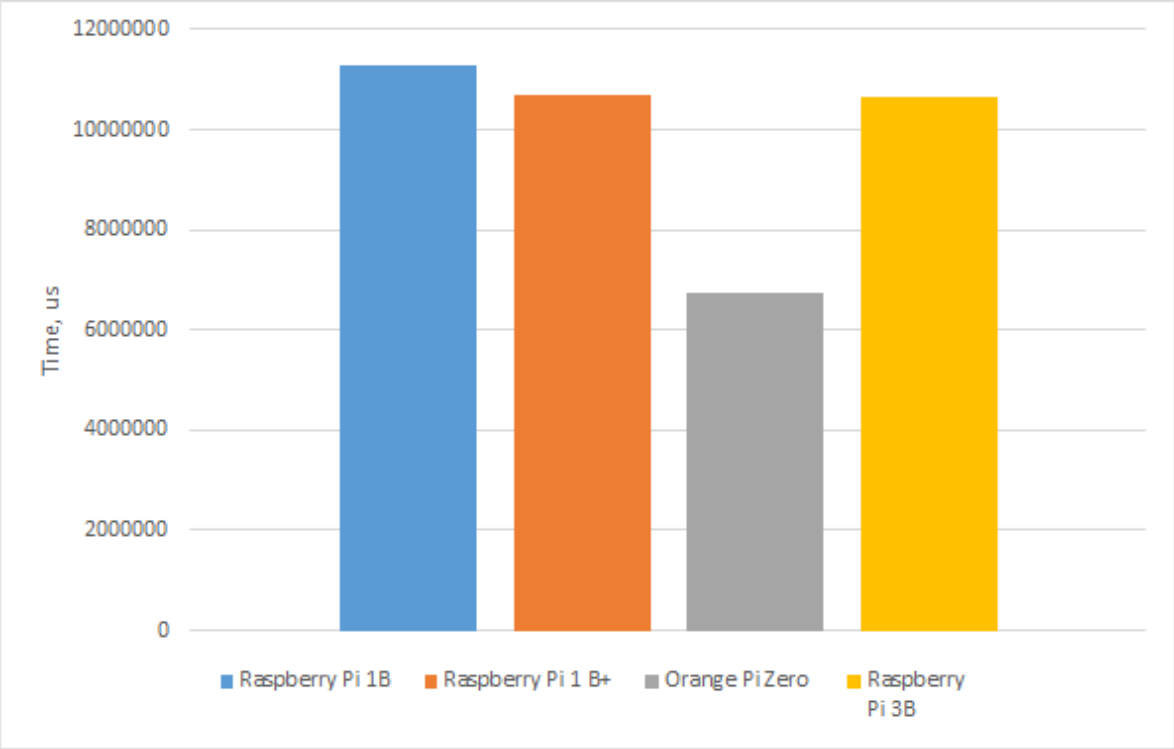

Рисунок 2.26 – Порівняння часу роботи міні ПК з get запитом

2.4.4 Висновки

Як показали тестування, рішення від Arduino мають найменшу ефективність роботи, до того ж їх ціна значно вище ніж у модолей ESP, а також вони не мають базової реалізації підключення до мережі, що робить їх використання недоцільним для IoT пристроїв. ESP32 на відміну від ESP8266 має більшу функціональність, але в нього присутні деякі проблеми з фреймворком, наприклад в Arduino фреймворку відсутня підтримка вбудованої пам’яті, та й ціна поки що на них вища ніж на ESP8266. Таким чином можна прийти до висновку, що серед мікроконтролерів ідеальним варіантом є ESP8266, адже він має вбудований Wi-Fi, внутрішню пам’ять, досить стабільний фреймворк, та помірну ціну, а також компактні розміри.

Серед міні ПК Raspberry Pi B та B+ показали найнижчі результати, так як Raspberry Pi zero, який побудований на тій самій платформі, то очевидно що його результати будуть схожі. Їх цілком достатньо для простого пристрою, але його

собівартість та складність налаштування робить його використання недоцільним ніяк hub пристрою, ні як звичайного пристрою. Оптимальним же для hub пристрою може бути або рішення від Orange Pi або Neo Pi, вони мають близькі характеристики до Raspberry Pi 3B, а але нижчу ціну або вбудовану пам’ять, що робить їх використання більш прийнятним.

3 ДОСЛІДЖЕННЯ ПРОТОКОЛІВ ПЕРЕДАЧІ ДАНИХ У IOT

Взаємодіє між пристроями всередині мережі суттєво залежить від протоколів для комунікації, які використовуються, розглянемо деякі з них.

3.1 Socket

Socket зазвичай використовуються для взаємодії клієнта з сервером. Типова конфігурація системи це коли сервер розміщується на одній машині, а клієнти - на інших машинах. Клієнти підключаються до сервера, обмінюються інформацією, а потім відключити.

Socket має типовий потік подій. У моделі орієнтований на встановлення з'єднань клієнт-сервер, Socket на серверній стороні чекає запитів від клієнта. Для цього, сервер спочатку налаштовую персональну адресу, яку клієнти можуть використовувати для пошуку сервера. Коли встановлено адресу, сервер чекає запитів від клієнтів. Обмін даними між клієнтом і сервером відбувається, коли клієнт підключається до сервера через Socket. Сервер виконує запит клієнта і відправляє відповідь назад клієнту. [14] Схема роботи наведено на рис 3.1.

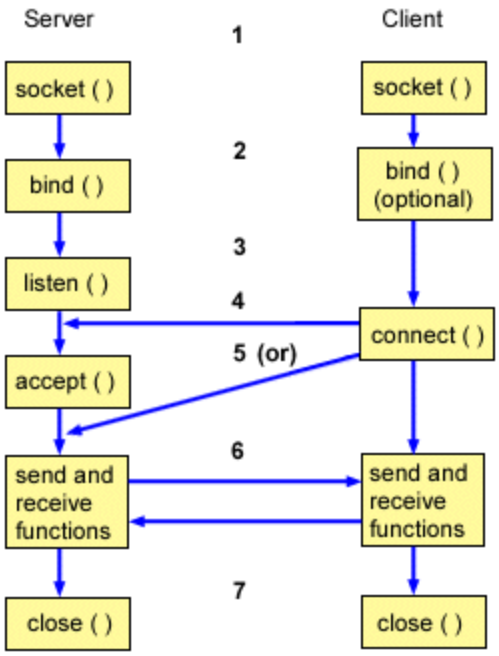

Рисунок 3.1 – Схема роботи Socket

3.2 REST

REST (передача репрезентативного стану) — це тип архітектури мережевих протоколів, що забезпечують доступ до інформаційних ресурсів. У 2000 році його описав та популяризовав один з засновників HTTP, Рой Філдінг. В основу REST закладено принципи функціонування Всесвітньої павутини і, зокрема, можливості HTTP протоколу.[15]

У протоколів REST дані мають передаватися у вигляді невеликої кількості стандартних форматів, таких як HTML, XML, JSON. REST протокол (як і HTTP) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформацію про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабування системи і дозволяє їй еволюціонувати з новими вимогами [16]. Приклад роботи наведено на рис. 3.2.

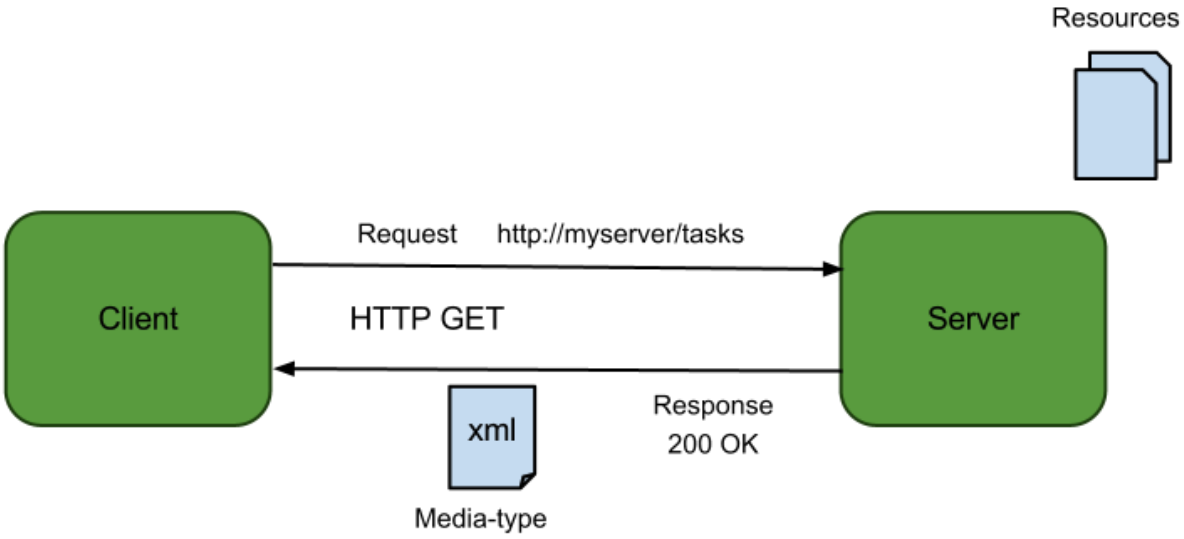

Рисунок 3.2 – Схема роботи REST [17]

REST пропонує розробникам використовувати HTTP-методи явно відповідно до визначення протоколу. Цей основний принцип проектування REST встановлює однозначну відповідність між операціями створення, читання, оновлення та видалення (CRUD) і HTTP-методами. Згідно з цим відповідності:

- Для створення ресурсу на сервері використовується POST.
- Для вилучення ресурсу використовується GET.
- Для зміни стану ресурсу або його поновлення використовується PUT.
- Для видалення ресурсу використовується DELETE.

3.3 MQTT

MQTT - це простий відкритий протокол, який був розроблений спеціально для застосування в IoT і обміну даними між пристроями. MQTT-мережа складається з MQTT-брокера, який виступає посередником у взаємодії MQTT-агентів - видавців і передплатників. Видавці публікують інформацію, призначену для передплатників. На рис. 3.3. наведено схему MQTT.

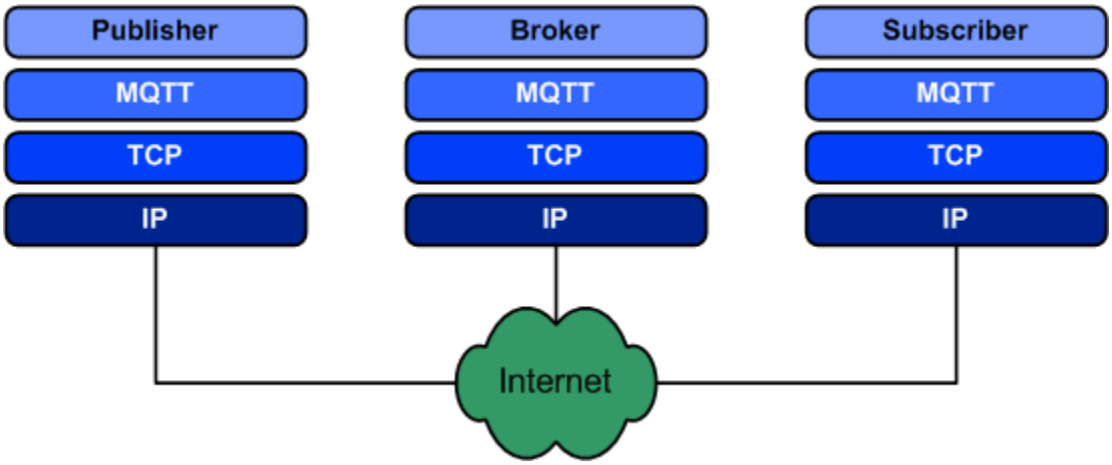

Рисунок 3.3 – Схема MQTT

MQTT працює за моделлю «видавець - передплатник», та використовує мінімальну кількість методів. Вони служать для вказівки дій, які необхідно виконувати. Ці дії зводяться до комунікації з брокером і до роботи з темами і повідомленнями. Агенти підключаються до брокера, а потім або публікують теми і повідомлення в них, або підписуються на теми і отримують повідомлення, в цих темах опубліковані. Завершивши роботу, агент відключається від брокера. Ось як виглядають методи MQTT:

- Connect - встановити з'єднання з брокером.
- Відключити - розірвати з'єднання з брокером.
- Опублікувати - опублікувати тему на брокера.
- Підписка - підписатися на тему на брокера.
- Відмовитися від підписки - відписатися від теми на брокера.

Спрощена схема взаємодії між передплатником і видавцем з використанням MQTT-брокера наведена на рис. 3.4.

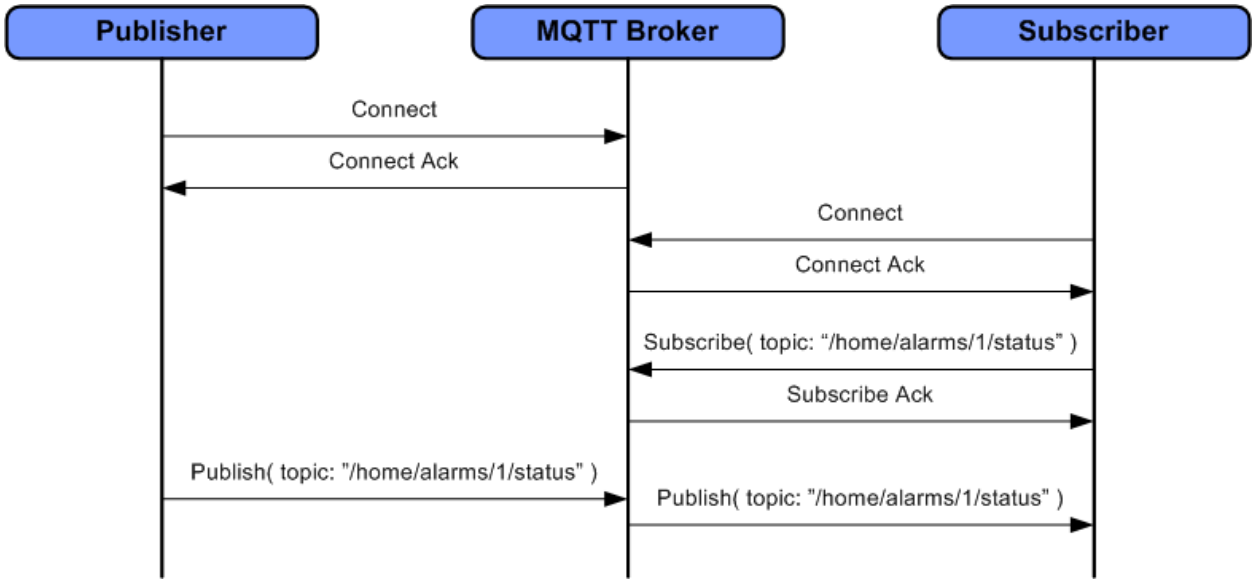

Рисунок 3.4 – Схема роботи MQTT

MQTT підтримує вказівку рівня якості (QoS Обслуговування). А саме, існують три таких рівні:

QoS 0. Цей рівень задіє стратегію «максимум одноразова доставка повідомлень». Приймач повідомлення не підтверджує їх отримання, відправник, відповідно, передає повідомлення лише раз, не роблячи спроб по їх повторної передачі. Це - метод «відправив і забув».

QoS 1. Тут застосовується підхід «мінімум одноразова доставка повідомлень". Гарантується, що приймач отримає повідомлення хоча б один раз. При цьому передплатник може отримати одне й те саме повідомлення кілька разів. А відправник буде робити повторні спроби відправки до тих пір, поки не отримає підтвердження в успішній доставку повідомлення.

QoS 2. Цьому рівню якості обслуговування відповідає найповільніша процедура доставки повідомлень, але при цьому він - найнадійніший. Його основна особливість - реалізація стратегії «одноразова доставка повідомлень». При його використанні застосовується чотириступінчаста процедура підтвердження доставки повідомлень.

Вибір конкретного рівня якості обслуговування залежить від особливостей переданих даних і від того, наскільки важливо, щоб вони були доставлені. [18]

3.4 SNMP

SNMP - Simple Network Management Protocol, він же Простий Протокол Мережевого Управління. Протокол створений в 1988 році з метою управління великою кількістю мережевих пристроїв. З того моменту протокол набрав відповідну популярність і став стандартом. З моменту розробки протокол SNMP був 3 рази перероблений з версії SNMPv1, SNMPv2 і SNMPv3.

Крім управління пристроями, дуже часто SNMP використовують для моніторингу. SNMP може отримувати різну інформацію від будь-яких мережевих пристроїв, будь то роутер, свіч або просто комп'ютер (в яких є підтримка даного протоколу (читай - запущений агент SNMP). Вміст одержуваної інформації може бути дуже різноманітним, наприклад: час аптайма, різні лічильники продуктивності CPU , мережі та ін., мережеві параметри пристроїв ...

Мережа, що використовує SNMP для управління містить три основні компоненти:

- SNMP менеджер - ПО, яке встановлюється на ПК адміністратора (системи моніторингу)
- SNMP агент - ПО, запущене на мережевому вузлі, за якими здійснюється моніторинг.
- SNMP MIB - MIB це Management information base. Цей компонент системи забезпечує структурування даних, якими обмінюються агенти і менеджери. По суті - це якась база даних у вигляді текстових фалів.

Схему взаємодії SNMP-агент - SNMP-менеджер наведено на рис. 3.5. [19]

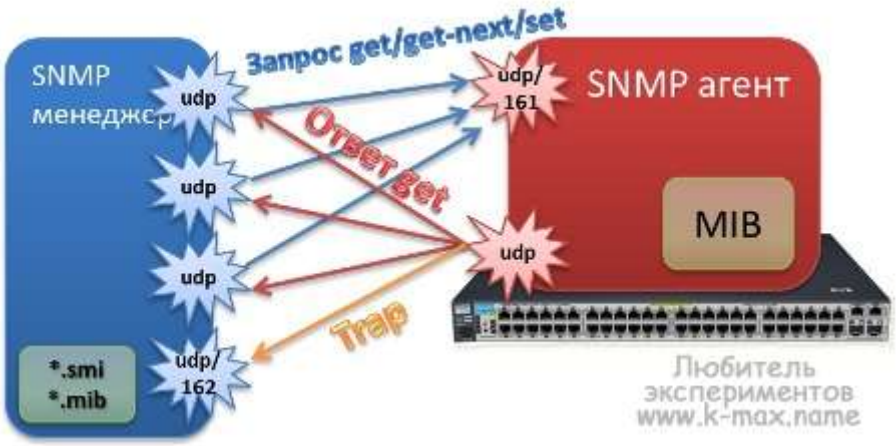

Рисунок 3.5 – Схема взаємодії SNMP-агент - SNMP-менеджер

4 ПАРАЛЕЛЬНІ ОБЧИСЛЕННЯ НА IOT ПРИСТРОЯХ

З метою покращення ефективності роботи IoT пристроїв можливе використання паралельних обчислень, за умови апаратної підтримки пристроями таких рішень. Тобто бажаним для такого є наявність декількох ядер. Ми дослідимо можливість використання потоків та роботу з Docker на міні ПК.

4.1 Багато поточність

Для міні комп'ютерів на основі Linux можна скористатися стандартними засобами для роботи з потоками різних мов програмування. Зазвичай є один основний потік з якого власне запускаються додаткові, схема наведена на рис. 4.1.

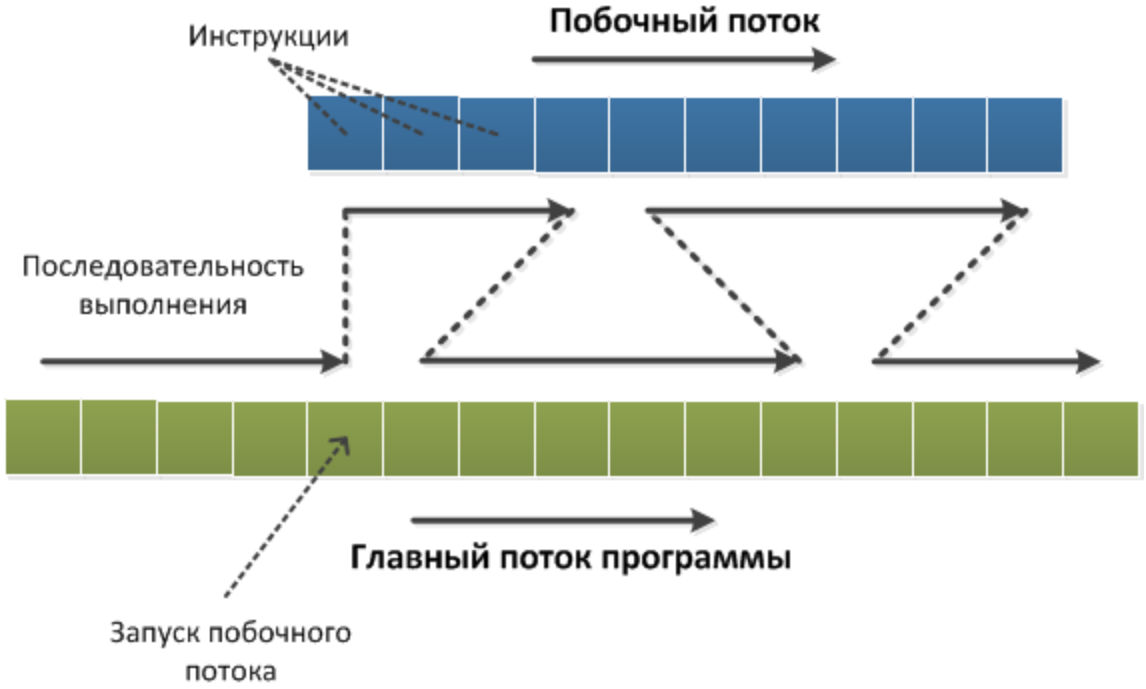

Рисунок 4.1 – Схема роботи потоків [20]

Загалом робота потоків на різних мовах програмування залежить від особливостей реалізації і вимагає додаткового вивчення специфікації цих мов.

Для дослідження скористаємось міні ПК Orange Pi Zero та Raspberry Pi 3B, які мають по 4 ядра. Розглянемо можливості роботи з потоками на мовах C++ та Java. Для тестування скористаємося тестами для роботи з масивами, де додаються усі елементи масиву, які ми використовували у розділі 2. Модифікуємо програми, щоб вони могли виконувати обчислення у різних потоках. Виконаємо тести для 2 та 4 потоків.

4.1.1 Мова C++

Для C++ скористаємося бібліотекою thread, компілятором g++ та прапорами std=c++11 та pthread. Результати наведено у таблиці 4.1.

Таблиця 4.1 – Порівняння часу роботи з масивом для міні ПК на мові C++

Розмір масиву	Потоки					
	1 Orange Pi Zero	1 Raspberry Pi 3B	2 Orange Pi Zero	2 Raspberry Pi 3B	4 Orange Pi Zero	4 Raspberry Pi 3B
500	38.4	29	266.8	296.8	400.8	444.2
1000	71.6	54.6	286.4	293.4	439.4	485
2500	169	131.2	261.8	390.2	494.6	555.4
10000	663.4	511.2	283.2	663.2	551.2	968
20000	1323.8	1009.4	326.8	1284.2	796	1600.4
50000	3287.2	2542.2	333.8	2838.2	814.8	3049.8
100000	6628	5072.6	397	5573.4	883.8	5689
500000	33962.8	25413	25457.2	25655	13425.6	25922
1000000	67339.6	50376	58609.6	50884	53692.2	51277.6

Графік порівняння часу виконання наведено на рис. 4.2.

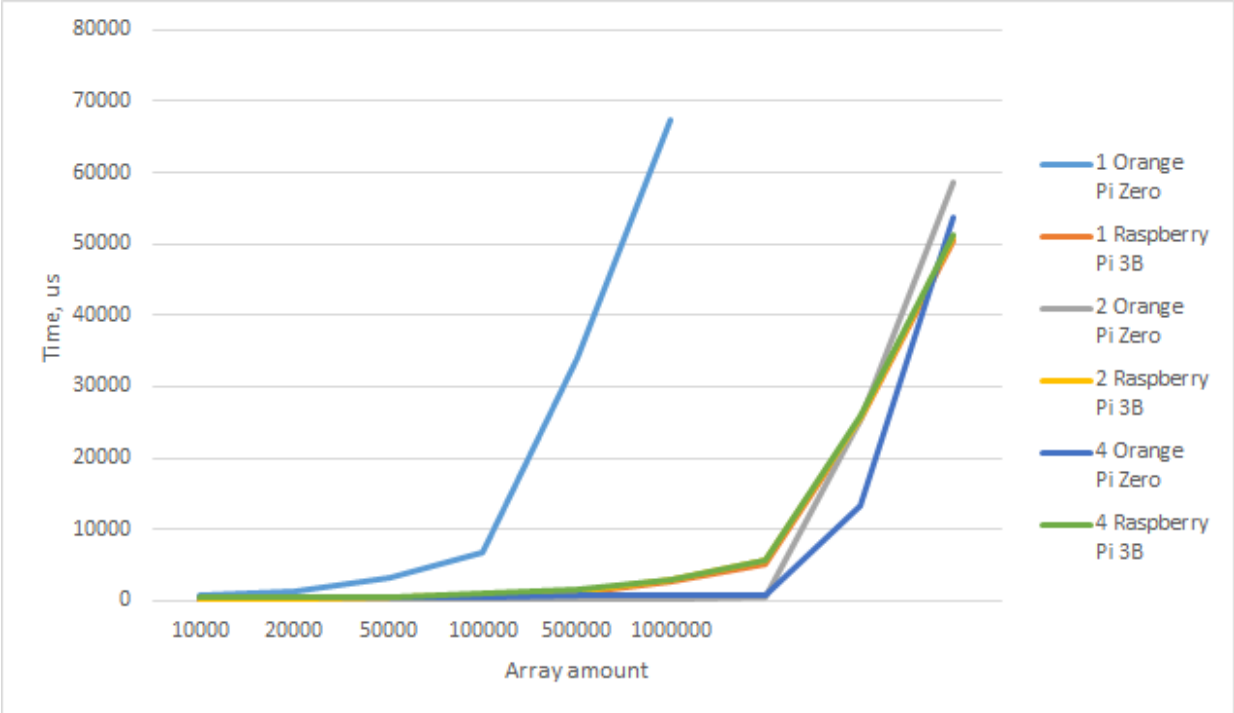

Рисунок 4.2 – Порівняння часу роботи з масивом для міні ПК на мові C++

Як можна помітити з наведених результатів, ефективність використання потоків з'являється для масивів в яких більше ніж 10000 елементів. Тут слід зауважити, що зростання ефективності відбулося лише на Orange Pi Zero, для Raspberry Pi 3B будь-який приріст відсутній. Такий результат може пояснюватися ти фактом, що для Orange Pi Zero використовувався образ Linux з більш новою версією ядра системи (4.11), у то час у Raspberry була 4.9. Orange Pi Zero виконав операціє додавання для елементів масиву с 1000000 елементів у 1.15 разів швидше на 2 потоках та у 1.25 – на 4 потоках.

4.1.2 Мова Java

Для тестування будемо використовувати Java 1.8 JRE від openjdk. Для реалізації паралельності будемо спадкувати обчислювальний клас від класу Thread. Результати тестування наведено у таблиці 4.2. Графік порівняння швидкості виконання наведено на рис. 4.3.

Таблиця 4.2 – Порівняння часу роботи з масивом для міні ПК на мові Java

Розмір масиву	Потоки					
	1 Orange Pi Zero	1 Raspberry Pi 3B	2 Orange Pi Zero	2 Raspberry Pi 3B	4 Orange Pi Zero	4 Raspberry Pi 3B
500	391.8	264	2190.8	990.2	3046	988
1000	793	584.2	2218.8	1719.6	3270	1682
2500	1927.6	740.4	2812.6	867.4	3366	1110
10000	7815.8	2933.4	6092.2	2733	5275	2287
20000	15731	5303.8	10272.2	3796	7448	3674
50000	39161	27461.2	22075.6	18110.6	13412	12176
100000	78514.6	59604	41919.6	39343.8	23258	24116
500000	394620.6	293536.4	200208.4	168527.8	102780	126631
1000000	789994	571169.8	402613.6	307087.4	202213	162173

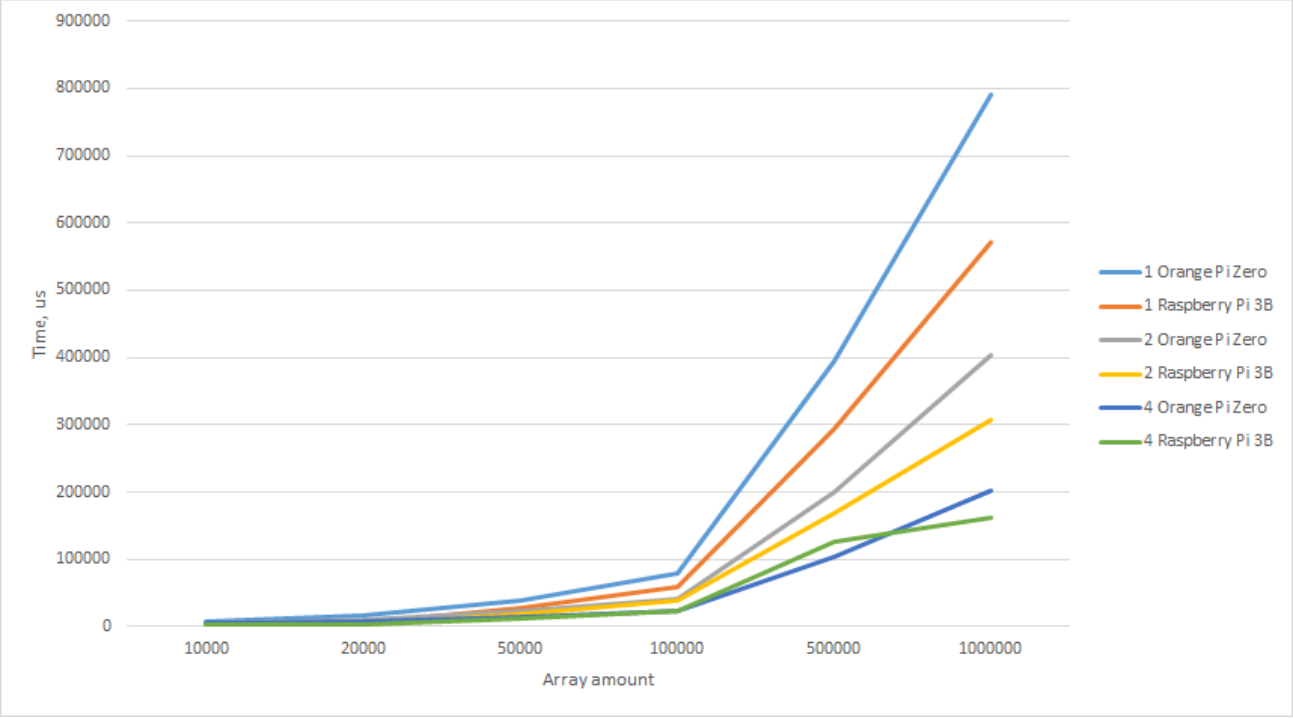

Рисунок 4.3 – Порівняння часу роботи з масивом для міні ПК на мові Java

4.2 Docker контейнери

Для тестування роботу з Docker встановимо на Raspberry Pi останню версію Raspbian ядром 4.4 [21] з оновленням до 4.9, а на Orange Pi Zero – armbian з версією ядра 4.11 [22].

4.2.1 Встановлення Docker та Docker Compose

Для встановлення Docker скористаємося інструкціє з офіційного сайту Raspberry [23]. Для встановлення Docker Compose скористаємося інструкцією [24]. В результаті отримаємо Docker версії 17.05.0-ce та Docker Compose версії 1.13.0.

Встановити Docker та Docker Compose вдалося на все 4 тестові міні ПК:

- Raspberry Pi B
- Raspberry Pi B+
- Orange Pi Zero
- Raspberry Pi 3B

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 побудову локальної системи для IoT пристроїв.

4.2.2 Docker swarm

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.

Для реалізації поставленої задачі скористаємося 4 міні ПК, на які ми встановили Docker у попередньому пункті. Для підключення використаємо наступне обладнання:

- Роутер Asus RT-N18U
- Комутатор TP-LINK TL-SF1008D
- 4 міні ПК

Роутер буде відповідати за підключення до мережі інтернет, комутатор за комутацію чотирьох міні ПК, ну а міні ПК власне і будуть вузлами нашого кластера. Топологію побудованої мережу, наведено на рис. 4.4.

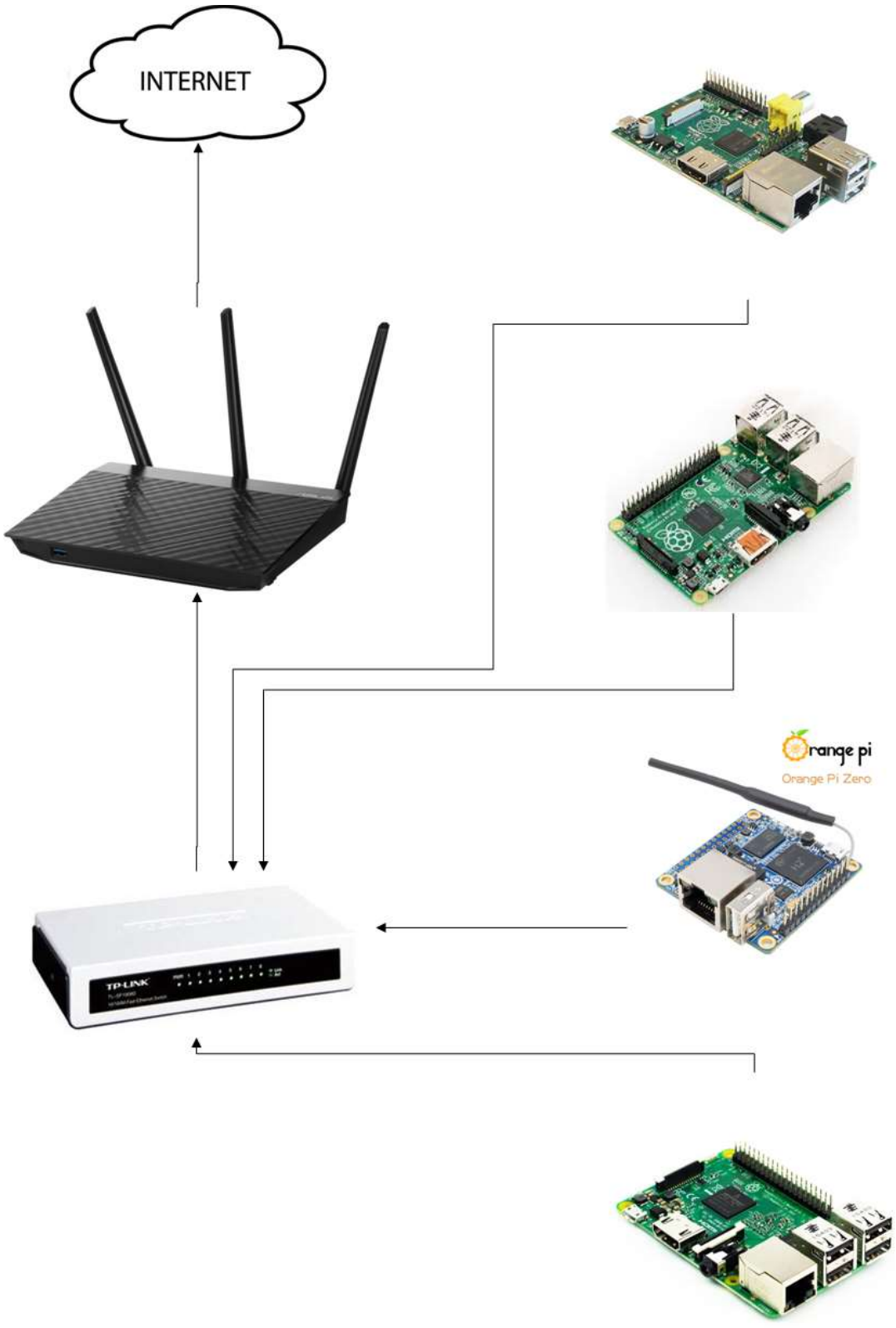

Рисунок 4.4 – топологія мережі для розгортання Docker swarm

Далі нам необхідно налаштувати власне Docker swarm. Для цього скористаємося інструкцією [25]. Головним вузлом зробимо Raspberry Pi 3B, резервним – Orange Pi Zero, а Raspberry Pi B та Raspberry Pi B+ - звичайними вузлами. На рис. 4.5 зображено результат команди відображення вузлів, як можна побачити все налаштовано так, як і планувалося.

Рисунок 4.5 – 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 рис. 4.6.

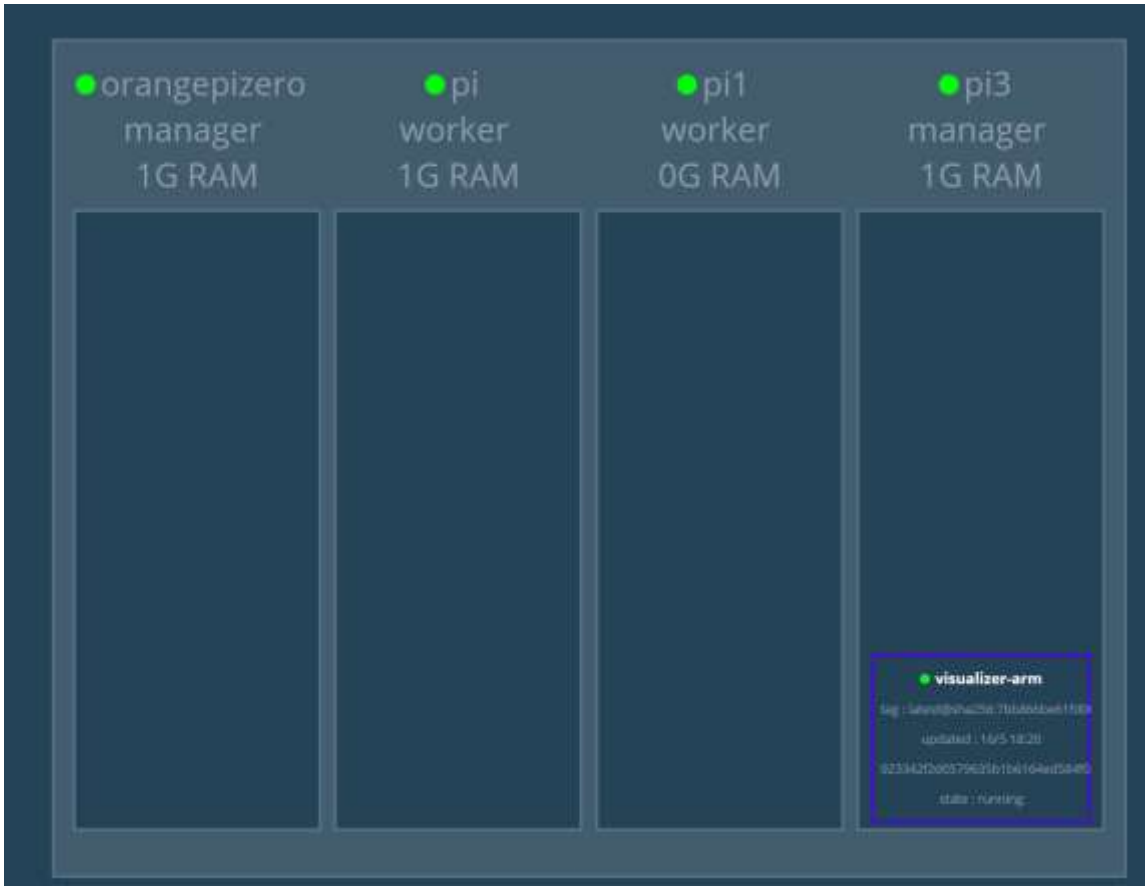

Рисунок 4.6 – Веб інтерфейс сервісу візуалізації

Для тестування роботи запустимо у нашому кластері контейнер, який відображає веб сторінку та виконаємо його масштабування до 10 екземплярів. Після цього ми маємо 10 контейнерів, доступ до яких відбувається за кожним з 4 IP адресів міні ПК. Далі за допомогою балансерів можна реалізувати розподілення навантаження на всі вузли. Візуалізацію масштабування 10 веб контейнерів наведено на рис. 4.7.

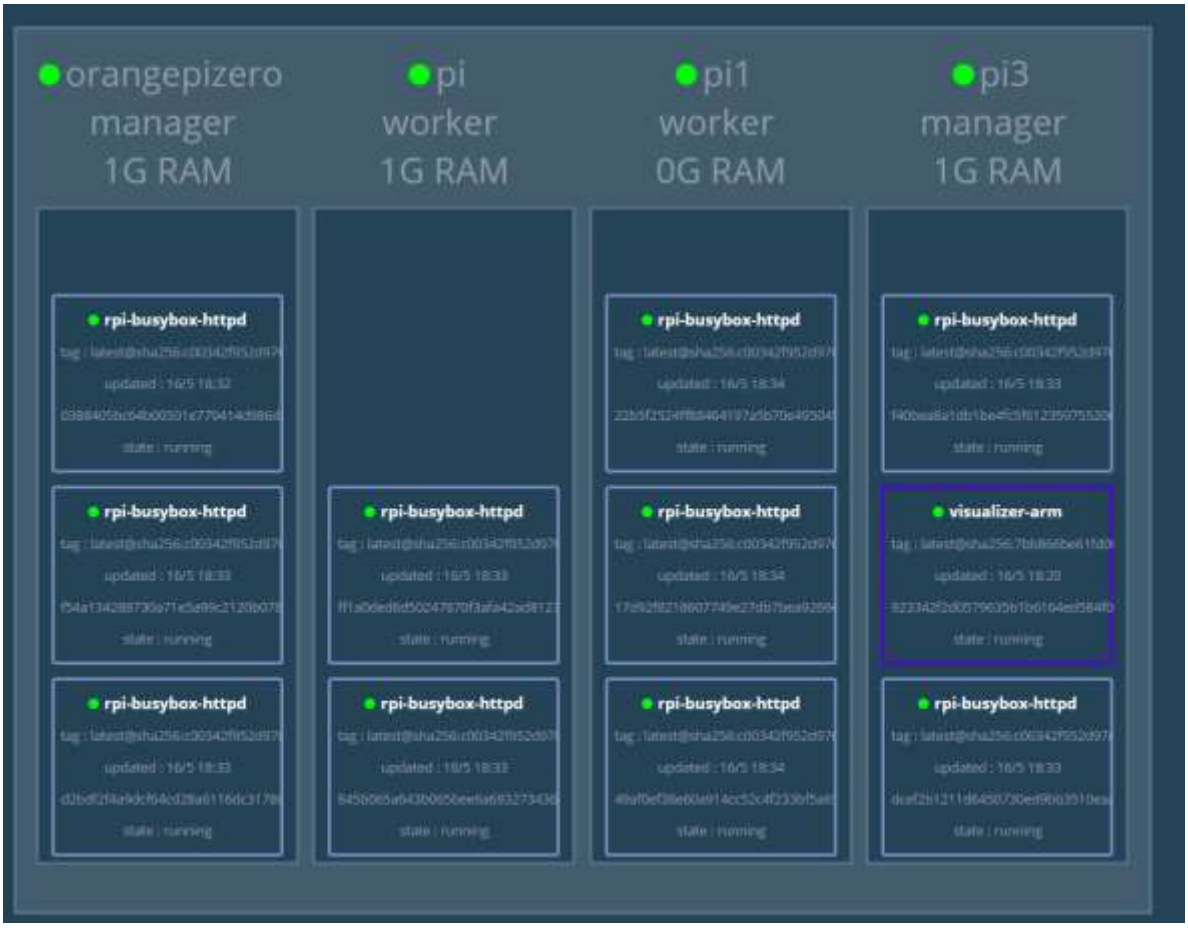

Рисунок 4.7 – Візуалізація масштабування веб контейнерів

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, адже у розглянутому у прикладі вони були абсолютно різними, що негативно впливало на роботу системи.

5 ЗАСОБИ ТА МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ В IOT ПРИСТРОЯХ

Виконати захист інформації можна різними методами. Можна кодувати та декодувати повідомлення на пристроях IoT, але тут ми можемо зіткнутися з такою проблемою як недостатня швидкодія пристроїв, здебільшого мікроконтролерів. Інший варіант це покласти усі питання надійності та захисту передачі інформації на комп'ютерну мережу. В такому випадку ми вважаємо що мережа в якій знаходяться пристрої є досить надійною та ізольованою. Для підключення до віддалених частин мережі можливо використовувати VPN або SSH тунелі.

5.1 Захист Wi-Fi

Wi-Fi це один з найпоширеніших видів бездротового підключення до локальної мережі. Сьогодні він використовується майже усюди, тому його використання є найпростішим варіантом побудови IoT системи. В загалі, Wi-Fi є досить безпечний, якщо вимкнути усі проблемні його режими. Такими режимами є WEP шифрування там QSS. Обидва ці технології дозволяють без проблем отримати доступ до мережі і бажано їх не використовувати.

Найпоширенішим варіантом шифрування на даний момент є WPA2-PSK, який використовує AES шифрування. Злом точок доступу розглянуто у статті [26]. Автор вказує, що для злому можна використовувати лише повний перебір можливих паролів і такого перебору словника на 250 мільйони паролів, розміром 2 Гб, на ноутбуку, буде необхідно витратити близько 66 годин. У той же час, словник на повних комбінацій 9 знакового паролю буде містити 900 мільйонів позицій і для його повного перебору необхідно декілька тижнів.

5.2 Обмеження доступу у протоколах передачі

Більшість протоколів для передачі даних підтримують аутентифкацію. Дуже часто для спрощення роботи її не використовують, але даремно, адже кожен з елементів безпеки може зробити більш складним життя хакера.

Отже, протоколи MQTT, REST, SNMP підтримують аутентифікацію, що також можна використати для забезпечення захисту інформації, що передається, та керування пристроями.

5.3 Модель підключення через VPN/SSH

VPN та SSH – це найпоширеніші засоби побудови тунелів.

У випадку використання VPN необхідно налаштувати мережеве обладнання, щоб воно виконувало маршрутизацію відповідним чином, або ж це може виконувати сам міні ПК.

У випадку використання SSH тунелів дані передаються через цей мережевий протокол. Для його використання є необхідним наявність деякого пристрою на якому вони будуть запущені та вказано переадресацію з яких портів виконувати.

Такий чином, обидва варіанти чудово підходять при роботі з міні комп'ютерами на ОС Linux.

Для забезпечення безпеки SSH є безліч методів, варіант у якому обмежується кількість невдалих підключень наведено у статі [27]. Він дозволяє суттєво ускладнити життя хакерам, які намагаються виконати атаку перебором паролів.

Схему підключення віддалених об’єктів до центральної мережі наведено на рис. 5.1.

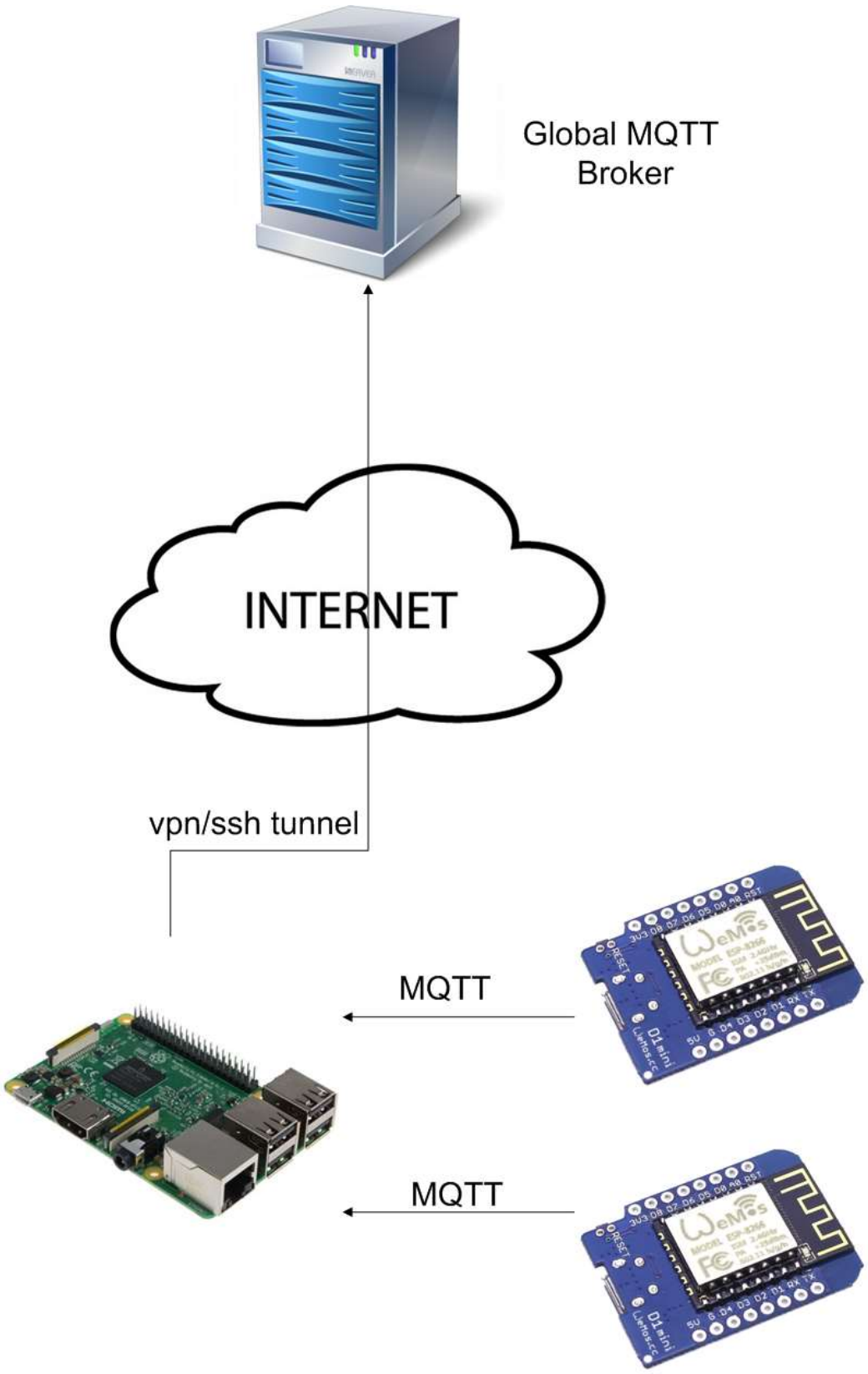

Рисунок 5.1 – Схема підключення віддалених об’єктів до центральної мережі

У схемі підключення на міні ПК налаштовується тунель через VPN або SSH до централізованої системи, а датчики на базі мікроконтролерів через нього підключаються до централізованого сервера або напряму через MQTT, або з використанням MQTT брокера.

У випадку використання MQTT брокера він відповідає за отримання повідомлень у своїй мережі та їх ретрансляцію до централізованого вузла.

5.4 Висновки

Таким чином, можна прийти до висновків, що доцільним є покладання питань безпеки на стандартні засоби протоколів взаємодії та мережевої передачі даних. Також слід керуватися правилом, що безпеки мало не буває і не використовувати прості та легкі паролі, а також включати режими аутентифікації всюди, де це можливо.

6 ПРИКЛАД РЕАЛІЗАЦІЇ

В якості демонстрації роботи пристроїв побудуємо невелику домашню систему, яка буде вимірювати температуру, вологість та освітленість у 2 приміщеннях, а також атмосферний тиск.

6.1 Концепція взаємодії

Для реалізації домашньої системи IoT пристроїв ми використаємо наступну концепцію:

- Датчики будуються на ESP8266 і підключаються до мережі по Wi-Fi
- В якості ПЗ для взаємодій використовується MQTT брокер
- В якості ПЗ для візуалізації використаємо Node-Red з доступом через веб інтерфейс
- MQTT брокер та Node-Red розгортаються як docker контейнери через docker compose на Orange Pi Zero
- Доступ до системи за межами локальної мережі відбувається за допомогою підключення через VPN

Схему тестової мережі наведено на рис. 6.1.

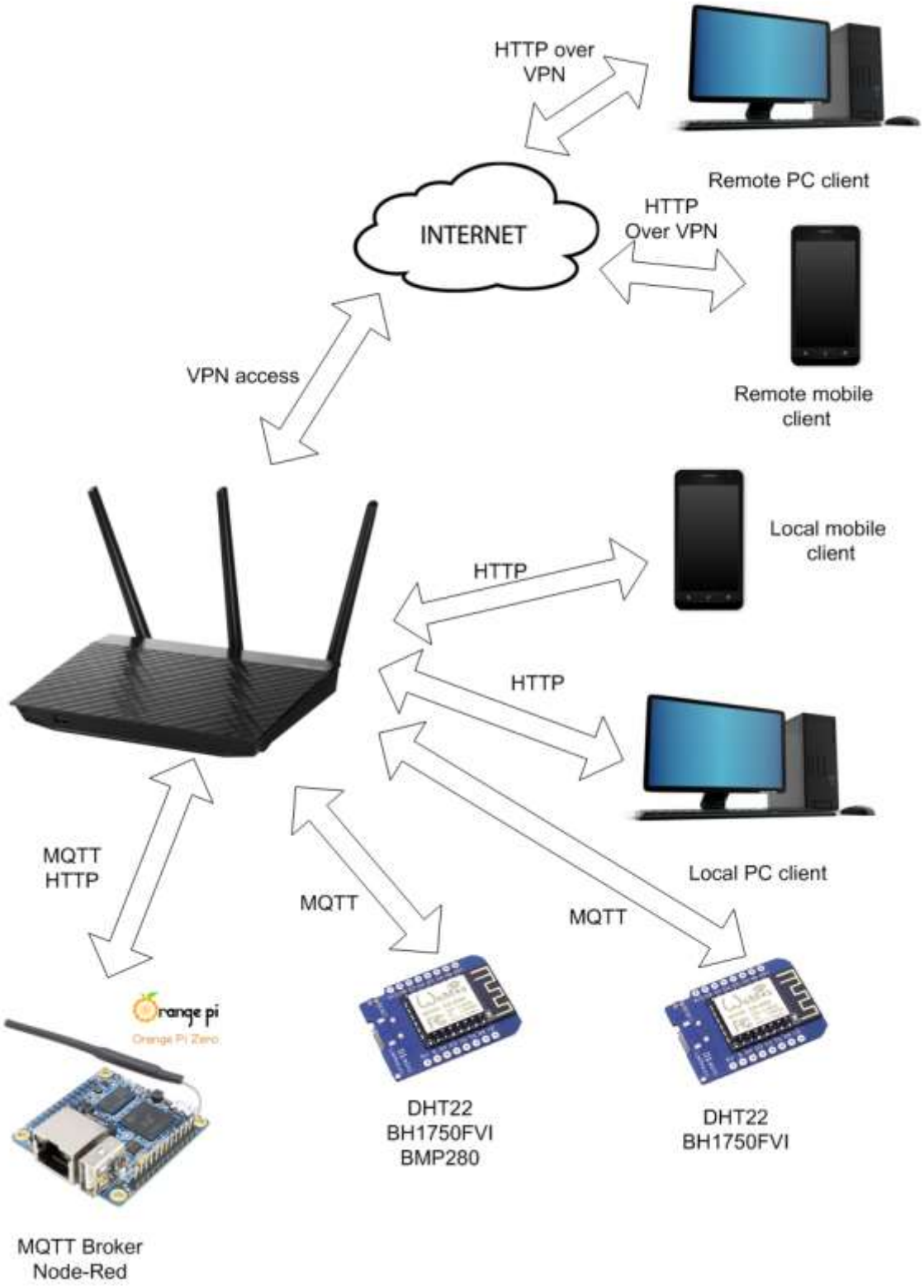

Рисунок 6.1 – Схема тестової домашньої IoT системи

Температуру та атмосферний тиск будемо вимірювати за допомогою датчиків DHT22, освітленість – BH1750FVI, а атмосферний тиск – BMP280.

Фотографію тестового датчика наведено на рис. 6.2.

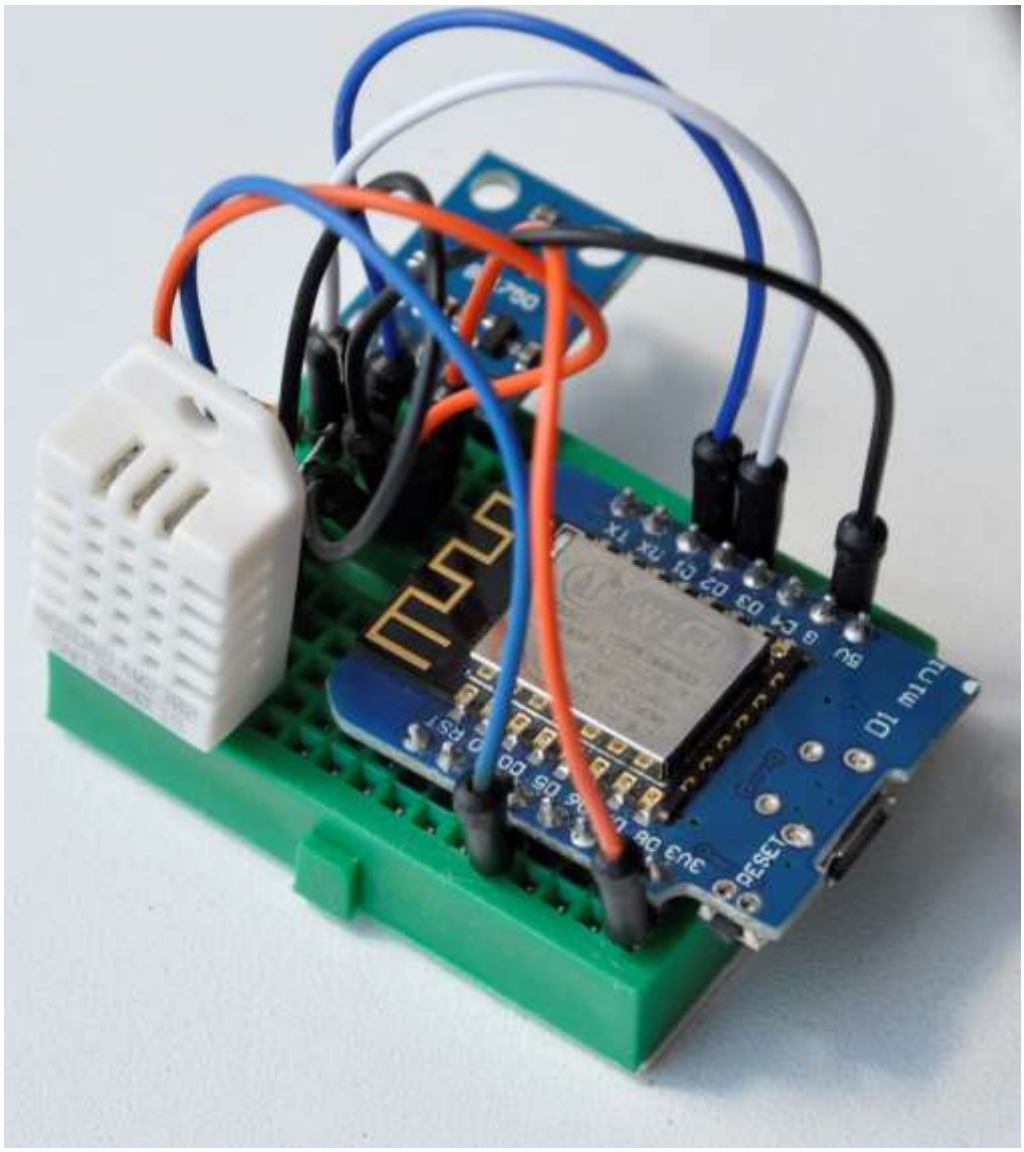

Рисунок 6.2 – Тестовий датчик

6.2 Результати роботи системи

Після усіх налаштувань та декількох днів роботи системи наведемо результати її роботи.

6.2.1 Використання ресурсів

Спочатку поглянемо на використання ресурсів, нас буде цікавити, чи справляється Orange Pi Zero зі своєю задачею роботи з Docker контейнерами. На рис. 6.3 зображено споживання ресурсів docker контейнерами у звичайному стані системи.

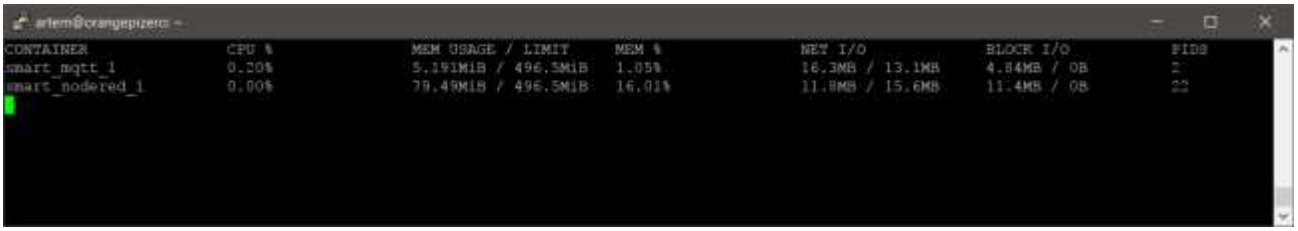

Рисунок 6.3 – Споживання ресурсів docker контейнерами у звичайному стані

На рис. 6.4 зображено споживання ресурсів docker контейнерами при запиті даних для графіків у Noder-Red.

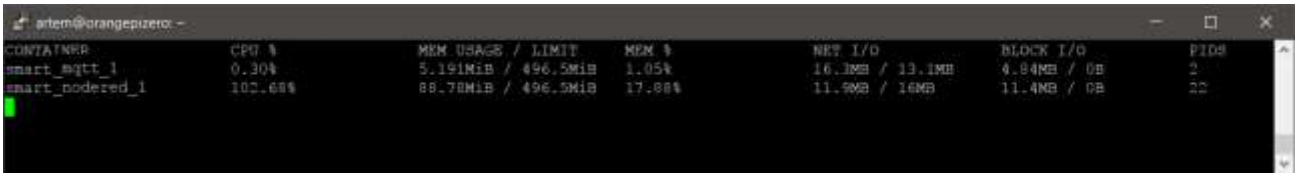

Рисунок 6.4 – Споживання ресурсів docker контейнерами при запиті

На рис. 6.5 зображено споживання ресурсів при запиті даних для графіків у Noder-Red у програмі htop.

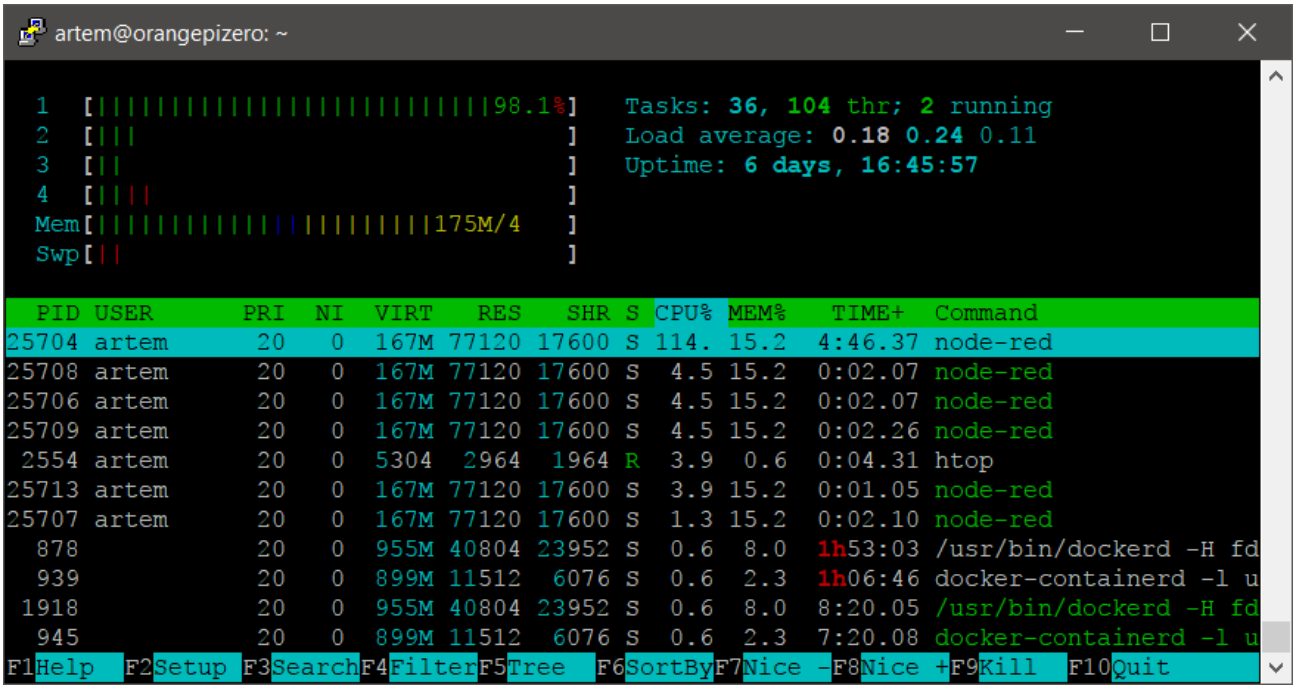

Рисунок 6.5 – Споживання при запиті

6.2.2 Візуалізація Node-Red

Як вже говорилося раніше, для візуалізації результатів роботи будемо використовувати Node-Red. На рис. 6.6 наведено приклад налаштування системи через веб інтерфейс.

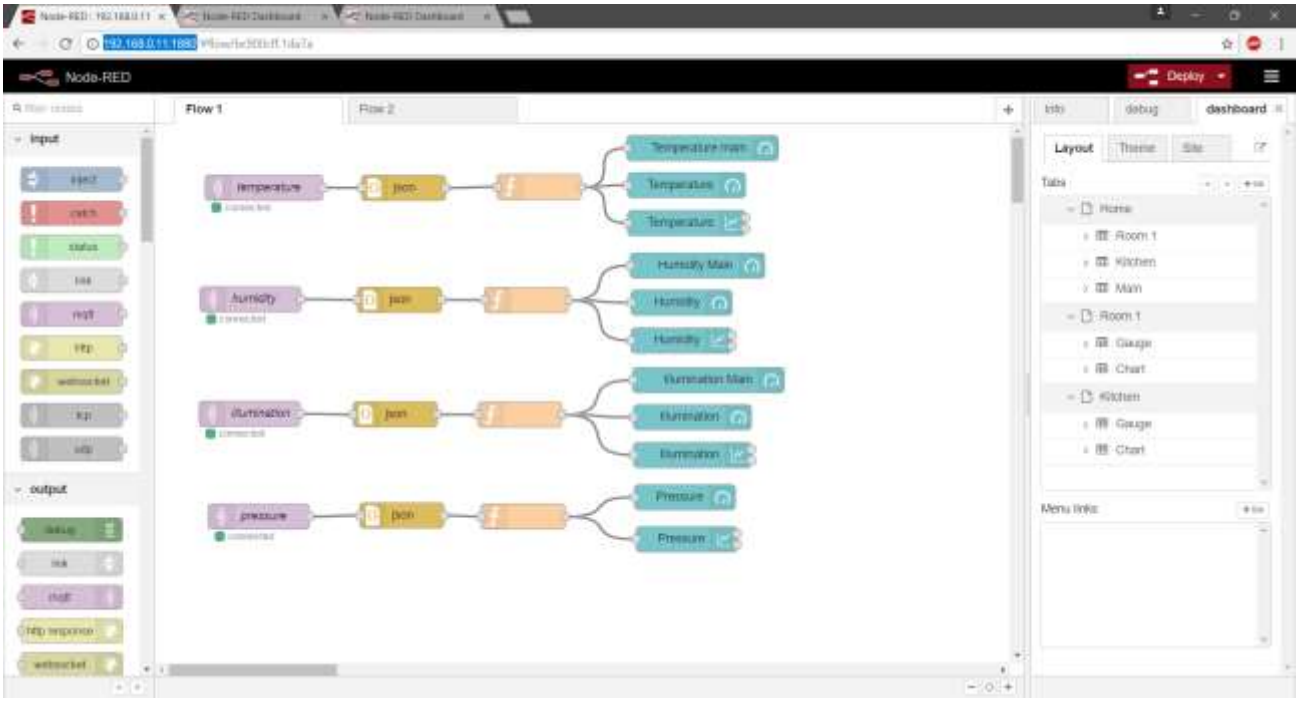

Рисунок 6.6 – Приклад конфігурації Node-Red

Для візуалізації результатів було створено 3 сторінки:

1. Головна
2. Кімната 1
3. Кухня

На головній сторінці відображаються наступні параметри:

- Поточна температура у обох приміщеннях
- Поточна вологість у обох приміщеннях
- Поточна освітленість у обох приміщеннях
- Поточний атмосферний тиск
- Графік атмосферного тиску

Зовнішній вигляд головної сторінки наведено на рис. 6.7.

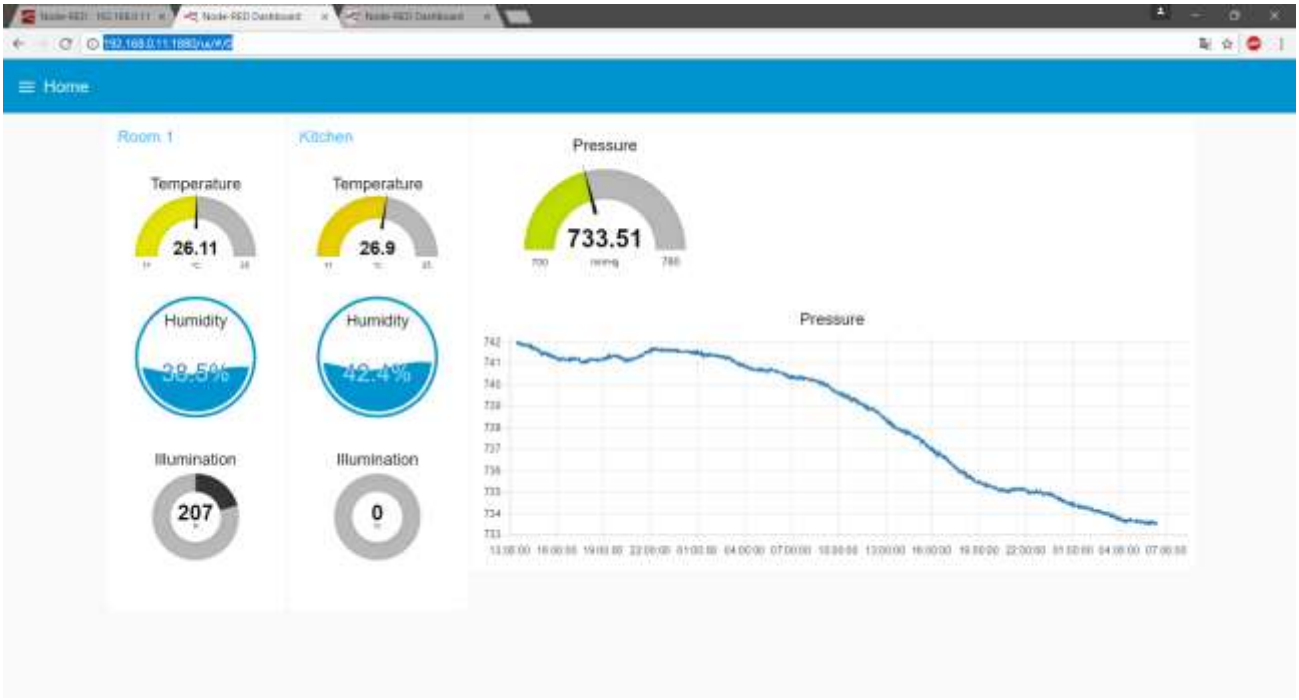

Рисунок 6.7 – Головна сторінка

На сторінках кімнати 1 та кухні розташовані наступні дані:

- Поточна температура
- Графік зміни температури
- Поточна вологість
- Графік зміни вологості
- Поточна освітленість
- Графік зміни освітленості

Зовнішній вигляд сторінки кімнати 1 наведено на рис. 6.8.

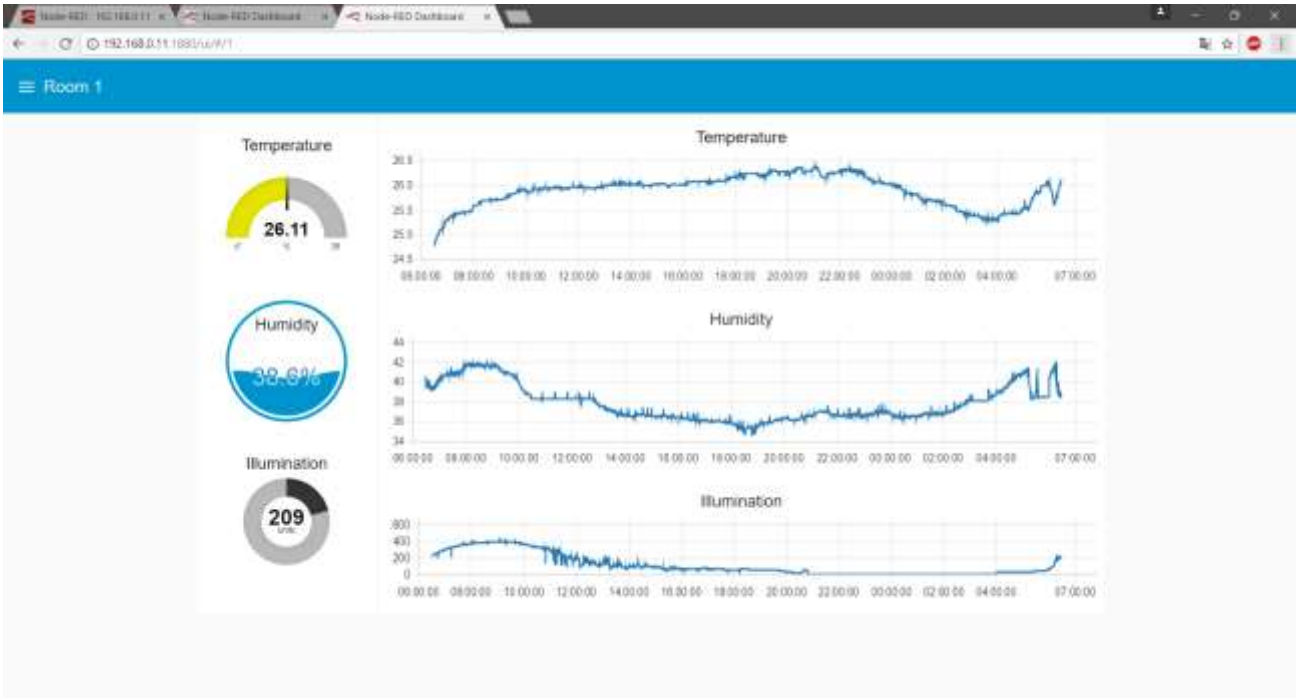

Рисунок 6.8 – Сторінка кімнати 1

Таким чином, усі пристрої чудово впоралися з поставленими задачами. Orange Pi Zero вистачає потужності для роботи 2-х контейнерів для обробки даних 2 приміщень, для значно більших проектів доцільним буде використання версій з процесорами H5 або A64. ESP8266 у свою чергу вистачає її можливостей для роботи з декількома датчиками та передачі даних через mqtt протокол.

7 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Метою даного розділу є проведення маркетингового аналізу стартап проекту для визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

7.1 Опис ідеї проекту

Опис ідеї стартап-проекту наведено у таблиці 7.1.

Таблиця 7.1 – Опис ідеї стартап-проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Розробка IoT системи, яка дозволяє контролювати параметри приміщення, віддалено вмикати/вимикати пристрої	1. Моніторинг параметрів приміщення	Дозволяє контролювати параметри у приміщенні, встановлювати бажані значення, переглядати статистику
	2. Віддалений контроль пристроїв	Дозволяє вмикати/вимикати пристрої
	3. Автоматизація пристроїв	Автоматизація вмикання вимикання пристроїв за сценаріями

У таблиці 7.2 наведено сильні, слабкі та нейтральні характеристики ідеї проекту.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент1	Конкурент2	Конкурент3			
1	Форма використання	Веб + мобільний телефон	Веб	Мобільний клієнт	Веб + мобільний телефон			+
2	Собівартість	Низька	Низька	Низька	Висока		+	
3	Наявність інтернету	-	+	+	+			+
4	Крос-платформеність	+/-	+	-	+	+		

7.2 Технологічний аудит ідеї проекту

Технологічну здійсненність ідей проекту наведено у таблиці 7.3.

Таблиця 7.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Створення веб-додатку	Flash	Наявна	Платна, недоступна
		Java-applets	Наявна	Безкоштовна, доступна
		ASP .NET	Наявна	Безкоштовна, доступна
2	Створення мобільного додатку	Android	Наявна	Безкоштовна, доступна
		IOS	Наявна	Платна, недоступна
3	Протокол взаємодії	MQTT	Наявна	Безкоштовна, доступна
		REST	Наявна	Безкоштовна, доступна
Обрана технологія реалізації ідеї проекту: Створення веб-додатку – Java тому що члени команди мають досвід роботи а також через поширеність технологій та простоті розробки, мобільного – Android бо безкоштовний а також пристроїв на цієї ОС значно більше, протокол взаємодії – MQTT, бо потребує менше ресурсів				

7.3 Аналіз ринкових можливостей запуску стартап-проекту

Попередню характеристику потенційного ринку стартап-проекту наведено у таблиці 7.4.

Таблиця 7.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п / п	**Показники стану ринку (найменування)**	**Характеристика**
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	5000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Характеристику потенційних клієнтів стартап-проекту наведено у таблиці 7.5.

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1.	Необхідність побудови розумної системи IoT у себе в дома	Люди з технічною освітою, які прагнуть оптимізувати своє життя	Різні розміри об’єктів, різні мобільні пристрої, різні сервіси інтеграції	Наявність веб інтерфейсу, віддаленого керування, програми для мобільного

Фактори загроз наведено у таблиці 7.6.

Таблиця 7.6 – Фактори загроз

№ п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1.	Конкуренція	Вихід на ринок великої компанії	1) Вихід з ринку 2) Запропонувати великій компанії поглинути себе 3) Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
2.	Зміна потреб користувачів	Користувачам необхідне програмне забезпечення з іншим функціоналом	1) Передбачити можливість додавання нового функціоналу до створюваного ПЗ

Фактори можливостей наведено у таблиці 7.7.

Таблиця 7.7 – Фактори можливостей

№ п/п	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1.	Зростання можливостей потенційних покупців	Ріст зацікавленості до продукту серед інших груп користувачів з різним рівнем технічної грамотності	Додати підказки, інструкції та демонстрації роботи системи
2.	Зниження довіри до конкурента 3	У ПЗ конкурента №3 нещодавно була знайдена помилка, завдяки чому вдалося отримати контроль над системою третій особі	При виході на ринок звертати увагу покупців на безпеку нашого ПЗ

У таблиці 7.8 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
1. Вказати тип конкуренції - досконала	Існує 3 фірми-конкурентки на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
2. За рівнем конкурентної боротьби - міжнародний	Компаній – з інших країни	Додати можливість вибору мови ПЗ, щоб легше було у майбутньому вийти на міжнародний ринок
3. За галузевою ознакою - внутрішньогалузева	Конкуренти мають ПЗ, яке використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко переробити дане ПЗ для використання у інших галузях
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме – програмне забезпечення	Створити ПЗ, враховуючи недоліки конкурентів
5. За характером конкурентних переваг - нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки, ніж використовують конкуренти
6. За інтенсивністю - марочна	Бренди присутні	Активна реклама, яка вказує на переваги саме даного рішення, натякаючи на недоліки конкурентів

У таблиці 7.9 наведено аналіз конкуренції в галузі за М. Портером.

Таблиця 7.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари-замінники**
	Навести перелік прямих конкурентів	**Визначити бар'єри входження в ринок**	**Визначити фактори сили постачальників**	**Визначити фактори сили споживачів**	**Фактори загроз з боку замінників**
Висновки:	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 3, так як його рішення також представлене у вигляді веб-додатку та мобільному додатку.	Є конкуренти, є можливість виходу на ринок	Постачальників багато, тому можна вважати, що вони не диктують умови на ринку	Важливим для користувача є наявність веб та мобільного додатку а також безпека системи	Товари-замінники можуть використати більш дешеву технологію створення ПЗ та зменшити собівартість товару.

У таблиці 7.10 наведено обґрунтування факторів конкурентоспроможності.

Таблиця 7.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)**
1.	Простота інтерфейсу користувача	Простота роботи х програмою спрощує роботу користувачеві, що робить її більш комфортною та зручною
2.	Наявність додатків під різні платформи	Дозволяє працювати з системою з різних пристроїв у зручній формі

У таблиці 7.11 наведено порівняльний аналіз сильних та слабких сторін

Таблиця 7.11 – Порівняльний аналіз сильних та слабких сторін

№ п/п	**Фактор конкурентоспроможності**	**Бали 1-20**	**Рейтинг товарів-конкурентів у порівнянні**						
			-3	**-2**	**-1**		**1**	**2**	**3**
1.	Простота інтерфейсу користувача	20				+			
2.	Наявність додатків під різні платформи	15			+				

У таблиці 7.12 наведено SWOT- аналіз стартап-проекту.

Таблиця 7.12 – SWOT- аналіз стартап-проекту

Сильні сторони: Простота інтерфейсу користувача, захищеність, наявність веб та мобільного додатку	Слабкі сторони: Відсутність IOS версії та розкрученості бренду
Можливості: у конкурента 3 виявлена проблема із безпекою ПЗ, зацікавленість продуктом ширших груп споживачів	Загрози: конкуренція, зміна потреб користувачів

У таблиці 7.13 наведено альтернативи ринкового впровадження стартап-проекту.

Таблиця 7.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	**Альтернатива (орієнтовний комплекс заходів) ринкової поведінки**	**Ймовірність отримання ресурсів**	**Строки реалізації**
1.	Використання MQTT для взаємодій пристроїв контролю параметрів	80%	1 місяць
2.	Використання RESTfull API для взаємодій пристроїв контролю параметрів	60%	2 місяці

Обираємо альтернативу 1, тому що вона має більшу ймовірність отримання ресурсів та менший час реалізації.

Після аналізу зазначити обрану альтернативу.

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими.

7.4 Розроблення ринкової стратегії проекту

У таблиці 7.14 наведено вибір цільових груп потенційних споживачів.

Таблиця 7.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Люди з технічною освітою, що мають схильність та ентузіазм до покращення оточуючої середи	Критичним є інтеграція з іншими сервісами, можливість використання сторонніх пристроїв	Контроль параметрів приміщення, автоматизація процесів	Існує 3 конкуренти, які надають схожі рішення. До того ж – лише 1 конкурент надає веб та мобільний додаток	У сегмент увійти просто, необхідно лише надати можливість до самостійної зміни деяких компонентів
2.	Люди, які мають достатньо коштів і небайдужі до впровадження нових рішень	Критичним є простій та зрозумілий інтерфейс	Контроль параметрів приміщень		Маючи простий та зрозумілий інтерфейс, вийти на ринок не складно
3.	Люди, які хочуть економити енергоносії	Критичним є енергоефективність	Автоматизація процесів, енергозбереження		Складно, необхідні складні механізми енергоефективності

Які цільові групи обрано: групи 1 та 2.

У таблиці 7.15 наведено визначення базової стратегії розвитку.

Таблиця 7.15 – Визначення базової стратегії розвитку

№ п/п	**Обрана альтернатива розвитку проекту**	**Стратегія охоплення ринку**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку***
1.	Використання MQTT для взаємодій пристроїв контролю параметрів	Ринкове позиціювання	Простота використання, пришвидшення роботи, крос-платформеність	Диференціації

У таблиці 7.16 наведено визначення базової стратегії конкурентної поведінки.

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	**Чи є проект «першопрохідцем» на ринку?**	**Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?**	**Чи буде компанія копіювати основні характеристики товару конкурента, і які?**	**Стратегія конкурентної поведінки***
1.	Ні	Так	Буде, наявність як веб так і мобільного додатку	Зайняття конкурентної ніші

У таблиці 7.17 наведено визначення стратегії позиціонування.

Таблиця 7.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Простота інтерфейсу, можливість кастомізації системи	Диференціації	Простота користувацького інтерфейсу, що дозволяє спростити роботу з системою, можливість кастомізації, що вимагає наявність можливості роботи з сторонніми системами та пристроями	Кастомізація, простота, гнучкість

7.5 Розроблення маркетингової програми стартап-проекту

У таблиці 7.18 наведено визначення ключових переваг концепції потенційного товару.

Таблиця 7.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1.	Кастомізація	Можливість інтеграції з іншими системами	Можливість роботи з модулями інших систем та пристроями
2.	Простота інтерфейсу	Простота та зручність ПЗ	Користувачам не потрібно замислюватися над тим як працювати з системою

У таблиці 7.19 наведено опис трьох рівнів моделі товару.

Таблиця 7.19 – Опис трьох рівнів моделі товару

Рівні товару	**Сутність та складові**		
I. Товар за задумом	Об’єкт дозволяє користувачам налаштувати параметри моніторингу стану приміщення, керувати віддалено пристроями		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Наявність веб додатку 2. Наявність мобільного додатку 3. Інтеграція з іншими сервісами 4. Простота використання	-	-
	Якість: згідно до стандарту ISO 29119 буде проведено тестування		
	Маркування присутнє.		
	Моя компанія. «ACIoT»		
III. Товар із підкріпленням	Відсутня підтримка до продажу		
	Постійна підтримка для користувачів після продажу		
За рахунок чого потенційний товар буде захищено від копіювання: ноу-хау.			

У таблиці 7.20 наведено визначення меж встановлення ціни.

Таблиця 7.20 – Визначення меж встановлення ціни

№ п/п	**Рівень цін на товари-замінники**	**Рівень цін на товари-аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі встановлення ціни на товар/послугу**
1	6000	7000	200000	5000

У таблиці 7.21 наведено формування системи збуту.

Таблиця 7.21 – Формування системи збуту

№ п/п	**Специфіка закупівельної поведінки цільових клієнтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
1.	Купують пристрої, ПЗ йде у комплекті	Продаж	0(напряму), 1(через одного посередника)	Власна та через посередників

У таблиці 7.22 наведено концепція маркетингових комунікацій.

Таблиця 7.22 – Концепція маркетингових комунікацій

№ п/п	**Специфіка поведінки цільових клієнтів**	**Канали комунікацій, якими користуються цільові клієнти**	**Ключові позиції, обрані для позиціонування**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
1.	Замовлення через інтернет. Використання пристроїв на різних платформах	Інтернет	Можливість кастомізацій, наявність веб та мобільного додатку, простота інтерфейсу	Показати переваги ПЗ, у тому числі і перед конкурентами	Демо-ролик із використання

7.6 Висновки

Згідно до проведених досліджень:

- існує можливість ринкової комерціалізації проекту;
- існують перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження не є високими, проект має дві значні переваги перед конкурентами;
- необхідно реалізувати веб-додаток, мобільний додаток та хаб з MQTT;
- подальша імплементація є доцільною.

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 та підвищити комфорт населення. Усе це створює чудові передумови до росту цієї галузі

В даній роботі було поставлено задачу провести аналіз можливостей різних моделей мікроконтролерів та міні комп'ютерів з метою використання їх для побудови IoT систем. Також було поставлено задачу дослідити можливості подібних пристроїв для організації паралельних обчислень і проаналізувати питання безпеки їх використання, а також протоколи для взаємодії.

Для реалізації поставленої задачі було обрано мікроконтролери від Arduino та ESP, а також міні комп'ютери Raspberry та Orange Pi. Порівняння обчислювальних можливостей виконувалося на мовах програмування: C++ для мікроконтролерів та міні комп'ютерів, Java та Python для міні ПК. Для порівняння було виконано 2 навантажувальні тести для кожної з мов програмування.

Перший тест полягав у складанні елементів масиву різного розміру, що дозволив оцінити швидкодію пристроїв. За результатами цього тестування можна прийти до висновків, що серед мікроконтролерів найкращий результат має ESP32, далі йде ESP8266. Серед міні ПК Raspberry Pi 3B показав найкращі результати у всіх 3 мовах програмування, результати Raspberry Pi B, який має ті ж самі характеристики, що й Raspberry Pi Zero є гіршими ніж Orange Pi Zero, який, в свою чергу, трохи відстає від Raspberry Pi 3B.

Другий тест полягав у виконані get запитів. Найкращий результат у мікроконтролерів у ESP8266, а серед міні ПК ситуація склалася аналогічним чином, що й з тестом 1.

Таким чином, можна прийти до висновків, що ESP8266 має достатню функціональність, потужність та досить не високу вартість, а враховуючи, вбудований модуль Wi-Fi, його можна вважати оптимальним рішенням серед мікроконтролерів для основи різноманітних датчиків.

Серед міні комп'ютерів ситуація склалась не так однозначно. Найкращі результати по більшості тестів має Raspberry Pi 3B, але його вартість значно вище рішень від Orange Pi та й до того присутні деякі проблеми з паралельністю на мові C++. Тому можна зробити висновок що найкращим варіантом для хаба пристроїв або міні сервера будуть рішення від Orange Pi з більш потужним апаратним забезпеченням, ніж модель Zero, їх вартість все одно буде нижче ніж у Raspberry.

Аналіз протоколів передачі даних показав, що найбільш доцільними є протоколу MQTT та REST. 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.

Дослідження паралельних обчислень продемонструвало, що вони можливі на багатоядерних міні ПК. Найбільш стабільні результати прирости швидкодії показують програми на Java. 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 Orange Pi Zero, розгортанням MQTT брокера та Node-Red

візуалізатора як Docker контейнерів. В якості датчиків було використано контролер ESP8266 з сенсорами температури, вологості та освітленості.

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

ПРЕЛІК ПОСИЛАНЬ

1. The Internet of Things - Cisco – Режим доступу : http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf – Дата доступу : 20.05.2017.
2. WHEN WOMAN IS BOSS – Режим доступу : http://www.tfcbooks.com/tesla/1926-01-30.htm. – Дата доступу : 20.05.2017.
3. Умный холодильник или будущее за интернетом вещей – Режим доступу : https://rb.ru/news/internet-of-things/. – Дата доступу : 20.05.2017.
4. Arduino Ethernet – Режим доступу : http://arduino.ua/ru/hardware/Ethernet. – Дата доступу : 19.02.2017.
5. Arduino YUN – Режим доступу : http://arduino.ua/ru/hardware/YUN. – Дата доступу : 19.02.2017.
6. Arduino TITAN – Режим доступу : https://www.arduino.cc/en/Main/ArduinoBoardTian. – Дата доступу : 20.02.2017.
7. D1 mini pro – Режим доступу : https://www.wemos.cc/product/d1-mini-pro.html. – Дата доступу : 20.02.2017.
8. SparkFun ESP32 Thing – Режим доступу : https://www.sparkfun.com/products/13907. – Дата доступу : 20.02.2017.
9. RASPBERRY PI ZERO – Режим доступу : https://www.raspberrypi.org/products/pi-zero/. – Дата доступу : 20.02.2017.
10. RASPBERRY PI 3 MODEL B – Режим доступу : https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. – Дата доступу : 20.02.2017.
11. NanoPi-NEO – Режим доступу : http://www.nanopi.org/NanoPi-NEO_Feature.html#order. – Дата доступу : 20.02.2017.

12. NanoPi-NEO-Air – Режим доступу : http://www.nanopi.org/NanoPi-NEO-Air_Feature.html. – Дата доступу : 20.02.2017.
13. Orange Pi Zero – Режим доступу : http://www.orangepi.org/orangepizero/. – Дата доступу : 20.02.2017.
14. IBM Knowledge Center - How sockets work – Режим доступу : http://www.ibm.com/support/knowledgecenter/ssw_ibm_i_71/rzab6/howdosockets.htm/. – Дата доступу : 20.02.2017.
15. REST – Режим доступу : https://uk.wikipedia.org/wiki/REST. – Дата доступу : 20.02.2017.
16. RESTful Web services: The basics – Режим доступу : http://www.ibm.com/developerworks/webservices/library/ws-restful/index.html?S_TACT=105AGX99&S_CMP/. – Дата доступу : 20.02.2017.
17. How to Create RESTful Java Client With Jersey Client – Example – Режим доступу : http://crunchify.com/how-to-create-restful-java-client-with-jersey-client-example/. – Дата доступу : 20.02.2017.
18. MQTT и Modbus: сравнение протоколов, используемых в шлюзах для IoT – Режим доступу : https://habrahabr.ru/company/intel/blog/304228/. – Дата доступу : 20.02.2017.
19. SNMP протокол (основы) – Режим доступу : http://www.k-max.name/linux/snmp-protocol/. – Дата доступу : 20.02.2017.
20. Многопоточность в Java – Режим доступу : https://habrahabr.ru/post/164487/. – Дата доступу : 20.02.2017.
21. Download Raspbian for Raspberry Pi – Режим доступу : https://www.raspberrypi.org/downloads/raspbian/ – Дата доступу : 02.03.2017.
22. Orange Pi Zero - armbian – Режим доступу : https://www.armbian.com/orange-pi-zero/ – Дата доступу : 02.03.2017.

23. DOCKER COMES TO RASPBERRY PI – Режим доступу : https://www.raspberrypi.org/blog/docker-comes-to-raspberry-pi/ – Дата доступу : 02.03.2017.
24. How to get docker-compose (fig) up and running on Raspberry Pi 2 – Режим доступу : https://gist.github.com/oysteinjakobsen/e59cdd38a688ee8a418a – Дата доступу : 02.03.2017.
25. How to run a Raspberry Pi cluster with Docker Swarm – Режим доступу : https://howchoo.com/g/njy4zdm3mwy/how-to-run-a-raspberry-pi-cluster-with-docker-swarm – Дата доступу : 02.03.2017.
26. Взлом Wi-Fi WPA/WPA2– Режим доступу : http://variable.pp.ua/взлом-wi-fi-wpawpa2-для-чайников/ – Дата доступу : 02.05.2017.
27. Защищаем SSH от брутфорса на любом порту – Режим доступу : https://habrahabr.ru/post/88461/ – Дата доступу : 02.05.2017.