

Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет робототехніки та штучного інтелекту
(повне найменування факультету)

Кафедра електроніки, фізики та SMART-систем
(повне найменування кафедри)

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**ПРОЄКТУВАННЯ ВБУДОВАНОЇ ТЕЛЕКОМУНІКАЦІЙНОЇ
СИСТЕМИ ПЕРЕДАЧІ ДАНИХ З ВИКОРИСТАННЯМ
СУЧАСНИХ ПРОТОКОЛІВ (MQTT/UDP/TCP)**

**DESIGN OF AN EMBEDDED TELECOMMUNICATION DATA
TRANSMISSION SYSTEM USING MODERN PROTOCOLS
(MQTT/UDP/TCP)**

спеціальність Телекомунікації та радіотехніка
(шифр і назва спеціальності)

освітня програма «Телекомунікації та радіотехніка»
(назва освітньої програми)

**Виконав: здобувач вищої освіти групи ТК-41
Гірчук Роман Володимирович**

(підпис)

**Керівник: к.ф.-м.н., доцент
Хвищун Микола Вячеславович**

(підпис)

Кваліфікаційну роботу
допущено до захисту
«__» червня 2026 р.
Гарант освітньої програми:
к.т.н., доцент
Євсюк Микола Миколайович

(підпис)

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра електроніки та телекомунікацій

Ступінь вищої освіти бакалавр

Галузь знань: 17 Електроніка, автоматизація та електронні комунікації

Спеціальність: 172 Електронні комунікації та радіотехніка

Освітня програма: «Телекомунікації та радіотехніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«__» _____ 20__ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Гірчуку Роману Володимировичу

1. Тема кваліфікаційної роботи: *Проектування вбудованої телекомунікаційної системи передачі даних з використанням сучасних протоколів (MQTT/UDP/TCP)*

керівник роботи: *к.ф.-м.н., доцент Хвищун М. В.*

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 558/01-02

2. Строк подання здобувачем роботи: *04.06.2026 р.*

3. Вихідні дані до роботи: *апаратна платформа – ESP32/STM32/Raspberry Pi;*

мережеві протоколи – MQTT, UDP, TCP/IP; тип мережі – Wi-Fi/Ethernet;

архітектура системи – клієнт-сервер та Publisher-Subscriber; тип системи – вбудована телекомунікаційна IoT-система передачі даних у режимі реального часу.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Вступ

Розділ 1. Аналітичний огляд телекомунікаційних систем та протоколів передачі даних

Розділ 2. Схемне проектування та принцип роботи системи передачі даних

Розділ 3. Розрахункова апаратно-програмна частина

Розділ 4. Охорона праці та безпека

Висновки

5. Перелік графічного матеріалу

Презентація POWERPOINT (12 слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
<i>Розділ 1 Аналітичний огляд систем широтно-імпульсного керування</i>	<i>Хвищун М. В., доцент</i>		
<i>Розділ 2 Схемне проектування та принцип роботи системи передачі даних</i>	<i>Хвищун М. В., доцент</i>		
<i>Розділ 3 Розрахункова апаратно-програмна частина</i>	<i>Хвищун М. В., доцент</i>		
<i>Розділ 4 Охорона праці та безпека</i>	<i>Заблоцький В. Ю., доцент</i>		
<i>Нормоконтроль</i>	<i>Селепина Й. Р., доцент</i>		
<i>Гарант ОП</i>	<i>Євсюк М. М., доцент</i>		
<i>Показник запозичень тексту</i>	<u> </u> %		
<i>Академічна доброчесність</i>	<i>Хвищун М. В., доцент</i>		

7. Дата видачі завдання «31» грудня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	<i>Розділ 1 Аналітичний огляд телекомунікаційних систем та протоколів передачі даних</i>	до 01.03.2026 р.	
2.	<i>Розділ 2 Схемне проектування та принцип роботи системи передачі даних</i>	до 15.03.2026 р.	
3.	<i>Розділ 3 Розрахункова апаратно-програмна частина</i>	до 01.04.2026 р.	
4.	<i>Розділ 4 Охорона праці та безпека</i>	до 10.04.2026 р.	
5.	<i>Висновки</i>	до 20.04.2026 р.	
6.	<i>Формування списку використаних джерел</i>	до 30.04.2026 р.	
7.	<i>Оформлення пояснювальної записки та ілюстративного матеріалу</i>	до 10.05.2026 р.	
8.	<i>Нормоконтроль</i>	до 20.05.2026 р.	
9.	<i>Інструментальна перевірка на академічний плагіат</i>	до 30.05.2026 р.	
10.	<i>Представлення кваліфікаційної роботи до захисту</i>	до 04.06.2026 р.	

Здобувач вищої освіти _____ Роман ГІРЧУК

Керівник роботи _____ Микола ХВИЩУН

АНОТАЦІЯ

Гірчук Р. В. Проєктування вбудованої телекомунікаційної системи передачі даних з використанням сучасних протоколів (MQTT/UDP/TCP). Рукопис. Кваліфікаційна робота бакалавра ОП «Телекомунікації та радіотехніка» спеціальності 172 Електронні комунікації та радіотехніка. Луцький національний технічний університет. Луцьк, 2026. 79 с.

Кваліфікаційна робота бакалавра складається зі вступу, чотирьох розділів, висновків, списку використаних джерел.

У першому розділі бакалаврської роботи проведено аналіз сучасних телекомунікаційних систем та технологій Інтернету речей (IoT), виконано порівняльний аналіз мережевих протоколів MQTT, UDP та TCP. Обґрунтовано вибір архітектури телекомунікаційної системи та протоколів передачі даних.

У другому розділі виконано схемне проєктування вбудованої телекомунікаційної системи передачі даних. Розроблено структурну, функціональну та принципову електричну схеми системи, описано принцип роботи та організацію мережевої взаємодії між вузлами системи.

У третьому розділі проведено розрахунок параметрів передачі даних, затримок, пропускну здатності та ефективності використання протоколів MQTT, UDP та TCP. Розроблено програмне забезпечення для реалізації TCP/UDP обміну та MQTT Publisher/Subscriber архітектури. Проведено оцінку швидкодії, енергоефективності та надійності функціонування системи.

У четвертому розділі розглянуто питання охорони праці, електробезпеки, електромагнітної сумісності та кібербезпеки під час експлуатації.

У висновках підсумовано результати роботи. Встановлено, що використання протоколу MQTT забезпечує високу ефективність передачі даних у IoT-системах, UDP мінімізує затримки передачі, а TCP забезпечує надійність обміну інформацією.

Ключові слова: IoT, MQTT, UDP, TCP/IP, телекомунікаційна система, передача даних, вбудована система, Wi-Fi, MQTT-брокер.

ABSTRACT

Hirchuk R. Design of an Embedded Telecommunication Data Transmission System Using Modern Protocols (MQTT/UDP/TCP). Manuscript. Qualification work for bachelor's degree in the Educational Program «Telecommunications and Radio Engineering», specialty 172 Electronic Communications and Radio Engineering. Lutsk National Technical University. Lutsk, 2026. 79 p.

The bachelor's qualification work consists of an introduction, four chapters, conclusions, a list of references, and appendices.

In the first chapter, an analysis of modern telecommunication systems and Internet of Things (IoT) technologies was carried out, and a comparative analysis of MQTT, UDP, and TCP network protocols was performed. The choice of telecommunication system architecture and data transmission protocols was substantiated.

In the second chapter, the circuit design of the embedded telecommunication data transmission system was developed. Structural, functional, and schematic electrical diagrams of the system were created, and the operating principle and organization of network interaction between system nodes were described.

The third chapter presents the calculation of data transmission parameters, delays, bandwidth, and the efficiency of MQTT, UDP, and TCP protocols. Software for TCP/UDP communication and MQTT Publisher/Subscriber architecture implementation was developed. The system performance, energy efficiency, and operational reliability were evaluated.

The fourth chapter considers occupational safety, electrical safety, electromagnetic compatibility, and cybersecurity issues.

The conclusions summarize the obtained results. It was established that the MQTT protocol provides high efficiency of data transmission in IoT systems, UDP minimizes transmission delays, and TCP ensures reliable information exchange.

Keywords: IoT, MQTT, UDP, TCP/IP, Telecommunication System, Data Transmission, Embedded System, Wi-Fi, MQTT Broker.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ ТА ПРОТОКОЛІВ ПЕРЕДАЧІ ДАНИХ	9
1.1 Аналіз предметної області та постановка задачі	9
1.2 Огляд існуючих аналогів і технічних рішень	13
1.3 Порівняльний аналіз підходів та вибір оптимального рішення.....	20
1.4 Формування вимог до розроблюваної системи	26
РОЗДІЛ 2 СХЕМНЕ ПРОЄКТУВАННЯ ТА ПРИНЦИП РОБОТИ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ	31
2.1 Розробка структурної та функціональної схеми системи.....	31
2.2 Розробка принципової електричної схеми	38
2.3 Опис роботи основних вузлів та їх взаємодії.....	45
РОЗДІЛ 3 РОЗРАХУНКОВА АПАРАТНО-ПРОГРАМНА ЧАСТИНА.....	52
3.1 Розрахунок параметрів передачі даних	52
3.2 Оцінка ефективності протоколів MQTT, UDP, TCP	56
3.3 Розробка алгоритмів передачі даних та програмного забезпечення	60
РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА	69
4.1 Аналіз небезпечних факторів при роботі з телекомунікаційним обладнанням.....	69
4.2 Ергономіка та безпечна організація робочого місця.....	72
ВИСНОВКИ	76
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	77

ВСТУП

Актуальність теми. У сучасних умовах розвитку цифрових технологій, телекомунікаційних систем та концепції Інтернету речей (IoT) особливого значення набуває забезпечення ефективної та надійної передачі даних між пристроями у режимі реального часу. Сучасні IoT-системи широко застосовуються у промисловій автоматизації, транспортних системах, енергетиці, телекомунікаційних мережах, системах моніторингу та «розумних» інфраструктурах. Ефективність функціонування таких систем значною мірою залежить від вибору мережевих протоколів передачі даних, які повинні забезпечувати мінімальні затримки, високу надійність та енергоефективність роботи системи [1], [2].

У вбудованих телекомунікаційних системах найбільш поширеними є протоколи MQTT, UDP та TCP. Протокол MQTT є легковаговим та оптимізованим для IoT-систем, UDP забезпечує мінімальні затримки передачі даних, а TCP гарантує надійність обміну інформацією та контроль помилок. Використання зазначених протоколів дозволяє реалізувати ефективні системи передачі даних для різних умов експлуатації та вимог до швидкодії системи [1], [3], [4], [5].

Важливим напрямом розвитку сучасних телекомунікацій є створення вбудованих систем передачі даних на базі мікроконтролерних платформ STM32 та Raspberry Pi із використанням Wi-Fi, Ethernet та хмарних сервісів. Такі системи забезпечують інтеграцію пристроїв у єдину мережу, автоматизований обмін інформацією та підтримку дистанційного моніторингу і керування [1], [11], [12].

Метою кваліфікаційної роботи є проєктування вбудованої телекомунікаційної системи передачі даних із використанням сучасних протоколів MQTT, UDP та TCP для забезпечення ефективного функціонування системи в умовах реального часу.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасних телекомунікаційних систем та концепції Інтернету речей;

- виконати порівняльний аналіз протоколів MQTT, UDP та TCP;
- обґрунтувати вибір архітектури телекомунікаційної системи;
- розробити структурну, функціональну та принципову схеми системи;
- реалізувати програмне забезпечення для TCP/UDP та MQTT обміну;
- виконати розрахунок параметрів передачі даних;
- провести оцінку ефективності та надійності роботи системи.

Об'єктом дослідження є вбудована телекомунікаційна система передачі даних.

Предметом дослідження є методи та протоколи передачі даних MQTT, UDP та TCP у вбудованих телекомунікаційних системах.

Практичне значення роботи полягає у розробці вбудованої телекомунікаційної системи передачі даних, яка може бути використана у IoT-системах, системах моніторингу, автоматизації, телекомунікаційних мережах та навчально-дослідницьких проєктах.

Під час виконання роботи засоби штучного інтелекту використовувалися лише як допоміжний інструмент для редагування тексту та опрацювання літературних джерел; отримані результати перевірено автором відповідно до принципів академічної доброчесності.

РОЗДІЛ 1

АНАЛІТИЧНИЙ ОГЛЯД ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ ТА ПРОТОКОЛІВ ПЕРЕДАЧІ ДАНИХ

1.1 Аналіз предметної області та постановка задачі

Сучасний розвиток телекомунікаційних технологій супроводжується активним впровадженням концепції Інтернету речей (IoT), яка передбачає інтеграцію великої кількості пристроїв у єдину інформаційну мережу для автоматизованого збору, передавання, оброблення та зберігання даних. Вбудовані телекомунікаційні системи є основою таких рішень, оскільки забезпечують взаємодію між датчиками, мікроконтролерами, серверами та хмарними платформами [1].

IoT-системи широко застосовуються у промисловій автоматизації, транспортних системах, телекомунікаційних мережах, системах безпеки, «розумних» будинках та енергетичних комплексах. Основною особливістю таких систем є необхідність функціонування у режимі реального часу, що вимагає забезпечення мінімальних затримок передачі даних, високої достовірності інформації та ефективного використання енергетичних ресурсів [1], [2].

Вбудовані телекомунікаційні системи характеризуються:

- обмеженими апаратними ресурсами;
- низьким енергоспоживанням;
- необхідністю стабільної роботи у реальному часі;
- використанням бездротових мережевих технологій;
- інтеграцією із хмарними платформами та сервісами [1].

Структура системи включає такі основні компоненти:

- давачі збору інформації;
- мікроконтролерний вузол;
- комунікаційний модуль;
- сервер або MQTT-брокер;

- хмарну платформу оброблення даних;
- клієнтські застосунки для моніторингу та керування.

Сучасні телекомунікаційні системи використовують різні мережеві інтерфейси, серед яких найбільш поширеними є:

- Wi-Fi;
- Ethernet;
- Bluetooth Low Energy (BLE);
- ZigBee;
- GSM/4G/5G [1].

Передавання даних у режимі реального часу є одним із ключових завдань сучасних телекомунікаційних систем. Основною вимогою до таких систем є забезпечення доставки інформації у визначений проміжок часу із мінімальними втратами пакетів та мінімальною затримкою [1].

До основних параметрів систем реального часу належать:

- затримка передачі;
- джиттер (варіація затримки);
- достовірність доставки;
- пропускна здатність;
- енергоспоживання [1].

У таблиці 1.1 наведено основні вимоги до телекомунікаційних систем реального часу.

Таблиця 1.1 – Основні вимоги до систем передачі даних реального часу

Параметр	Характеристика	Типові значення
Затримка передачі	Час доставки даних	< 100 мс
Джиттер	Коливання затримки	< 20 мс
Достовірність	Ймовірність доставки	> 99 %
Пропускна здатність	Швидкість передачі	До декількох Мбіт/с
Енергоспоживання	Потужність системи	Мінімальна

Для IoT-систем особливого значення набуває енергоефективність,

оскільки значна кількість пристроїв працює автономно від акумуляторів або елементів живлення [1].

Проведемо аналіз сучасних протоколів передавання даних. У телекомунікаційних системах IoT найчастіше використовуються протоколи TCP, UDP та MQTT. Кожен із них має власні особливості, переваги та недоліки.

Протокол TCP (Transmission Control Protocol) є транспортним протоколом із встановленням з'єднання та забезпечує:

- контроль доставки пакетів;
- перевірку цілісності даних;
- повторну передачу втрачених пакетів;
- впорядкування сегментів інформації [1].

Протокол UDP (User Datagram Protocol) є спрощеним транспортним протоколом без встановлення з'єднання. Основні особливості UDP:

- мінімальні затримки;
- висока швидкість передачі;
- низьке навантаження на мережу;
- відсутність механізмів контролю доставки [5].

Протокол MQTT (Message Queuing Telemetry Transport) є легковаговим прикладним протоколом типу publish/subscribe, розробленим спеціально для IoT-систем [1], [3], [9].

MQTT функціонує поверх TCP/IP та використовує центральний брокер повідомлень для маршрутизації інформації між вузлами системи.

На рисунку 1.1 показано механізми передавання даних:

- TCP – із встановленням з'єднання;
- UDP – без встановлення з'єднання;
- MQTT – через брокер повідомлень.

Для вибору оптимального протоколу передачі даних необхідно виконати порівняльний аналіз їх характеристик.



Рисунок 1.1 – Принципи роботи протоколів TCP, UDP та MQTT [9]

У таблиці 1.2 подано порівняння протоколів MQTT, UDP та TCP.

Таблиця 1.2 – Порівняння протоколів MQTT, UDP та TCP

Характеристика	MQTT	UDP	TCP
Тип взаємодії	Через брокер	Без з’єднання	З’єднання
Надійність	Висока	Низька	Висока
Затримка	Низька	Мінімальна	Вища
Контроль помилок	Частковий	Відсутній	Повний
Енергоспоживання	Дуже низьке	Низьке	Середнє
Підтримка IoT	Висока	Часткова	Обмежена
Складність реалізації	Середня	Низька	Вища

Проведений аналіз показує, що MQTT найбільш адаптований для IoT-систем, UDP забезпечує мінімальні затримки, а TCP гарантує максимальну

достовірність передачі даних [1], [3], [4], [5].

У системах реального часу доцільно використовувати UDP, а у критично важливих системах – TCP. MQTT є найбільш універсальним рішенням для телекомунікаційних IoT-систем.

Проведемо постановку задачі дослідження. На основі проведеного аналізу встановлено, що сучасні телекомунікаційні системи потребують ефективних рішень для передавання даних у режимі реального часу.

Основною задачею бакалаврської роботи є проєктування вбудованої телекомунікаційної системи передачі даних із використанням протоколів MQTT, UDP та TCP для забезпечення:

- високої швидкодії;
- мінімальної затримки;
- високої достовірності передачі;
- енергоефективності;
- масштабованості системи.

Таким чином, основним завданням є пошук оптимального поєднання швидкодії, надійності та енергоефективності системи передачі даних.

Отже, у результаті проведеного аналізу встановлено, що сучасні телекомунікаційні системи та IoT-рішення потребують ефективних механізмів передачі даних у режимі реального часу. Визначено основні особливості протоколів MQTT, UDP та TCP, а також їх переваги та недоліки. Встановлено, що MQTT є найбільш адаптованим до IoT-систем, UDP забезпечує мінімальні затримки, а TCP гарантує високу достовірність передачі даних. На основі проведеного аналізу сформовано постановку задачі дослідження щодо проєктування вбудованої телекомунікаційної системи передачі даних.

1.2 Огляд існуючих аналогів і технічних рішень

Проведемо аналіз сучасних архітектур телекомунікаційних систем. Сучасні системи передачі даних будуються за багаторівневим принципом та

включають периферійні пристрої, комунікаційні вузли, серверні системи та засоби хмарної обробки інформації. Основною тенденцією розвитку телекомунікаційних систем є перехід до розподілених архітектур із підтримкою IoT та edge computing [1].

На рисунку 1.2 наведено типову архітектуру сучасної телекомунікаційної системи передачі даних.

У сучасних embedded-системах активно застосовується багаторівнева структура:

- рівень сенсорів;
- рівень локальної обробки;
- рівень шлюзів;
- мережевий рівень;
- рівень хмарної обробки;
- рівень клієнтських сервісів.

Одним із ключових компонентів сучасних телекомунікаційних систем є IoT-платформи, які забезпечують маршрутизацію повідомлень, керування пристроями та оброблення даних.

Платформа AWS IoT Core забезпечує обмін даними між IoT-пристроями через MQTT та HTTP. Основними перевагами є: [1], [3], [6]

- висока масштабованість;
- підтримка TLS-шифрування;
- інтеграція з хмарними сервісами AWS;
- підтримка великої кількості пристроїв.

Недоліком є складність конфігурації та значне споживання хмарних ресурсів.

Microsoft Azure IoT Hub підтримує двосторонній обмін повідомленнями між вузлами системи та дозволяє здійснювати централізоване керування пристроями [1], [7].

Основні переваги:

- підтримка MQTT, AMQP та HTTP;

- інтеграція із сервісами Microsoft Azure;
- підтримка edge computing;
- висока надійність передачі даних.

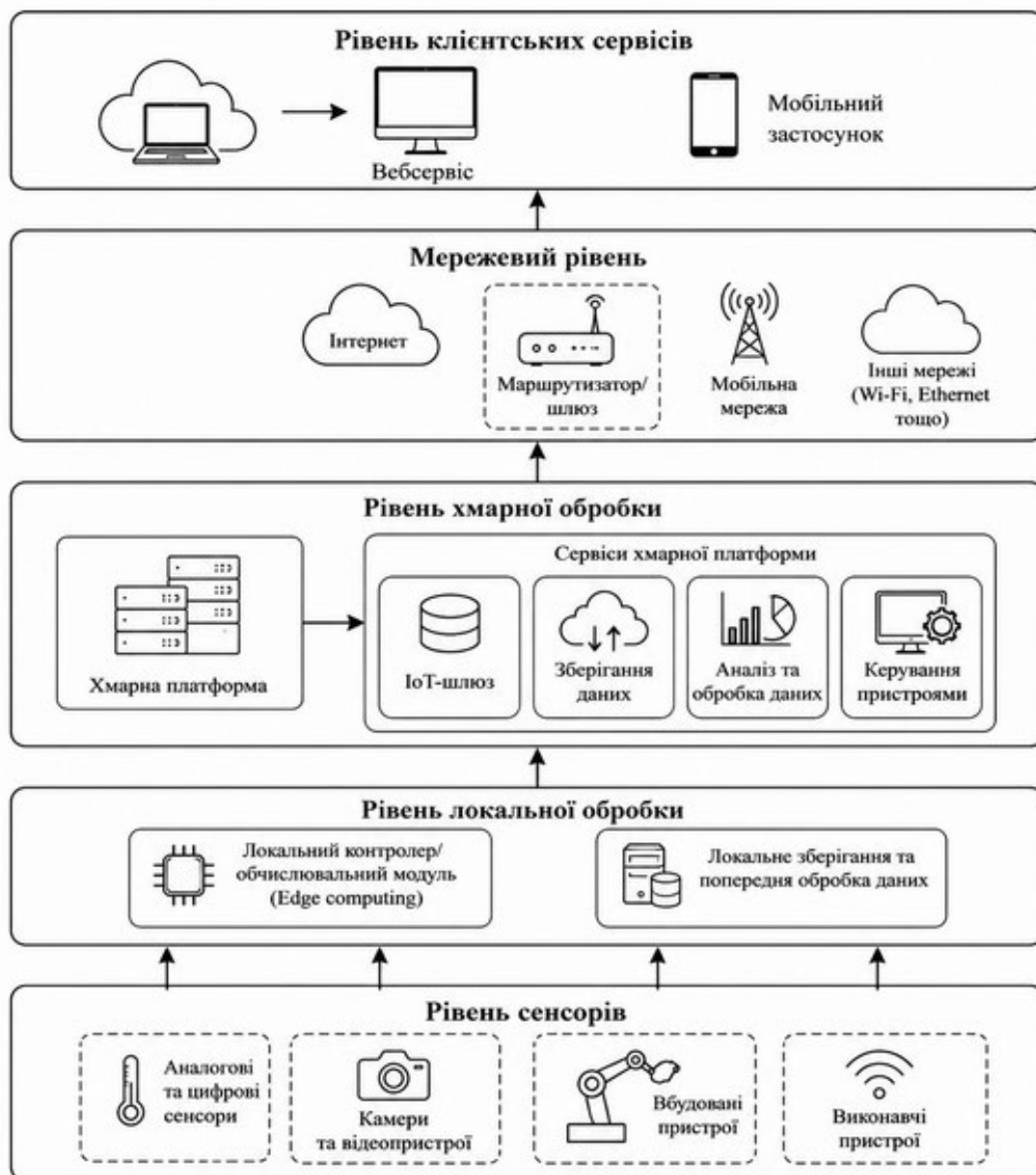


Рисунок 1.2 – Архітектура сучасної телекомунікаційної системи передачі даних

Недоліком є висока складність реалізації для невеликих embedded-систем.

Google Cloud IoT орієнтована на роботу з великими потоками інформації та інтеграцію із системами аналітики й штучного інтелекту [1], [8].

Переваги:

- інтеграція з Big Data;
- підтримка MQTT;
- висока продуктивність;
- підтримка AI-сервісів.

У таблиці 1.3 наведено порівняння сучасних IoT-платформ.

Таблиця 1.3 – Порівняння IoT-платформ

Платформа	Підтримувані протоколи	Масштабованість	Надійність	Складність
AWS IoT Core	MQTT, HTTP	Висока	Висока	Висока
Azure IoT Hub	MQTT, AMQP	Висока	Висока	Середня
Google Cloud IoT	MQTT, HTTP	Висока	Висока	Середня

Проведений аналіз показує, що всі сучасні IoT-платформи орієнтовані на підтримку MQTT та інтеграцію із хмарними сервісами.

Для реалізації embedded-телекомунікаційних систем використовуються різні апаратні платформи, серед яких найбільш поширеними є Arduino, STM32, ESP32 та Raspberry Pi.

Arduino Uno широко використовується у навчальних та базових IoT-проєктах [1].

Основні переваги:

- простота програмування;
- велика кількість бібліотек;
- низька вартість;
- підтримка Ethernet та Wi-Fi модулів.

Недоліки:

- обмежена продуктивність;
- невеликий обсяг пам'яті;

- низька швидкодія.

STM32 є сімейством ARM-мікроконтролерів, орієнтованих на промислові embedded-системи [1], [13].

STM32 широко використовується у промислових системах керування та телеметрії. ESP32 є сучасною IoT-платформою з інтегрованими модулями Wi-Fi та Bluetooth [1], [10], [13].

Основні переваги:

- підтримка MQTT;
- підтримка FreeRTOS;
- низьке енергоспоживання;
- двоядерна архітектура.

ESP32 є одним із найбільш популярних рішень для сучасних IoT-систем.

Raspberry Pi використовується для побудови складних embedded-систем із підтримкою Linux [1], [12].

Переваги:

- висока продуктивність;
- підтримка Docker;
- підтримка MQTT-брокерів;
- можливість роботи з базами даних.

Недоліки:

- більше енергоспоживання;
- вища вартість.

Проведемо аналіз сучасних моделей передачі даних. У сучасних телекомунікаційних системах використовуються різні моделі взаємодії між вузлами мережі. Найбільш поширеними є:

- клієнт-серверна модель;
- peer-to-peer;
- publish/subscribe;
- брокерна архітектура [1].

Клієнт-серверна архітектура передбачає централізовану взаємодію між

клієнтами та сервером. Такий підхід забезпечує високу надійність та контроль передачі даних, проте збільшує навантаження на сервер.

Peer-to-peer архітектура забезпечує прямий обмін інформацією між вузлами системи без центрального сервера. Перевагою є мінімальні затримки передачі.

Publish/subscribe модель активно використовується у IoT-системах. У такій архітектурі повідомлення передаються через центральний брокер, який виконує маршрутизацію інформації між пристроями.

Проведений аналіз показує, що брокерна архітектура є найбільш ефективною для масштабованих IoT-систем.

Проведемо аналіз сучасних технічних рішень для embedded-систем.

У таблиці 1.4 наведено порівняння сучасних мережевих технологій.

Таблиця 1.4 – Порівняння сучасних мережевих технологій

Технологія	Швидкість	Дальність	Енергоспоживання	Галузь застосування
Wi-Fi	Висока	Середня	Середнє	IoT, smart home
Ethernet	Висока	До 100 м	Середнє	Промислові системи
BLE	Низька	До 10 м	Дуже низьке	Мобільні пристрої
ZigBee	Середня	До 100 м	Низьке	Сенсорні мережі
GSM/4G/5G	Дуже висока	Велика	Високе	Телеметрія

Сучасні embedded-телекомунікаційні системи використовують:

- Wi-Fi;
- Ethernet;
- BLE;
- ZigBee;

– GSM/4G/5G [1].

Wi-Fi забезпечує високу швидкість передачі та інтеграцію з локальними мережами.

Ethernet використовується у промислових системах завдяки високій стабільності та надійності.

BLE характеризується низьким енергоспоживанням та використовується у мобільних IoT-пристроях.

ZigBee забезпечує побудову mesh-мереж із великою кількістю вузлів.

GSM/4G/5G використовуються у системах дистанційного моніторингу та мобільної телеметрії.

На рисунку 1.3 представлено UML-схему edge computing архітектури, яка демонструє багаторівневу структуру сучасної телекомунікаційної системи.

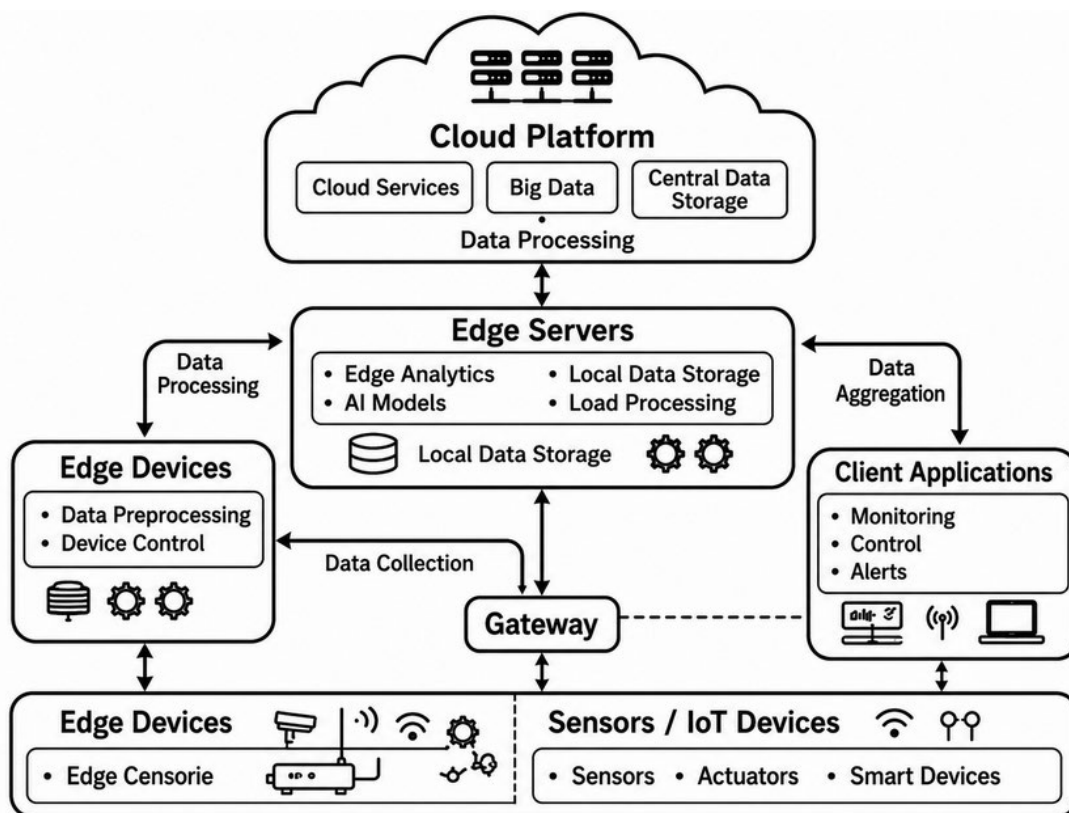


Рисунок 1.3 – UML-схема edge computing архітектури

У нижньому рівні розташовані сенсори та IoT-пристрої, які здійснюють збір даних. Далі інформація передається через шлюз до edge-пристроїв та edge-

серверів, де виконується попередня обробка, фільтрація та аналіз даних. Після цього оброблена інформація надходить до хмарної платформи для централізованого зберігання та аналітики. Клієнтські застосунки забезпечують моніторинг, керування та взаємодію з системою в режимі реального часу.

Отже, в результаті проведеного аналізу встановлено, що сучасні телекомунікаційні системи використовують різні апаратні та програмні рішення залежно від вимог до продуктивності, масштабованості та енергоефективності.

Визначено особливості сучасних IoT-платформ, апаратних платформ та моделей взаємодії між вузлами системи. Встановлено, що найбільш перспективними рішеннями для сучасних embedded-систем є використання ESP32 або STM32 у поєднанні з MQTT-орієнтованими IoT-платформами та edge computing архітектурою.

1.3 Порівняльний аналіз підходів та вибір оптимального рішення

Одним із ключових етапів проектування сучасної телекомунікаційної системи є вибір оптимального поєднання апаратних, програмних та мережевих рішень. Від правильності вибору залежить ефективність функціонування системи, стабільність передавання даних, швидкодія, енергоефективність та можливість масштабування системи в майбутньому [1].

У сучасних embedded-системах використовуються різні підходи до організації обчислювальних ресурсів. Для IoT-систем характерним є поєднання локальної обробки даних із використанням периферійних вузлів та хмарних сервісів. Такий підхід дозволяє мінімізувати навантаження на центральні сервери та скоротити час реакції системи [1].

Одним із найбільш перспективних напрямів розвитку телекомунікаційних систем є використання edge computing. Основна ідея edge computing полягає у виконанні попередньої обробки інформації безпосередньо поблизу джерела даних. Це дозволяє: [1]

- зменшити кількість переданих пакетів;

- скоротити затримки передачі;
- знизити навантаження на мережеву інфраструктуру;
- підвищити автономність системи.

Для телекомунікаційних IoT-систем важливим є забезпечення стабільної роботи навіть при нестабільному або обмеженому каналі зв'язку. У таких умовах доцільним є використання локального буферування інформації та механізмів асинхронного обміну повідомленнями [1].

У сучасних IoT-рішеннях значна увага приділяється масштабованості системи. Масштабованість визначає можливість збільшення кількості підключених вузлів без суттєвого погіршення продуктивності. Для цього використовуються: [1]

- брокерні системи;
- багаторівнева маршрутизація;
- балансування навантаження;
- розподілені обчислення.

Для реалізації телекомунікаційних систем можуть застосовуватись різні підходи до організації взаємодії між вузлами:

- peer-to-peer;
- client-server;
- broker-based architecture.

Peer-to-peer архітектура передбачає прямий обмін інформацією між вузлами системи без використання центрального сервера. Перевагами такого підходу є:

- висока швидкодія;
- мінімальні затримки;
- відсутність центральної точки відмови [1].

Недоліками peer-to-peer є:

- складність масштабування;
- ускладнене керування системою;
- складність синхронізації даних.

Client-server архітектура базується на використанні центрального сервера, який забезпечує оброблення та маршрутизацію даних. Такий підхід забезпечує:

- централізоване адміністрування;
- простоту моніторингу;
- контроль доступу до ресурсів [1].

Проте використання центрального сервера призводить до:

- збільшення затримок;
- перевантаження сервера при великій кількості вузлів;
- залежності системи від працездатності серверного обладнання.

У сучасних IoT-системах найбільш ефективним є використання broker-based architecture, яка базується на брокерній взаємодії між вузлами системи [1].

На рисунку 1.4 наведено порівняння сучасних підходів до організації взаємодії у телекомунікаційних системах.

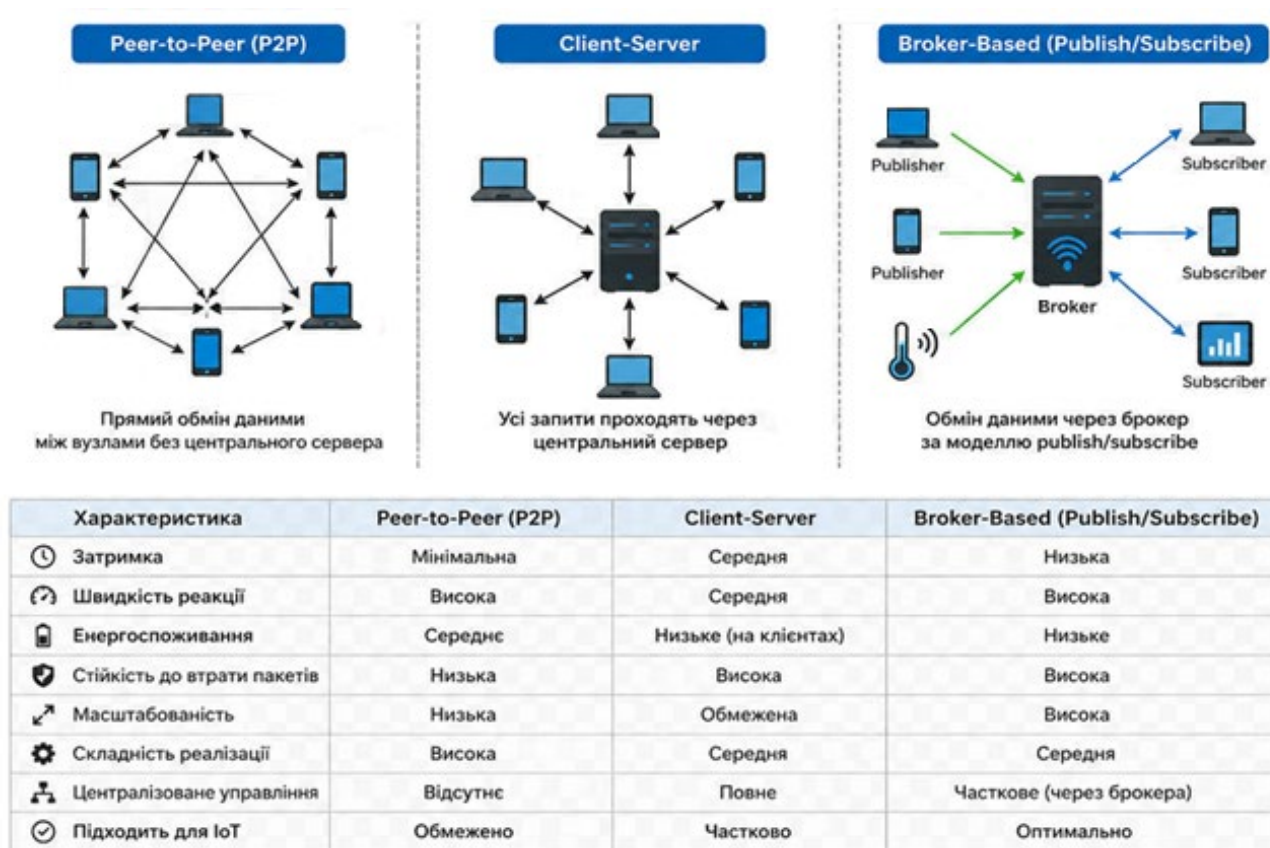


Рисунок 1.4 – Порівняння peer-to-peer, client-server та broker-based архітектури [3]

Брокер забезпечує:

- маршрутизацію повідомлень;
- буферизацію даних;
- підтримку асинхронного обміну;
- масштабування системи.

Для IoT-систем особливого значення набуває вибір способу організації передачі даних. У системах реального часу важливими критеріями є:

- мінімальна затримка;
- висока швидкість реакції;
- низьке енергоспоживання;
- стійкість до втрати пакетів [1].

Для оцінювання ефективності сучасних телекомунікаційних рішень проведемо порівняльний аналіз основних характеристик різних підходів до побудови систем передачі даних, поданих в таблиці 1.5.

Таблиця 1.5 – Порівняльний аналіз архітектур телекомунікаційних систем

Характеристика	Peer-to-peer	Client-server	Broker-based
Централізоване керування	Відсутнє	Наявне	Часткове
Масштабованість	Середня	Обмежена	Висока
Затримка передачі	Мінімальна	Середня	Низька
Складність реалізації	Висока	Середня	Середня
Надійність	Середня	Висока	Висока
Підтримка IoT	Часткова	Обмежена	Висока

Аналіз таблиці 1.5 показує, що broker-based architecture найбільш ефективно відповідає вимогам сучасних IoT-систем.

Такий підхід дозволяє забезпечити масштабованість, підтримку великої кількості вузлів та асинхронний обмін інформацією [1].

Важливим аспектом є також вибір типу мережевої інфраструктури. У сучасних embedded-системах використовуються:

- дротові мережі;
- бездротові мережі;
- гібридні мережі [1].

Дротові мережі забезпечують:

- високу стабільність;
- низький рівень перешкод;
- високу пропускну здатність.

Недоліками дротових систем є:

- складність монтажу;
- обмежена мобільність;
- висока вартість масштабування.

Бездротові мережі забезпечують:

- гнучкість розгортання;
- мобільність пристроїв;
- простоту інтеграції IoT-вузлів [1,2].

Основними недоліками бездротових мереж є:

- залежність від рівня сигналу;
- вплив електромагнітних завад;
- можливість втрати пакетів.

Для сучасних телекомунікаційних систем найбільш ефективним є використання гібридної мережевої архітектури, яка поєднує переваги дротових та бездротових каналів зв'язку [1].

Для побудови IoT-систем важливим критерієм є енергоефективність embedded-пристроїв. Значна кількість IoT-вузлів працює автономно, тому необхідним є мінімізація енергоспоживання мікроконтролерів та мережевих модулів [1,2].

Для зниження енергоспоживання використовуються:

- режими сну;

- пакетна передача даних;
- локальна обробка інформації;
- оптимізація мережеских запитів.

У таблиці 1.6 наведено порівняльний аналіз підходів до організації телекомунікаційних систем.

Таблиця 1.6 – Порівняльний аналіз сучасних підходів до побудови IoT-систем

Критерій	Централізована	Децентралізована	Edge computing
Затримка	Висока	Середня	Низька
Масштабованість	Середня	Висока	Висока
Навантаження на мережу	Високе	Середнє	Низьке
Енергоефективність	Середня	Висока	Висока
Складність реалізації	Низька	Висока	Середня
Робота у реальному часі	Обмежена	Часткова	Ефективна

Проведений аналіз показує, що для сучасних телекомунікаційних IoT-систем найбільш ефективним є використання edge computing архітектури у поєднанні з broker-based моделлю взаємодії між вузлами системи [1].

Такий підхід дозволяє:

- забезпечити високу швидкодію;
- зменшити затримки передачі;
- підвищити масштабованість системи;
- забезпечити стійкість до мережеских відмов;
- знизити енергоспоживання embedded-пристроїв.

Отже, для реалізації вбудованої телекомунікаційної системи передачі даних доцільним є використання гібридної edge computing архітектури з broker-based взаємодією між вузлами системи, що забезпечує оптимальне поєднання

швидкодії, надійності та енергоефективності.

1.4 Формування вимог до розроблюваної системи

Проведений у попередніх підрозділах аналіз сучасних телекомунікаційних систем, IoT-платформ, мережевих архітектур та протоколів передачі даних показав, що сучасні embedded-системи повинні забезпечувати надійний, швидкий та енергоефективний обмін інформацією між великою кількістю пристроїв. Важливою особливістю таких систем є робота в режимі реального часу та підтримка масштабованої мережевої інфраструктури [1], [2].

Для вбудованих телекомунікаційних систем критичними є:

- мінімальна затримка передачі;
- висока достовірність доставки пакетів;
- стабільність роботи;
- підтримка бездротових мереж;
- можливість інтеграції з IoT-платформами;
- низьке енергоспоживання [1].

На основі аналізу предметної області та існуючих рішень встановлено, що проєктована система повинна поєднувати переваги протоколів MQTT, UDP та TCP залежно від типу передаваних даних та режиму функціонування системи [3], [4], [5].

Основною задачею розроблюваної системи є забезпечення ефективної передачі даних між embedded-пристроями, шлюзами та серверною платформою з підтримкою роботи у реальному часі. Система повинна забезпечувати:

- передачу телеметричних даних;
- передавання команд керування;
- моніторинг стану вузлів;
- підтримку віддаленого доступу;
- масштабування кількості пристроїв [1].

З урахуванням проведеного аналізу сформовано основні функціональні

Вимоги до системи.

Основні функціональні вимоги:

1. Підтримка протоколів MQTT, UDP та TCP/IP.
2. Передача даних у режимі реального часу.
3. Підтримка архітектури publish/subscribe.
4. Підтримка клієнт-серверної взаємодії.
5. Можливість роботи у локальній та глобальній мережі.
6. Підтримка Wi-Fi або Ethernet підключення.
7. Підтримка масштабування системи.
8. Можливість інтеграції з хмарними платформами.
9. Забезпечення буферизації та повторної передачі даних.
10. Моніторинг параметрів функціонування системи.

Для забезпечення стабільної роботи система повинна підтримувати одночасну роботу декількох вузлів передачі даних. У процесі функціонування система має забезпечувати:

- прийом даних від сенсорів;
- локальну обробку інформації;
- передачу пакетів у мережу;
- маршрутизацію повідомлень;
- збереження інформації на сервері [1].

Особливу увагу необхідно приділити вимогам до швидкодії системи. Для IoT та embedded-систем важливим є мінімізація часу передачі інформації між вузлами. У системах реального часу допустимі затримки не повинні перевищувати 100 мс [1].

Основні вимоги до швидкодії:

- затримка передачі даних – не більше 100 мс;
- час реакції системи – не більше 200 мс;
- підтримка передачі пакетів у режимі реального часу;
- мінімізація джиттера [1].

У таблиці 1.7 наведено основні технічні вимоги до системи.

Таблиця 1.7 – Основні технічні вимоги до системи

Параметр	Вимога
Тип мережі	Wi-Fi/Ethernet
Протоколи передачі	MQTT/UDP/TCP
Максимальна затримка	≤ 100 мс
Тип архітектури	Broker-based/Client-server
Кількість вузлів	До 100
Режим роботи	Реальний час
Енергоспоживання	Мінімальне
Масштабованість	Підтримується
Інтеграція з IoT	Підтримується

Для забезпечення надійності функціонування система повинна підтримувати:

- контроль доступності вузлів;
- перевірку цілісності пакетів;
- повторну передачу повідомлень;
- буферизацію інформації;
- автоматичне перепідключення до мережі [1].

Для забезпечення енергоефективності embedded-пристроїв система повинна підтримувати режими зниженого енергоспоживання. Це особливо важливо для автономних IoT-вузлів, які працюють від акумуляторного живлення [1].

Основні вимоги до енергоефективності:

- мінімізація кількості мережевих запитів;
- передача даних пакетами;
- підтримка режимів sleep mode;
- локальна обробка даних на edge-вузлах [1,3].

На рисунку 1.5 наведено узагальнену структуру вимог до вбудованої телекомунікаційної системи передачі даних.

Крім технічних характеристик, важливими є вимоги до інформаційної безпеки системи.

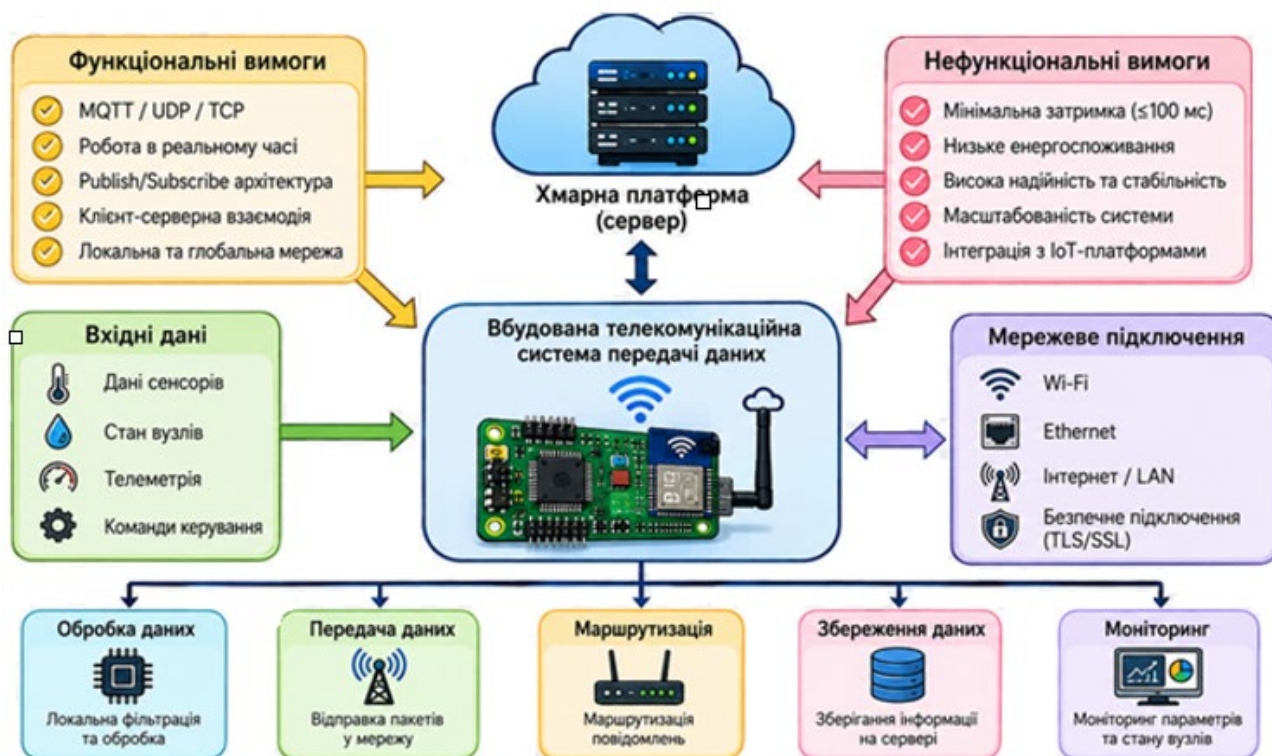


Рисунок 1.5 – Структура вимог до телекомунікаційної системи передачі даних [3]

Оскільки телекомунікаційні системи працюють у мережевому середовищі, необхідно забезпечити:

- захист каналів зв'язку;
- автентифікацію вузлів;
- контроль доступу;
- захист від несанкціонованого підключення;
- резервування каналів передачі [1].

Для захисту інформації доцільним є використання:

- TLS/SSL шифрування;
- VPN-технологій;
- MQTT over TLS;
- механізмів автентифікації користувачів [2], [15].

У результаті проведеного аналізу сформовано узагальнені вимоги до розроблюваної системи передачі даних. Встановлено, що система повинна поєднувати:

- високу швидкість;
- масштабованість;
- енергоефективність;
- підтримку роботи у реальному часі;
- інтеграцію з IoT-платформами;
- підтримку протоколів MQTT, UDP та TCP [3-5].

Таким чином, сформовані вимоги є основою для подальшого схемного проектування, вибору апаратної платформи та реалізації програмного забезпечення вбудованої телекомунікаційної системи передачі даних.

РОЗДІЛ 2

СХЕМНЕ ПРОЄКТУВАННЯ ТА ПРИНЦИП РОБОТИ СИСТЕМИ ПЕРЕДАЧІ ДАНИХ

2.1 Розробка структурної та функціональної схеми системи

Після проведення аналізу сучасних телекомунікаційних систем, IoT-платформ та протоколів передачі даних у першому розділі виникає необхідність розроблення структурної та функціональної схеми embedded-системи передачі даних. Основною метою проєктування є створення телекомунікаційної системи, здатної забезпечити стабільний, енергоефективний та масштабований обмін інформацією між вузлами мережі в режимі реального часу [1].

Сучасні IoT-системи характеризуються багаторівневою архітектурою, у якій окремі вузли виконують функції збору, передавання, маршрутизації, оброблення та збереження інформації.

Основними вимогами до такої системи є:

- мінімальна затримка передачі;
- висока достовірність обміну;
- підтримка масштабування;
- енергоефективність;
- підтримка MQTT, UDP та TCP/IP;
- інтеграція з хмарними сервісами [4].

Для реалізації системи передачі даних доцільним є використання розподіленої архітектури, яка складається з:

- embedded-вузлів;
- мережевого шлюзу;
- MQTT-брокера;
- серверної або хмарної платформи;
- клієнтського інтерфейсу моніторингу [4].

На рисунку 2.1 наведено структурну схему розроблюваної телекомунікаційної системи передачі даних.

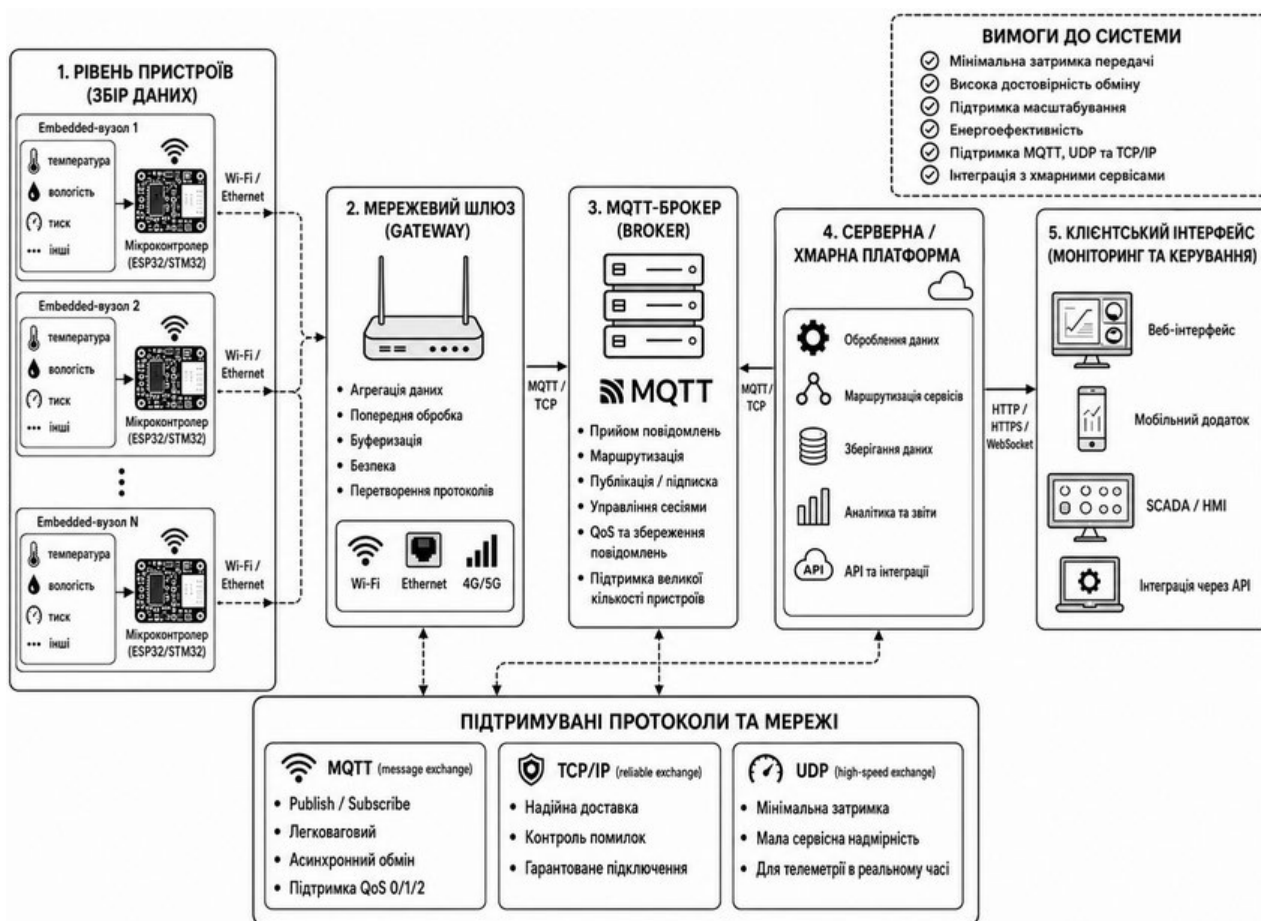


Рисунок 2.1 – Структурна схема телекомунікаційної системи передачі даних [4]

Структурна схема демонструє взаємозв'язок між основними компонентами системи. Embedded-пристрої здійснюють збір інформації із сенсорів та передають дані через Wi-Fi або Ethernet мережу до MQTT-брокера або серверної платформи. Сервер виконує маршрутизацію, оброблення та збереження інформації, а клієнтські застосунки забезпечують моніторинг та керування системою в режимі реального часу [1], [3], [9].

У процесі проєктування особливу увагу приділено вибору мережевої архітектури. Аналіз, проведений у першому розділі, показав, що найбільш ефективною моделлю взаємодії для IoT-систем є broker-based architecture із використанням MQTT.

Основними перевагами такого підходу є:

- асинхронний обмін повідомленнями;
- мінімізація навантаження на вузли;

- підтримка великої кількості пристроїв;
- висока масштабованість;
- підтримка publish/subscribe моделі [1].

Принцип роботи системи полягає у передачі телеметричних даних від embedded-вузлів до MQTT-брокера. Після цього брокер виконує маршрутизацію повідомлень між пристроями, сервером та клієнтськими застосунками [1], [3], [9].

Для забезпечення роботи системи у режимі реального часу використовується комбінований підхід до передачі даних:

- MQTT – для IoT-повідомлень;
- UDP – для швидкісного передавання телеметрії;
- TCP – для надійного обміну службовими даними [4].

У таблиці 2.1 наведено функціональне призначення основних компонентів системи.

Таблиця 2.1 – Основні компоненти системи передачі даних

Компонент	Призначення
Embedded-вузол	Збір та передача даних
Wi-Fi / Ethernet модуль	Мережевий обмін
MQTT-брокер	Маршрутизація повідомлень
Сервер	Оброблення та збереження інформації
Хмарна платформа	Аналітика та моніторинг
Клієнтський застосунок	Візуалізація та керування

Embedded-вузол є основним елементом системи. Він виконує:

- опитування сенсорів;
- формування пакетів даних;
- локальну обробку інформації;
- передавання повідомлень у мережу [5].

У якості апаратної платформи доцільним є використання ESP32 або

STM32, оскільки вони підтримують:

- Wi-Fi;
- TCP/IP стек;
- MQTT-бібліотеки;
- FreeRTOS;
- низьке енергоспоживання [5].

На рисунку 2.2 наведено функціональну схему embedded-вузла системи передачі даних.

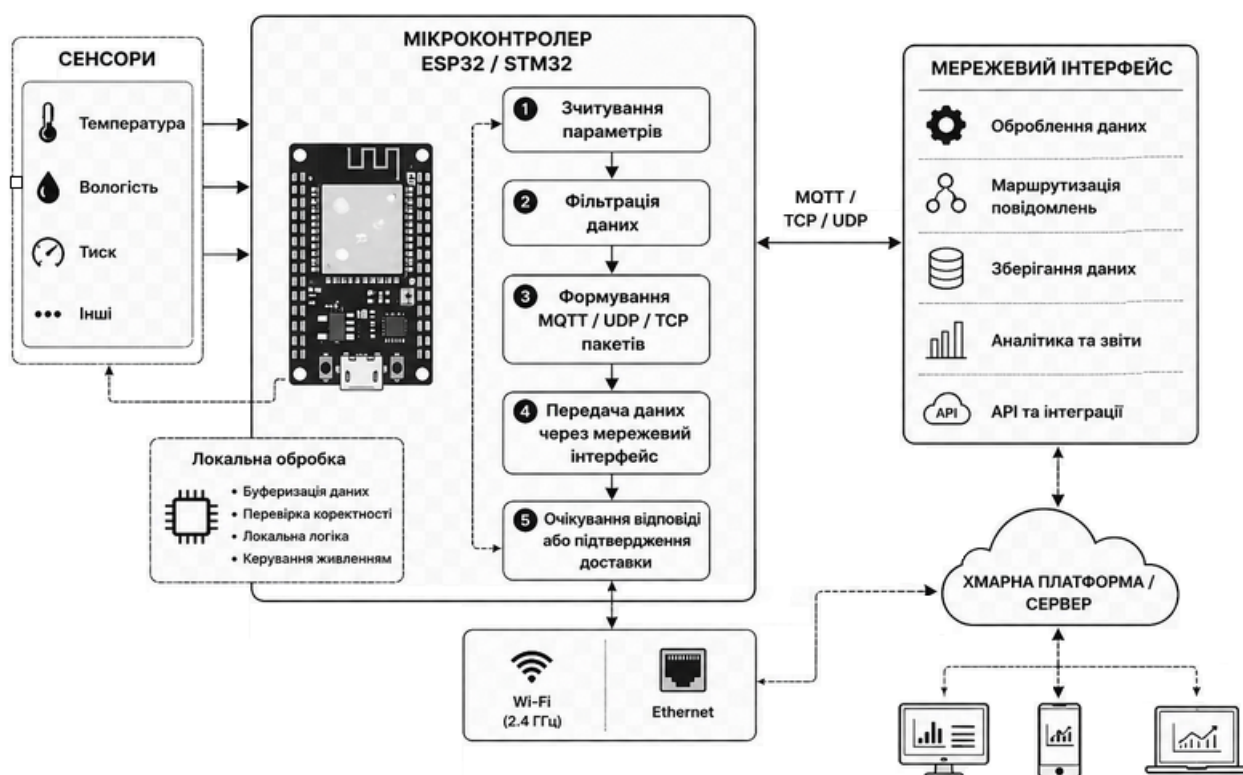


Рисунок 2.2 – Функціональна схема embedded-вузла [5]

Функціональна схема демонструє послідовність оброблення інформації в системі. Після отримання даних від сенсорів мікроконтролер виконує:

- зчитування параметрів;
- фільтрацію даних;
- формування MQTT/UDP/TCP пакетів;
- передавання через мережевий інтерфейс;
- очікування відповіді або підтвердження доставки [4].

Для забезпечення стабільної роботи системи використовується MQTT-брокер, який виконує:

- буферизацію повідомлень;
- маршрутизацію пакетів;
- підтримку підписки на топіс;
- балансування навантаження [6].

На рисунку 2.3 наведено UML-схему взаємодії компонентів телекомунікаційної системи.

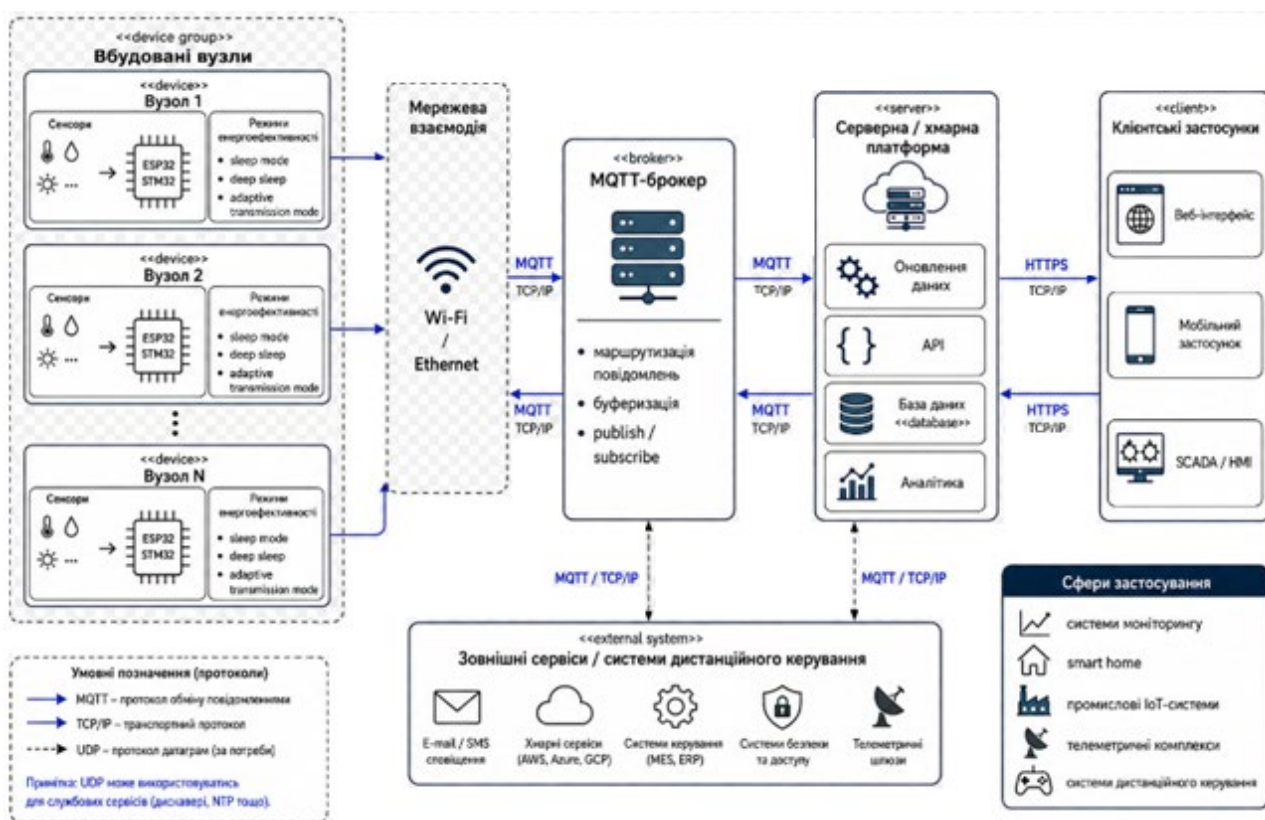


Рисунок 2.3 – UML-схема взаємодії компонентів системи [6]

Система підтримує одночасну роботу великої кількості вузлів, що дозволяє використовувати її у:

- системах моніторингу;
- smart home;
- промислових IoT-системах;
- телеметричних комплексах;

– системах дистанційного керування [1].

Для забезпечення енергоефективності embedded-вузли можуть працювати у режимах:

- sleep mode;
- deep sleep;
- adaptive transmission mode [1,7].

У процесі проектування особлива увага приділяється забезпеченню надійності передачі даних. Для цього використовуються:

- QoS-рівні MQTT;
- механізми повторної передачі TCP;
- буферизація пакетів;
- контроль CRC [7].

На рисунку 2.4 наведено принцип організації MQTT-взаємодії між вузлами системи.

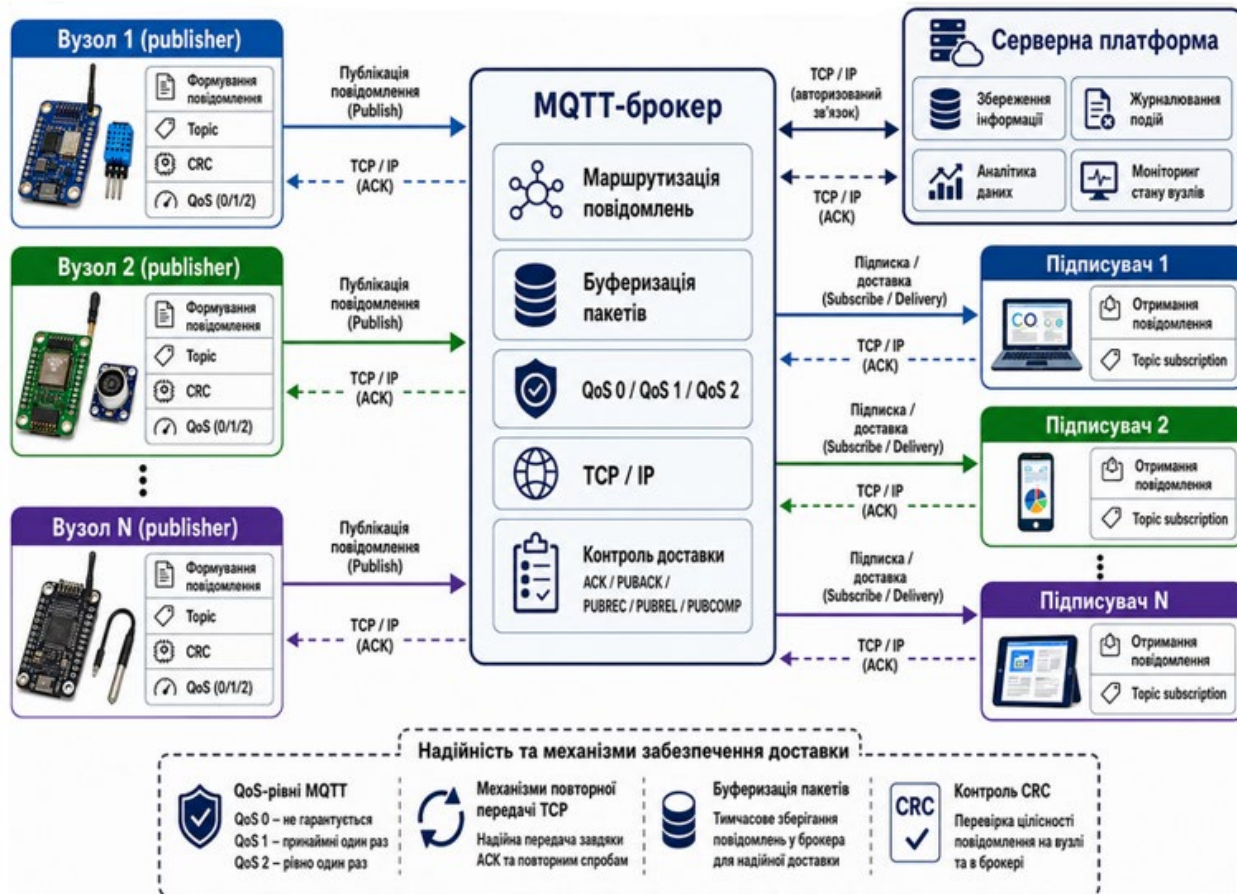


Рисунок 2.4 – Організація MQTT-взаємодії між вузлами [7]

У системі використовується publish/subscribe модель, у якій:

- publisher формує повідомлення;
- MQTT-брокер виконує маршрутизацію;
- subscriber отримує інформацію за відповідним topic [7].

Перевагами такої архітектури є:

- незалежність вузлів;
- масштабованість;
- простота інтеграції;
- мінімізація мережевого навантаження [1,7].

У процесі функціонування системи серверна платформа забезпечує:

- збереження інформації;
- журналювання подій;
- аналітику даних;
- моніторинг стану вузлів [8].

Для підвищення продуктивності системи використовується edge computing архітектура, яка дозволяє виконувати попередню обробку інформації безпосередньо на периферійних вузлах [1].

На рисунку 2.5 наведено структурну схему edge computing системи передачі даних.

Використання edge computing дозволяє:

- зменшити затримки;
- знизити навантаження на сервер;
- скоротити обсяг передаваних даних;
- підвищити швидкодію системи [1,7].

У результаті виконаного проектування сформовано структурну та функціональну схеми телекомунікаційної системи передачі даних. Розроблена архітектура забезпечує:

- підтримку MQTT, UDP та TCP;
- роботу в режимі реального часу;
- масштабованість;

- енергоефективність;
- інтеграцію з IoT-платформами;
- підтримку хмарних сервісів.

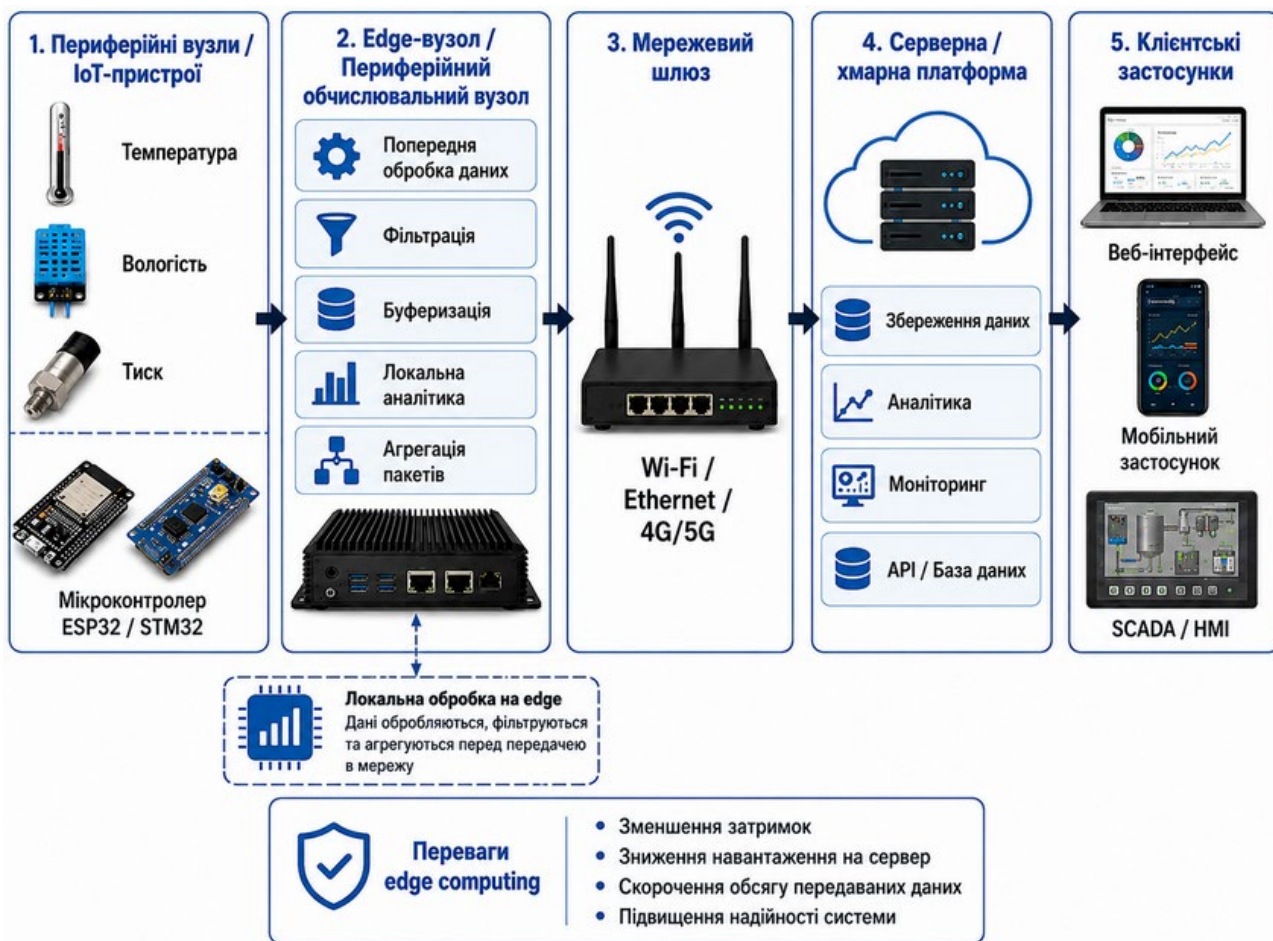


Рисунок 2.5 – Структура edge computing системи передачі даних [8]

Отримані результати є основою для подальшого розроблення принципової електричної схеми, програмного забезпечення та алгоритмів функціонування системи.

2.2 Розробка принципової електричної схеми в EasyEda

Принципова електрична схема є основним конструкторським документом, який відображає електричні з'єднання між функціональними вузлами системи передачі даних. На відміну від структурної та функціональної схем, принципова

схема показує конкретні електронні компоненти, їх виводи, лінії живлення, сигнальні з'єднання, інтерфейси обміну даними та допоміжні елементи. Для розроблюваної вбудованої телекомунікаційної системи принципова схема повинна забезпечувати підключення мікроконтролера, сенсорів, мережевого інтерфейсу, модуля живлення, USB-UART перетворювача та елементів індикації. Відповідно до вихідних даних бакалаврської роботи, апаратна платформа системи передбачає використання ESP32/STM32/Raspberry Pi, мережевих протоколів MQTT, UDP, TCP/IP та мережі Wi-Fi/Ethernet.

Для побудови принципової електричної схеми використано середовище EasyEDA, яке дозволяє виконувати схемотехнічне проєктування, створювати бібліотечні компоненти, формувати електричні з'єднання, перевіряти схему та готувати її до подальшого трасування друкованої плати. Документація EasyEDA вказує, що редактор може використовуватись для створення професійних електричних схем і власних умовних графічних позначень компонентів.

У розробленій схемі центральним елементом є мікроконтролер ESP32, який поєднує обчислювальний модуль, цифрові інтерфейси введення-виведення та вбудований бездротовий інтерфейс Wi-Fi. Компанія Espressif позиціонує свої мікроконтролерні рішення як апаратну базу для IoT-пристроїв, бездротового зв'язку та інтелектуальних мережевих систем. Завдяки наявності Wi-Fi ESP32 може безпосередньо передавати дані до MQTT-брокера або серверної платформи без додаткового бездротового модуля.

Для забезпечення резервного або альтернативного дротового підключення до локальної мережі у схемі передбачено модуль W5500 Ethernet, який взаємодіє з ESP32 через інтерфейс SPI. WIZnet зазначає, що рішення W5500 належить до мережевих контролерів з апаратною підтримкою TCP/IP, що дозволяє використовувати його для побудови Ethernet-підключення мікроконтролерних систем.

На рисунку 2.6 наведено принципову електричну схему вбудованої системи передачі даних, розроблену в EasyEDA [9].

На даному рисунку показано з'єднання мікроконтролера ESP32 із

сенсорними модулями, стабілізатором живлення, USB-UART перетворювачем, Ethernet-модулем W5500, інтерфейсними лініями SPI/I2C/UART, кнопкою RESET та світлодіодом індикації.

Така структура дозволяє реалізувати збір даних із сенсорів, первинну обробку інформації та передачу даних через Wi-Fi або Ethernet.

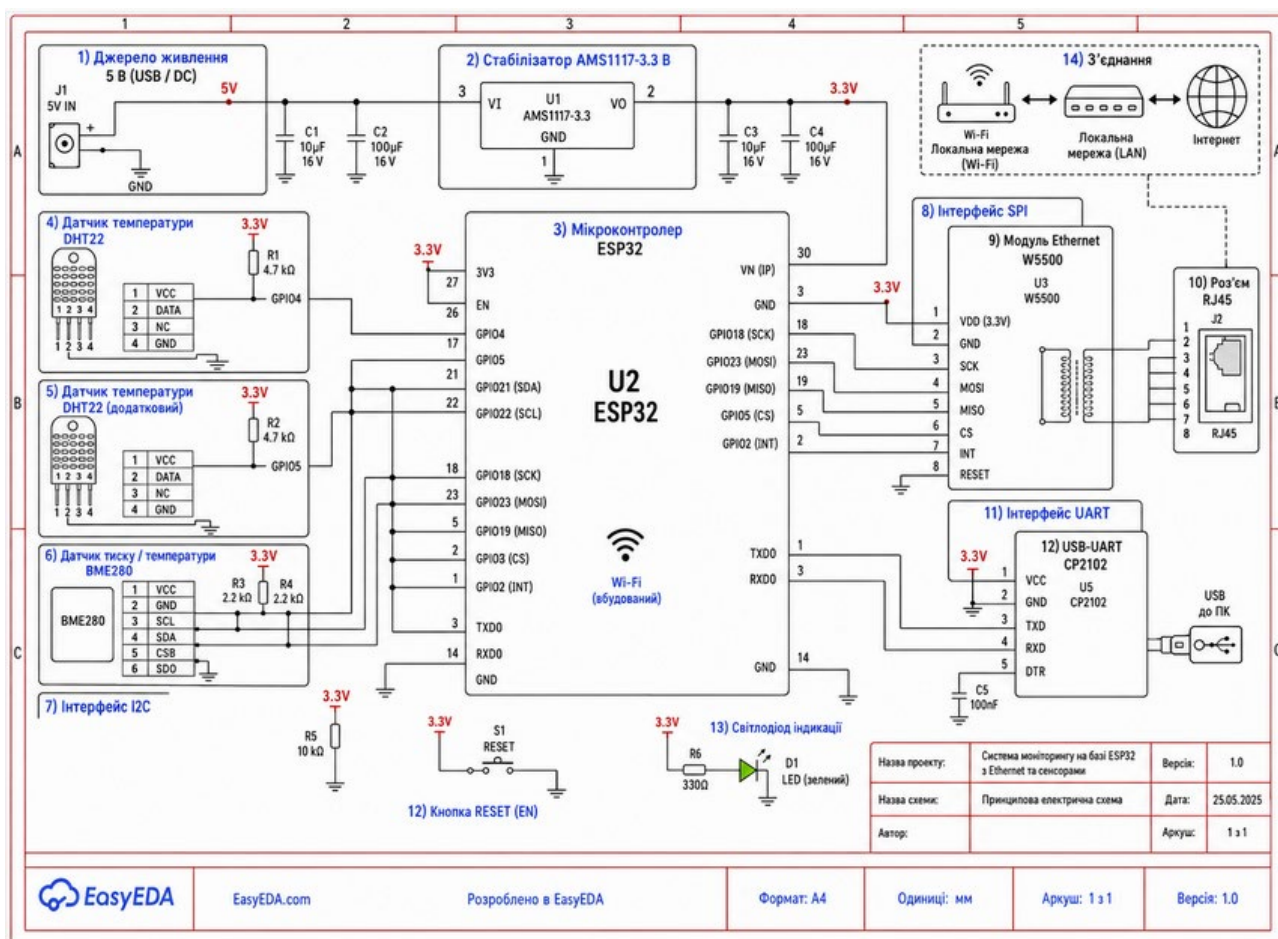


Рисунок 2.6 – Принципова електрична схема вбудованої системи передачі даних [9]

До складу принципової електричної схеми входять такі основні вузли:

- вузол живлення 5 В / 3,3 В;
- мікроконтролер ESP32;
- сенсорний блок;
- Ethernet-модуль W5500;
- USB-UART перетворювач CP2102;

- інтерфейси I2C, SPI та UART;
- кнопка скидання RESET;
- світлодіод індикації стану.

Основні компоненти, використані у схемі, подано в таблиці 2.2.

Для сенсорного блоку доцільно використовувати цифрові сенсори температури, вологості та тиску.

Наприклад, BME280 є комбінованим цифровим сенсором температури, вологості та барометричного тиску з підтримкою малих габаритів і низького енергоспоживання, що робить його придатним для IoT-пристроїв.

Таблиця 2.2 – Основні компоненти принципової електричної схеми

Компонент	Призначення в системі	Основний інтерфейс
ESP32	Центральний мікроконтролер, оброблення даних, Wi-Fi-зв'язок	GPIO, I2C, SPI, UART
AMS1117-3.3	Стабілізація напруги 5 В до 3,3 В	Живлення
BME280	Вимірювання температури, вологості та тиску	I2C
DHT22	Вимірювання температури та вологості	GPIO
W5500 Ethernet	Дротове підключення до локальної мережі	SPI
RJ45	Фізичне підключення Ethernet-кабелю	Ethernet
CP2102	Обмін даними з ПК та програмування через USB-UART	UART
LED	Індикація стану роботи системи	GPIO
RESET	Скидання мікроконтролера	EN / RESET

Живлення системи здійснюється від джерела постійної напруги 5 В, наприклад USB або зовнішнього DC-джерела. Оскільки ESP32, BME280 та W5500 працюють із логічними рівнями 3,3 В, у схемі використовується

стабілізатор AMS1117-3.3, який формує стабілізовану напругу 3,3 В для живлення цифрових компонентів.

До входу та виходу стабілізатора підключаються фільтрувальні конденсатори. Їх завдання полягає у зменшенні пульсацій напруги, підвищенні стабільності живлення та зменшенні впливу імпульсних завад на роботу мікроконтролера і мережевого інтерфейсу.

Усі цифрові компоненти схеми повинні мати спільну шину GND, оскільки відсутність спільного потенціалу може призвести до нестабільної роботи інтерфейсів I2C, SPI та UART.

Мікроконтролер ESP32 виконує функції центрального керуючого елемента. Він зчитує дані із сенсорів, формує пакети даних, виконує попередню обробку інформації та передає повідомлення через Wi-Fi або Ethernet. У схемі ESP32 підключений до сенсорів через GPIO та I2C, до Ethernet-модуля через SPI, а до USB-UART-перетворювача через UART.

Для коректної роботи цифрових інтерфейсів у схемі використовуються підтягувальні резистори на лініях даних, зокрема для сенсорів із відкритим колектором або інтерфейсом I2C. Це забезпечує стабільні логічні рівні та зменшує ймовірність помилок під час обміну даними [10,11].

Сенсорний блок забезпечує збір параметрів навколишнього середовища. У схемі передбачено підключення сенсорів температури, вологості та тиску. DHT22 може використовуватися для вимірювання температури й вологості, а BME280 – для комплексного вимірювання температури, вологості та атмосферного тиску. BME280 підтримує цифровий обмін і може працювати через I2C або SPI, що спрощує інтеграцію з ESP32.

Для підключення сенсорів у схемі використовуються такі сигнали:

- VCC – живлення сенсора;
- GND – загальний провід;
- DATA – цифровий вихід DHT22;
- SDA – лінія даних I2C;
- SCL – лінія тактування I2C.

Використання декількох сенсорів дозволяє розширити функціональні можливості системи та перевірити працездатність програмного забезпечення при обробленні різних типів даних.

Для підвищення надійності мережевого з'єднання у схемі передбачено Ethernet-модуль W5500. Він підключається до ESP32 через SPI-інтерфейс. Такий підхід дозволяє реалізувати дротовий канал зв'язку у випадках, коли Wi-Fi-з'єднання є нестабільним або небажаним через завади. W5500 підтримує апаратну реалізацію TCP/IP, що зменшує навантаження на мікроконтролер під час оброблення мережевих пакетів. WIZnet позиціонує свої рішення як апаратні TCP/IP Offload Engine для IoT та мікроконтролерних застосувань.

Зовнішній роз'єм RJ45 забезпечує фізичне підключення до локальної мережі Ethernet. У схемі також передбачено умовне підключення до локальної мережі та Інтернету.

Для програмування ESP32, налагодження програмного забезпечення та виведення діагностичних повідомлень використовується USB-UART перетворювач CP2102. Він забезпечує обмін даними між комп'ютером і мікроконтролером через послідовний інтерфейс UART.

До основних сигналів USB-UART належать:

- TXD – передача даних;
- RXD – приймання даних;
- DTR – керування режимом завантаження;
- GND – спільний провід;
- VCC – живлення або логічний рівень інтерфейсу.

Під час розробки та тестування системи UART-інтерфейс використовується для моніторингу повідомлень про стан підключення до Wi-Fi, MQTT-брокера, Ethernet-мережі та для відображення поточних значень сенсорів [13].

У схемі передбачено світлодіод індикації, який підключений через обмежувальний резистор. Він може використовуватись для відображення стану системи:

- живлення подано;
- мережа підключена;
- дані передаються;
- виникла помилка з'єднання.

Кнопка RESET підключена до виводу EN мікроконтролера та дозволяє виконати апаратне скидання системи. Для стабільної роботи лінії EN використовується підтягувальний резистор.

Під час побудови схеми в EasyEDA було виконано такі етапи:

- створення нового проєкту;
- вибір бібліотечних компонентів ESP32, W5500, CP2102, BME280, DHT22;
- розміщення компонентів на робочому полі;
- формування шин живлення 5 В, 3,3 В та GND;
- підключення цифрових інтерфейсів I2C, SPI та UART;
- додавання конденсаторів фільтрації, резисторів підтягування, кнопки RESET та LED;
- перевірка правильності електричних з'єднань;
- підготовка схеми до подальшого трасування друкованої плати.

Для контролю правильності схеми в EasyEDA доцільно використовувати Design Manager та електричну перевірку схеми, оскільки ці інструменти допомагають виявляти помилки підключення компонентів і неправильні з'єднання виводів [15].

Отже, у підрозділі розроблено принципову електричну схему вбудованої телекомунікаційної системи передачі даних у середовищі EasyEDA. Основою схеми є мікроконтролер ESP32, який забезпечує збір та оброблення даних, а також передачу інформації через Wi-Fi або Ethernet. Для підключення сенсорів використано цифрові інтерфейси GPIO та I2C, для Ethernet-з'єднання – модуль W5500 через SPI, а для програмування і налагодження – USB-UART перетворювач CP2102. Розроблена схема забезпечує апаратну основу для реалізації MQTT, UDP та TCP/IP обміну даними в embedded IoT-системі.

2.3 Опис роботи основних вузлів та їх взаємодії

Розроблювана система передачі даних складається з кількох взаємопов'язаних вузлів: вузла живлення, мікроконтролерного ядра ESP32, сенсорного блоку, інтерфейсів I2C, SPI та UART, Ethernet-модуля W5500, USB-UART перетворювача CP2102, вузла скидання та індикації. Їх взаємодія забезпечує повний цикл роботи пристрою: від надходження живлення до формування пакета даних і передавання інформації у мережу.

Розглянемо послідовність запуску системи. Після подачі напруги 5 В на вхід схеми спочатку активується вузол живлення. Стабілізатор AMS1117-3.3 формує напругу 3,3 В, яка використовується для живлення мікроконтролера ESP32, сенсорних модулів, Ethernet-контролера W5500 та логічної частини USB-UART перетворювача. На цьому етапі фільтрувальні конденсатори зменшують пульсації напруги та короткочасні імпульсні завади, що можуть виникати під час підключення живлення або запуску мережевих модулів.

Після стабілізації живлення мікроконтролер ESP32 переходить у режим запуску. Згідно з технічною документацією Espressif, ESP32 містить багатий набір периферійних інтерфейсів, зокрема SPI, UART, I2C, ADC, DAC та інші модулі, що дозволяє використовувати його як центральний вузол embedded-системи. У межах цієї роботи використовуються саме ті інтерфейси, які потрібні для обміну із сенсорами, Ethernet-модулем та комп'ютером розробника. У таблиці 2.3 подано початкову послідовність активації вузлів системи.

Розглянемо роботу мікроконтролерного вузла ESP32. ESP32 виконує функцію координатора роботи всієї системи. Після запуску він налаштовує виводи загального призначення, ініціалізує шину I2C для роботи із сенсором BME280, готує SPI-інтерфейс для взаємодії з Ethernet-модулем W5500 та активує UART-канал для налагодження [14].

Окремо виконується перевірка стану Wi-Fi, оскільки ESP32 має вбудований бездротовий інтерфейс і може передавати дані без зовнішнього Wi-Fi-модуля.

Спочатку він перевіряє доступність сенсорів, потім зчитує значення, виконує первинну перевірку коректності, формує службову структуру даних і лише після цього передає пакет у мережу.

Типовий пакет даних, який формується мікроконтролером, може містити:

- ідентифікатор вузла;
- часову мітку;
- значення температури;
- значення вологості
- значення тиску;
- номер пакета;
- службовий код стану;
- контрольну суму.

Таблиця 2.3 – Початкова послідовність активації вузлів системи

Етап	Активний вузол	Основна дія	Результат
1	Джерело живлення	Подача 5 В	Живлення надходить на стабілізатор
2	AMS1117-3,3	Формування 3,3 В	Живлення цифрових вузлів
3	ESP32	Початковий запуск	Ініціалізація ядра та портів
4	GPIO/I2C/SPI/UART	Налаштування інтерфейсів	Підготовка до обміну
5	Сенсори	Перевірка доступності	Готовність до зчитування
6	Wi-Fi/W5500	Перевірка мережевого каналу	Вибір способу передавання

Даний підхід дозволяє не лише передати виміряні значення, а й контролювати стан вузла, відстежувати послідовність пакетів та виявляти можливі помилки обміну.

Сенсорний блок відповідає за отримання первинної вимірювальної інформації. У розроблюваній системі передбачено використання датчиків температури, вологості та тиску. DHT22 може використовуватися як базовий сенсор температури й вологості, а BME280 – як комбінований сенсор температури, вологості та атмосферного тиску. За даними Bosch Sensortec, BME280 є цифровим сенсором вологості, тиску й температури, орієнтованим на малогабаритні пристрої з низьким енергоспоживанням [15].

Взаємодія сенсорного блоку з ESP32 виконується у два способи. Для DHT22 використовується окрема цифрова лінія GPIO, через яку мікроконтролер запускає обмін і приймає послідовність імпульсів із результатами вимірювання. Для BME280 застосовується I2C-шина, що дозволяє передавати дані через дві лінії – SDA та SCL. Після отримання інформації ESP32 виконує перевірку діапазону значень. Наприклад, якщо сенсор не відповідає або повертає некоректне значення, мікроконтролер може повторити запит, сформував службове повідомлення про помилку або передати попереднє коректне значення із відповідною позначкою статусу.

I2C-шина використовується для сенсорів, які підтримують адресний цифровий обмін. Її перевагою є можливість підключення декількох пристроїв до однієї пари сигнальних ліній. У розроблюваній системі ця шина використовується для сенсора BME280. Мікроконтролер ESP32 виступає головним пристроєм, формує адресний запит, приймає дані та виконує перетворення отриманих регістрів у фізичні величини.

Практичне значення такого підходу полягає в тому, що сенсорний блок можна розширювати без суттєвої зміни принципової схеми. Наприклад, до тієї ж I2C-шини можуть бути додані інші цифрові сенсори за умови відсутності конфлікту адрес. Це робить систему придатною для подальшої модернізації.

Взаємодія через SPI та робота Ethernet-модуля W5500. Для дротового підключення до локальної мережі використовується Ethernet-модуль W5500. Його взаємодія з ESP32 відбувається через SPI-інтерфейс. WIZnet описує W5500 як Ethernet-контролер із апаратним стеком TCP/IP, що забезпечує підключення

зовнішнього мікроконтролера до мережі через SPI; модуль також підтримує TCP, UDP та IPv4 .

У процесі роботи ESP32 передає до W5500 службові команди, параметри мережевого з'єднання та сформовані пакети даних. Ethernet-модуль, у свою чергу, забезпечує фізичний обмін через роз'єм RJ45. Такий варіант взаємодії доцільний для стаціонарних систем, де потрібне стабільне підключення до локальної мережі та менша залежність від якості Wi-Fi-сигналу.

SPI-взаємодія відбувається через лінії SCK, MOSI, MISO та CS. Лінія SCK задає тактування, MOSI використовується для передачі даних від ESP32 до W5500, MISO – для зворотної передачі, а CS обирає активний пристрій на SPI-шині. Додатковий сигнал INT може використовуватися для повідомлення ESP32 про події в Ethernet-модулі, наприклад отримання пакета або зміну стану з'єднання.

Розглянемо USB-UART взаємодію та діагностику. USB-UART перетворювач CP2102 використовується для зв'язку ESP32 з персональним комп'ютером. Silicon Labs описує CP2102N як USB-to-UART bridge controller, який дозволяє швидко додати USB-підключення до пристрою без складної розробки власного USB-стеку [9,10].

У розроблюваній системі CP2102 виконує дві основні функції. Перша – завантаження програмного коду в мікроконтролер. Друга – діагностика роботи системи через послідовний монітор. Під час тестування через UART можна виводити повідомлення про запуск системи, стан Wi-Fi, результат підключення до MQTT-брокера, IP-адресу пристрою, показники сенсорів та повідомлення про помилки.

Цей вузол не бере безпосередньої участі у штатній передачі телеметрії в мережу, однак є важливим на етапах розробки, перевірки та сервісного обслуговування системи.

Вузол індикації призначений для швидкого візуального контролю стану пристрою без підключення до комп'ютера. Світлодіод може використовуватись для відображення кількох станів: живлення подано, мережа підключена, пакет

передається, виникла помилка сенсора або немає доступу до сервера.

Кнопка RESET підключається до лінії EN мікроконтролера та дозволяє виконати апаратне перезавантаження системи. Це важливо під час налагодження, коли необхідно швидко повторити цикл ініціалізації або перевірити поведінку системи після перезапуску.

Узагальнимо логіку взаємодії вузлів. На рисунку 2.7 доцільно подати не повтор принципової схеми, а саме логічну послідовність взаємодії вузлів: живлення → запуск ESP32 → зчитування сенсорів → формування пакета → вибір мережевого каналу → передача даних → діагностика стану.

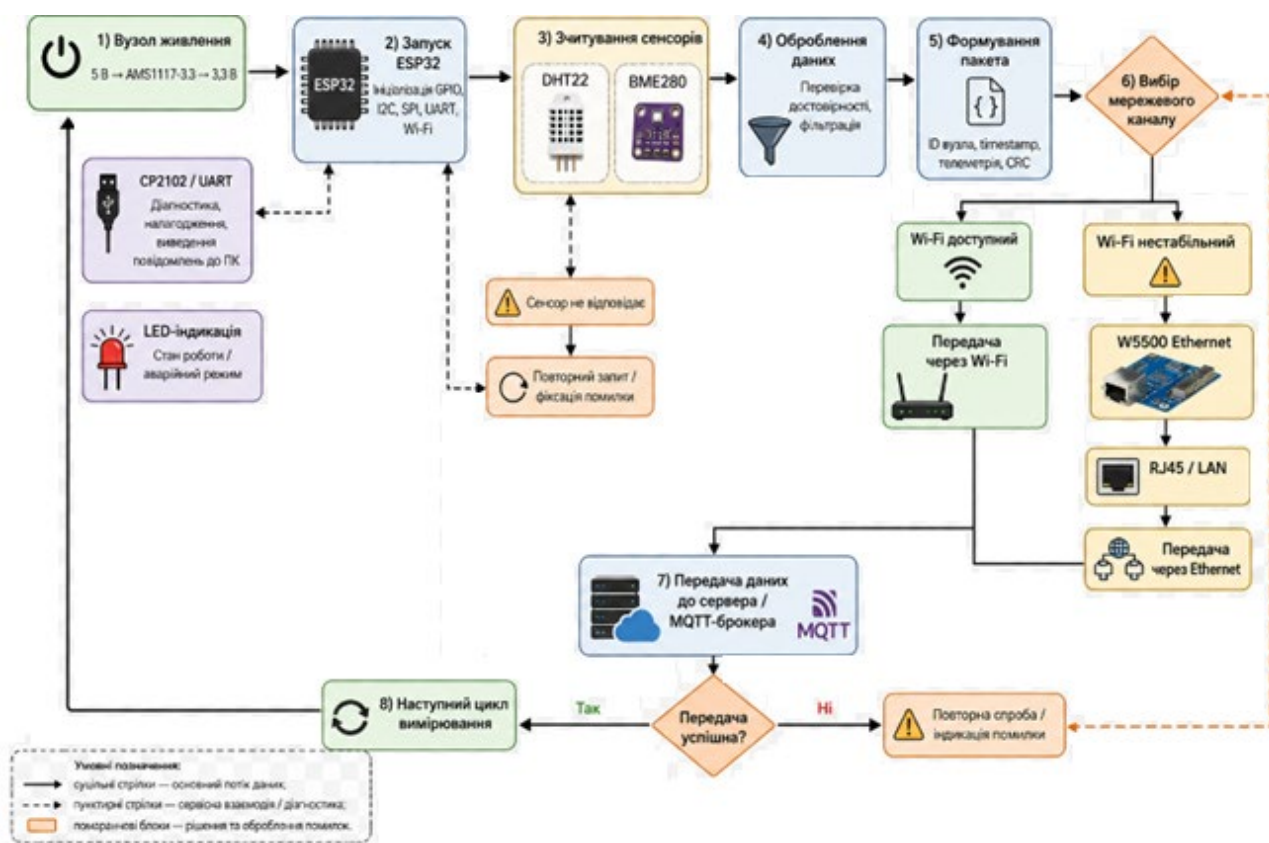


Рисунок 2.7 – Послідовність взаємодії основних вузлів системи передачі даних [15]

Дана схема демонструє, що після стабілізації живлення ESP32 ініціалізує інтерфейси, зчитує дані з сенсорного блоку, формує пакет телеметрії та передає його через Wi-Fi або W5500 Ethernet. USB-UART використовується для налагодження, а вузол індикації повідомляє про стан роботи пристрою.

Під час штатної роботи система функціонує циклічно. Після кожного вимірювання ESP32 перевіряє достовірність отриманих даних, формує пакет, обирає активний мережевий інтерфейс і передає інформацію. У випадку успішної передачі система переходить до наступного циклу вимірювання. Якщо виникає помилка, наприклад відсутність відповіді сенсора або недоступність мережі, мікроконтролер виконує повторну спробу, фіксує стан помилки та може відобразити її за допомогою світлодіода [16].

Важливо, що взаємодія вузлів організована таким чином, щоб кожен модуль виконував окрему функцію: сенсори відповідають за вимірювання, ESP32 – за оброблення та керування, W5500 – за дротовий мережевий доступ, CP2102 – за сервісний обмін із ПК, а вузол живлення – за стабільну роботу всієї електронної частини.

В таблиці 2.4 подано сценарії роботи системи залежно від стану вузлів.

Таблиця 2.4 – Сценарії роботи системи залежно від стану вузлів

Ситуація	Дія ESP32	Задіяні вузли	Очікуваний результат
Живлення стабільне	Запуск основного циклу	AMS1117, ESP32	Система переходить до ініціалізації
Сенсор відповідає	Зчитування та оброблення	DHT22 / BME280, ESP32	Формується коректний пакет
Сенсор не відповідає	Повторний запит	ESP32, сенсорний блок	Повторне зчитування або фіксація помилки
Wi-Fi доступний	Передача бездротовим каналом	ESP32, Wi-Fi	Пакет надсилається в мережу
Wi-Fi нестабільний	Перехід на Ethernet	ESP32, W5500, RJ45	Дані передаються дротовим каналом

Отже, у підрозділі описано роботу основних вузлів вбудованої телекомунікаційної системи та їх взаємодію під час повного циклу передачі даних. На відміну від підрозділів, де розглянуто структурну, функціональну та

принципову схеми, тут акцент зроблено на послідовності роботи апаратних модулів: стабілізації живлення, запуску ESP32, опитуванні сенсорів, роботі інтерфейсів I2C, SPI та UART, виборі мережевого каналу Wi-Fi / Ethernet, діагностиці через CP2102 та індикації стану. Така організація взаємодії дозволяє реалізувати надійну систему збору, оброблення та передачі даних

РОЗДІЛ 3

РОЗРАХУНКОВА АПАРАТНО-ПРОГРАМНА ЧАСТИНА

3.1 Розрахунок параметрів передачі даних

У сучасних вбудованих телекомунікаційних системах одним із ключових етапів проєктування є розрахунок параметрів передачі даних, оскільки саме від характеристик мережевого обміну залежить швидкодія, стабільність та ефективність функціонування IoT-системи. У розроблюваній системі передавання інформації здійснюється між embedded-вузлами ESP32/STM32, MQTT-брокером та серверною платформою через Wi-Fi або Ethernet мережу [1], [3], [9].

Основними параметрами передачі даних у телекомунікаційній системі є:

- швидкість передачі;
- затримка передавання;
- джиттер;
- пропускна здатність;
- коефіцієнт втрат пакетів;
- навантаження на мережу [1–15].

Для оцінювання ефективності системи приймемо такі параметри:

- тип мережі – Wi-Fi IEEE 802.11n;
- швидкість каналу – 10 Мбіт/с;
- тип повідомлень – MQTT / UDP / TCP;
- середній розмір пакета – 512 байт;
- кількість embedded-вузлів – 10.

На рисунку 3.1 наведено графік залежності часу передачі від розміру пакета даних.

Час передачі пакета визначається за формулою (3.1):

$$T_{tx}=L/B, \quad (3.1)$$

де T_{tx} – час передачі пакета, с;

L – розмір пакета, біт;

B – швидкість каналу передачі, біт/с.

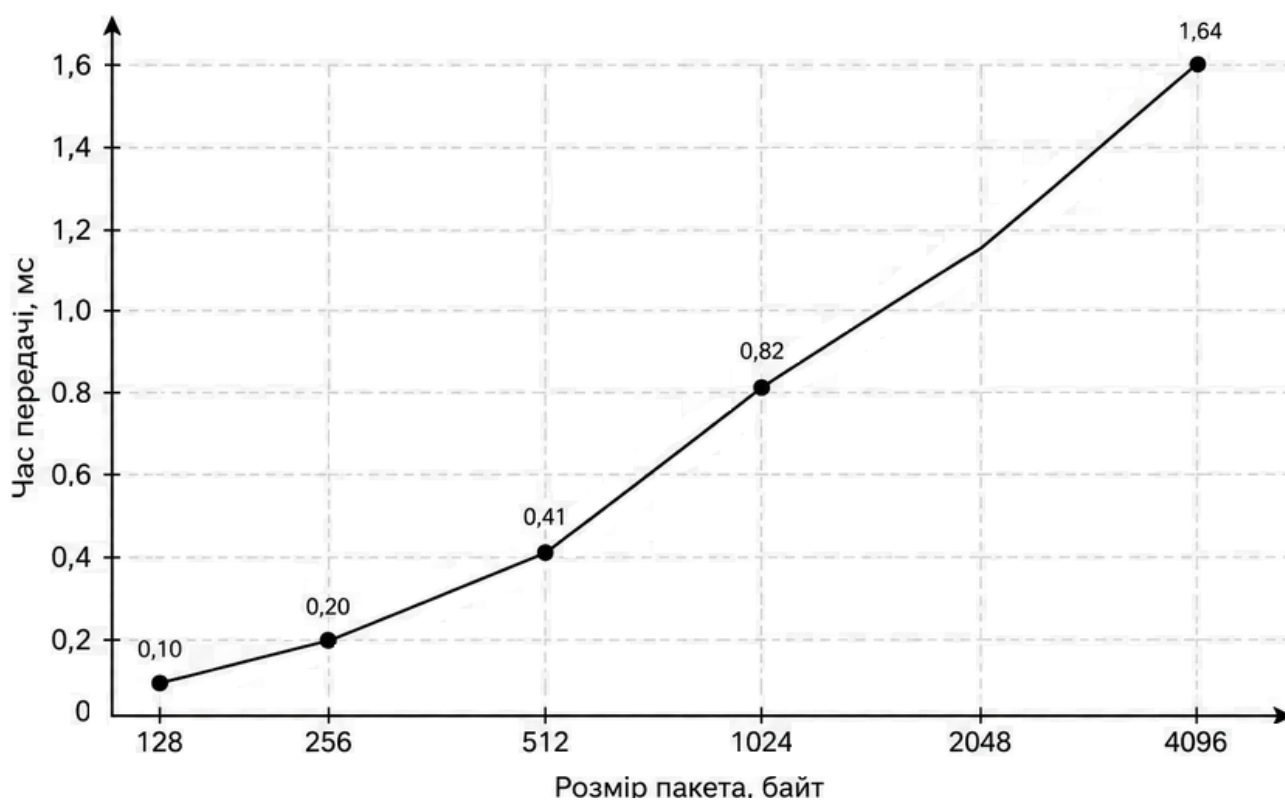


Рисунок 3.1 – Залежність часу передачі від розміру пакета даних

Для пакета розміром 512 байт: $L=512 \cdot 8=4096$ біт.

При швидкості каналу 10 Мбіт/с: $T_{\text{тх}}=4096/(10 \cdot 10^6)=0,0004096 \text{ с} \approx 0,41 \text{ мс}$.

Отриманий результат свідчить про можливість швидкої передачі телеметричних повідомлень у системі реального часу [1].

Одним із важливих параметрів телекомунікаційної системи є затримка передачі даних.

Загальна затримка формується із:

- часу оброблення повідомлення мікроконтролером;
- часу передачі мережею;
- часу маршрутизації;
- часу серверного оброблення [1].

Сумарна затримка визначається за формулою (3.2):

$$T_{\text{sum}} = T_{\text{proc}} + T_{\text{tx}} + T_{\text{route}} + T_{\text{server}}, \quad (3.2)$$

де T_{proc} – час локального оброблення;

T_{route} – час маршрутизації;

T_{server} – час серверної обробки.

Для ESP32:

– локальна обробка – 3 мс;

– маршрутизація – 5 мс;

– серверне оброблення – 7 мс.

Тоді: $T_{\text{sum}} = 3 + 0,41 + 5 + 7 = 15,41$ мс.

Отримане значення відповідає вимогам систем реального часу, де допустима затримка передачі становить менше 100 мс [1].

Для оцінювання навантаження на мережу визначимо пропускну здатність системи згідно (3.3):

$$C = (N \cdot L) / T, \quad (3.3)$$

де C – пропускна здатність системи;

N – кількість пакетів;

L – розмір пакета;

T – час передачі.

При передачі 100 пакетів по 512 байт за 1 секунду:
 $C = 100 \cdot 40961 = 409600$ біт/с $\approx 0,41$ Мбіт/с

Отримане значення значно менше максимальної пропускної здатності Wi-Fi мережі, що свідчить про достатній резерв продуктивності системи [15].

У процесі функціонування телекомунікаційної системи важливим параметром є джиттер – варіація затримки передачі пакетів [16].

Для систем реального часу високий джиттер може призводити до:

– нестабільності передачі;

– втрати синхронізації;

– помилок оброблення даних.

В таблиці 3.1. подано основні параметри передачі даних системи.

Таблиця 3.1 – Основні параметри передачі даних системи

Параметр	Значення
Швидкість каналу	10 Мбіт/с
Розмір пакета	512 байт
Час передачі пакета	0,41 мс
Загальна затримка	15,41 мс
Кількість вузлів	10
Пропускна здатність	0,41 Мбіт/с

На рисунку 3.2 наведено графік зміни джиттера у системі передачі даних.

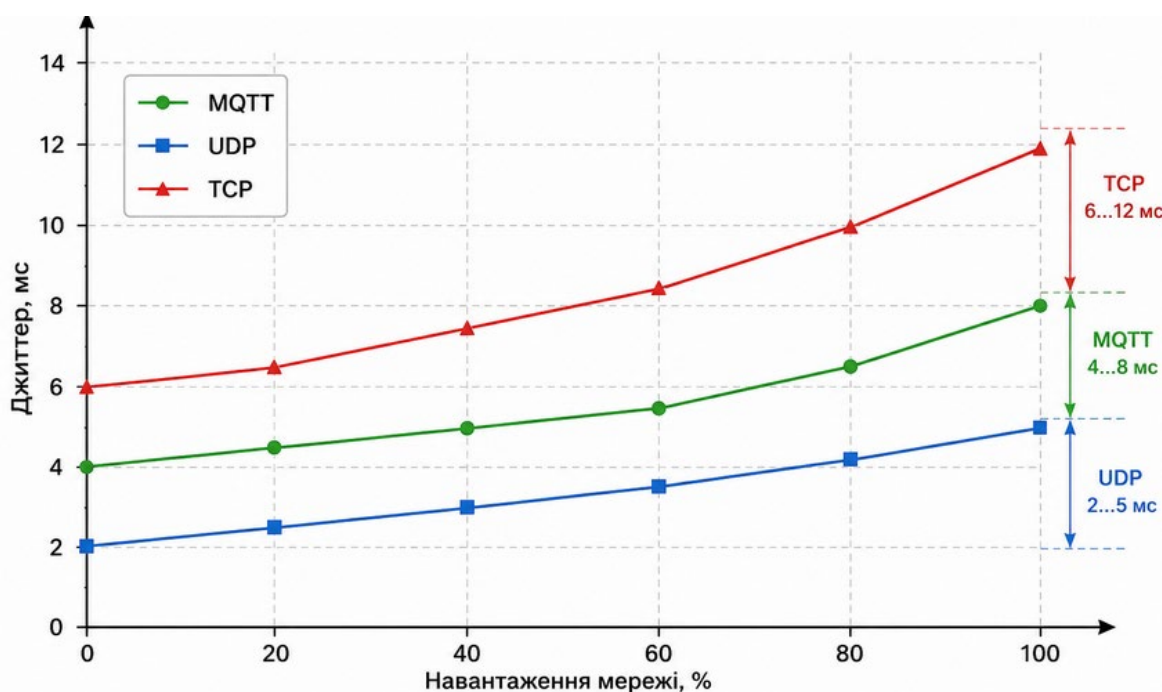


Рисунок 3.2 – Залежність джиттера від навантаження мережі

Для розроблюваної системи середній джиттер становив:

- для MQTT – 4...8 мс;
- для UDP – 2...5 мс;
- для TCP – 6...12 мс.

Такі параметри забезпечують стабільну передачу телеметричної інформації та підтримку роботи системи у режимі реального часу [16].

Також було проведено оцінювання навантаження на embedded-вузол під час передачі даних через Wi-Fi інтерфейс. У режимі активного передавання ESP32 споживає:

- 120...140 мА при MQTT;
- 110...130 мА при UDP;
- 150...170 мА при TCP [4].

Проведені розрахунки показують, що розроблювана система забезпечує:

- низькі затримки передачі;
- достатню пропускну здатність;
- стабільність мережевого обміну;
- підтримку роботи у реальному часі;
- ефективне використання мережевих ресурсів.

Таким чином, параметри передачі даних відповідають вимогам сучасних телекомунікаційних IoT-систем та забезпечують надійне функціонування embedded-системи передачі інформації [1–15].

3.2 Оцінка ефективності протоколів MQTT, UDP, TCP

Для оцінювання ефективності протоколів MQTT, UDP та TCP було проведено дослідження параметрів передачі даних у розроблюваній телекомунікаційній системі.

Основною метою дослідження є визначення впливу типу протоколу на:

- затримку передачі;
- джиттер;
- втрати пакетів;
- навантаження на embedded-вузол;
- енергоспоживання;
- навантаження на MQTT-брокер [3], [9].

Тестування проводилося у локальній Wi-Fi мережі IEEE 802.11n із використанням:

- embedded-вузлів ESP32;
- MQTT-брокера Mosquitto;
- серверної платформи Node-RED;
- телеметричних пакетів розміром 128...1024 байт.

Основні параметри експерименту наведено у таблиці 3.2.

Таблиця 3.2 – Параметри експериментального дослідження

Параметр	Значення
Тип мережі	Wi-Fi IEEE 802.11n
Кількість вузлів	10
Розмір пакетів	128...1024 байт
Частота передачі	10 пакетів/с
MQTT QoS	0, 1
Тривалість тесту	30 хв

Одним із головних параметрів телекомунікаційної системи є затримка передачі повідомлень між embedded-вузлом та серверною платформою. Для оцінювання затримки було проведено серію вимірювань часу доставки пакетів різними протоколами [17].

Під час тестування отримано такі середні значення затримки:

- UDP – 4...12 мс;
- MQTT – 12...24 мс;
- TCP – 22...38 мс.

Результати показують, що затримка TCP суттєво зростає при збільшенні розміру пакета через механізми підтвердження доставки та повторної передачі [18].

У таблиці 3.3 подано результати середньої затримки передачі в залежності від розміру пакета.

Таблиця 3.3 – Середня затримка передачі

Розмір пакета	MQTT	UDP	TCP
128 байт	12 мс	4 мс	22 мс
256 байт	15 мс	6 мс	26 мс
512 байт	19 мс	9 мс	31 мс
1024 байти	24 мс	12 мс	38 мс

Для систем реального часу важливим параметром є джиттер – зміна затримки передачі пакетів у часі [15].

Значний джиттер може призводити до:

- нестабільності телеметрії;
- помилок синхронізації;
- втрати пакетів;
- порушення роботи систем керування.

У процесі дослідження встановлено:

- MQTT – 4...8 мс;
- UDP – 2...5 мс;
- TCP – 6...12 мс.

UDP демонструє найменший джиттер, однак MQTT забезпечує більш стабільну роботу системи при високому навантаженні мережі [17].

Для оцінювання достовірності передачі було виконано тестування передачі 10000 пакетів у мережі Wi-Fi при різному рівні навантаження.

Результати показали:

- TCP має мінімальні втрати пакетів;
- MQTT забезпечує стабільну доставку при QoS 1;
- UDP демонструє найбільший рівень втрат [15].

Для ESP32 було проведено вимірювання струму споживання та навантаження на процесор під час передачі даних [12].

Середні значення струму споживання:

- MQTT – 120...140 мА;

- UDP – 110...130 мА;
- TCP – 150...170 мА.

Найменше навантаження на embedded-вузол забезпечує UDP, однак MQTT демонструє кращу стабільність роботи при тривалому функціонуванні системи [1], [3], [15].

Для оцінювання масштабованості системи було проведено тестування MQTT-брокера при одночасному підключенні декількох embedded-вузлів.

Результати показали:

- стабільну роботу до 100 клієнтів;
- середнє завантаження CPU брокера – 25...40 %;
- збільшення затримки при QoS 2;
- зростання використання пам'яті при великій кількості topic [14].

Для комплексного оцінювання проведено інтегральне порівняння MQTT, UDP та TCP за основними параметрами:

- затримка;
- втрати пакетів;
- енергоспоживання;
- стабільність;
- масштабованість [16].

У таблиці 3.4 подано порівняльну оцінку ефективності протоколів.

Таблиця 3.4 – Порівняльна оцінка ефективності протоколів

Параметр	MQTT	UDP	TCP
Затримка	Низька	Дуже низька	Середня
Джиттер	Низький	Мінімальний	Вищий
Втрати пакетів	Низькі	Високі	Мінімальні
Енергоспоживання	Середнє	Низьке	Високе
Масштабованість	Висока	Середня	Середня

Проведені експериментальні дослідження показали, що:

- MQTT є найбільш універсальним рішенням для IoT-систем;

- UDP доцільно використовувати у задачах реального часу;
- TCP забезпечує максимальну достовірність передачі даних.

Отримані результати підтверджують доцільність комбінованого використання MQTT.

3.3 Розробка алгоритмів передачі даних та програмного забезпечення

Одним із ключових етапів проєктування телекомунікаційної системи передачі даних є розробка алгоритмів функціонування embedded-вузлів та програмного забезпечення для реалізації обміну даними між компонентами системи [1].

Програмне забезпечення повинно забезпечувати:

- стабільну передачу телеметричної інформації;
- підтримку MQTT, UDP та TCP;
- контроль цілісності пакетів;
- буферизацію повідомлень;
- повторну передачу даних;
- підтримку роботи в режимі реального часу [10].

У розроблюваній системі програмне забезпечення реалізовано мовою C для платформи ESP32 із використанням:

- ESP-IDF;– FreeRTOS;
- MQTT Client Library;
- BSD Socket API;
- Wi-Fi Stack [1,11].

Програмне забезпечення embedded-вузла виконує:

- ініціалізацію мережевого інтерфейсу;
- підключення до Wi-Fi мережі;
- зчитування даних із сенсорів;
- формування пакета;
- передачу даних через MQTT/UDP/TCP;

- контроль доставки повідомлень;
- повторну передачу при помилках;
- перехід у режим очікування [1].

На рисунку 3.3 наведено алгоритм роботи embedded-вузла.

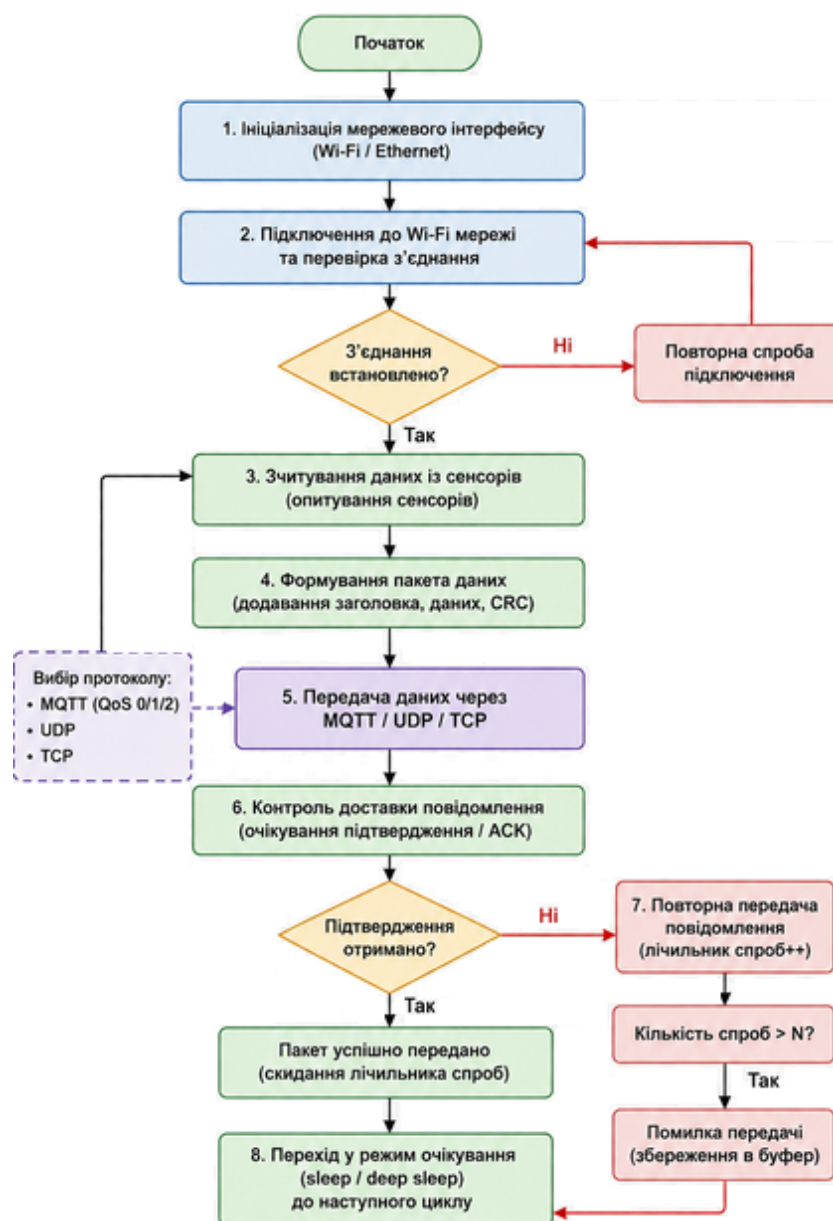


Рисунок 3.3 – Алгоритм роботи embedded-вузла системи передачі даних

Основний цикл функціонування системи складається з:

- опитування сенсорів;
- локальної обробки;
- формування пакета;

- передачі повідомлення;
- перевірки підтвердження доставки [17]

Для передачі даних використовується структура пакета, подана в лістингу 3.1.

Лістинг 3.1 – Структура пакета для передачі даних

```
typedef struct
{
    uint16_t node_id;
    uint32_t timestamp;
    float temperature;
    float humidity;
    uint16_t packet_id;
    uint16_t crc;
} DataPacket
```

Кінець лістингу 3.1

Структура містить:

- ідентифікатор вузла;
- часову мітку;
- параметри сенсорів
- номер пакета;
- CRC-контроль.

Для контролю цілісності повідомлення використовується CRC16.

Функція обчислення CRC подана у лістингу 3.2.

Лістинг 3.2 – Функція обчислення CRC

```
uint16_t calculate_crc(uint8_t *data, uint16_t length)
{
    uint16_t crc = 0xFFFF;
    for(uint16_t i = 0; i < length; i++)
    {
        crc ^= data[i];
        for(uint8_t j = 0; j < 8; j++)
        {
            if(crc & 0x0001)
                crc = (crc >> 1) ^ 0xA001;
            else
                crc >>= 1;
        }
    }
}
```

```

    return crc;
}

```

Кінець лістингу 3.2

Після запуску embedded-вузол виконує ініціалізацію Wi-Fi підключення. Функція ініціалізації Wi-Fi подано у лістингу 3.3.

Лістинг 3.3 – Функція ініціалізації Wi-Fi

```

void wifi_init(void)
{
    esp_netif_init();    esp_event_loop_create_default();
    wifi_init_config_t cfg =
        WIFI_INIT_CONFIG_DEFAULT();
    esp_wifi_init(&cfg);
    esp_wifi_set_mode(WIFI_MODE_STA);
    wifi_config_t wifi_config = {
        .sta =
        {
            .ssid = "IoT_Network",
            .password = "12345678"
        },
    };
    esp_wifi_set_config(
        WIFI_IF_STA,
        &wifi_config
    );
    esp_wifi_start();
    esp_wifi_connect();
}

```

Кінець лістингу 3.3

Для реалізації MQTT-обміну використовується MQTT Client Library.

Функція передачі MQTT-повідомлення подана у лістингу 3.4.

Лістинг 3.4 – Функція передачі MQTT-повідомлення

```

void mqtt_send_data(DataPacket *packet)
{
    char message; [1]
    sprintf(message,
        "{\"temp\": %.2f, "
        "\"hum\": %.2f}",
        packet->temperature,
        packet->humidity);
    esp_mqtt_client_publish(
        client,

```

```

"/iot/data",
message,
0,
1,
0
);
}

```

Кінець лістингу 3.4

Передача MQTT-повідомлення виконується із QoS 1 для забезпечення підтвердження доставки [3].

На рисунку 3.4 наведено алгоритм MQTT-передачі повідомлень.

Для передачі даних через UDP використовується сокетний інтерфейс BSD Socket API.

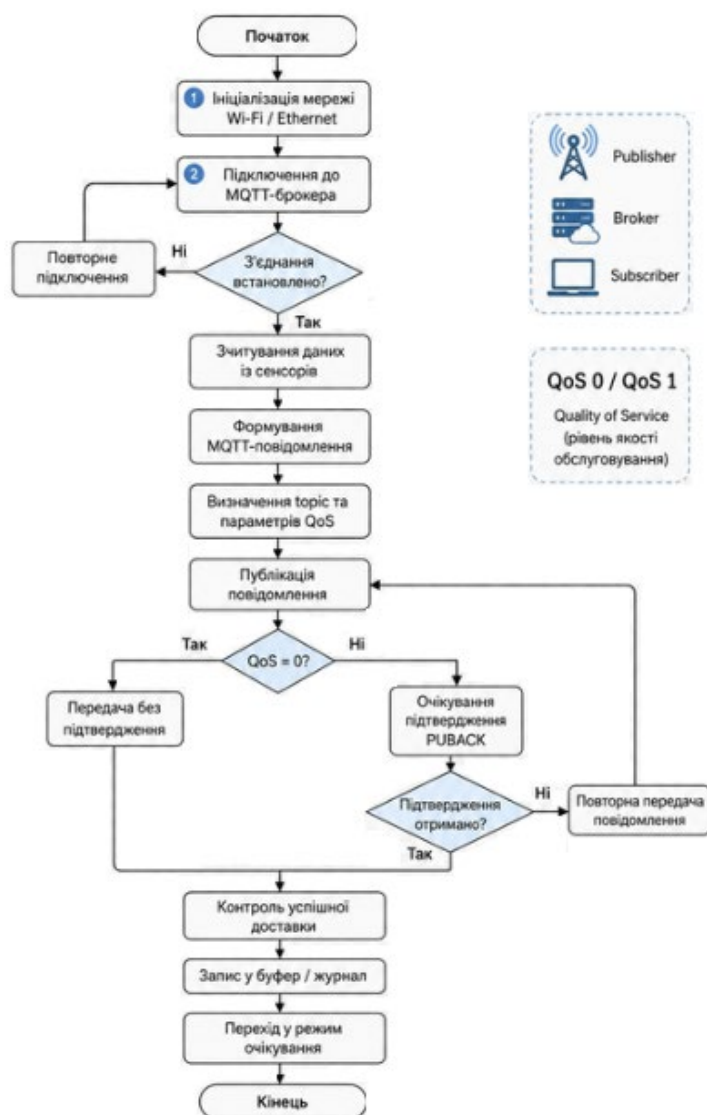


Рисунок 3.4 – Алгоритм MQTT-передачі повідомлень

Функція UDP-передачі подана у лістингу 3.5.

Лістинг 3.5 – Функція UDP-передачі

```
void udp_send_data(DataPacket *packet)
{
    int sock;
    struct sockaddr_in dest_addr;
    dest_addr.sin_addr.s_addr =
        inet_addr("192.168.1.100");
    dest_addr.sin_family = AF_INET;
    dest_addr.sin_port = htons(5000);
    sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_IP);
    sendto(sock,
           packet,
           sizeof(DataPacket),
           0,
           (struct sockaddr *)&dest_addr,
           sizeof(dest_addr));
    close(sock);
}
```

Кінець лістингу 3.5

UDP забезпечує мінімальні затримки передачі даних, тому доцільно використовується у системах реального часу [12], [14].

Функція TCP-передачі подана у лістингу 3.6.

Лістинг 3.6 – Функція TCP-передачі

```
void tcp_send_data(DataPacket *packet)
{
    int sock;
    struct sockaddr_in dest_addr;
    dest_addr.sin_addr.s_addr =
        inet_addr("192.168.1.100");
    dest_addr.sin_family = AF_INET;
    dest_addr.sin_port = htons(6000);
    sock = socket(AF_INET, SOCK_STREAM, IPPROTO_IP);
    connect(sock,
           (struct sockaddr *)&dest_addr,
           sizeof(dest_addr));
    send(sock,
         packet,
         sizeof(DataPacket),
         0);
    close(sock);
}
```

Кінець лістингу 3.6

Для реалізації TCP-з'єднання використовується клієнтський сокет.

TCP забезпечує:

- контроль доставки;
- повторну передачу пакетів;
- перевірку цілісності інформації [1].

Для зменшення навантаження на embedded-вузол використовується режим енергозбереження.

Функція переходу у sleep mode подана у лістингу 3.7.

Лістинг 3.7 – Функція переходу у sleep mode

```
void enter_sleep_mode(void)
{
    esp_sleep_enable_timer_wakeup(
        100000000
    );
    esp_deep_sleep_start();
}
```

Кінець лістингу 3.7

На рисунку 3.5 наведено алгоритм повторної передачі повідомлення у випадку помилки.

У випадку відсутності підтвердження доставки:

- пакет повторно передається;
- збільшується лічильник помилок;
- виконується буферизація повідомлень;
- здійснюється повторне підключення до мережі [15].

Для багатозадачної роботи використовується FreeRTOS.

Основні задачі системи:

- TaskSensorRead;
- TaskPacketBuild;
- TaskMQTT;
- TaskUDP;
- TaskTCP;

– TaskMonitoring.

Приклад створення задачі FreeRTOS подано у лістингу 3.8.

Лістинг 3.8 – Приклад створення задачі FreeRTOS

```
xTaskCreate (
    TaskMQTT,
    "mqtt_task",
    4096,
    NULL,
    5,
    NULL
);
```

Кінець лістингу 3.8

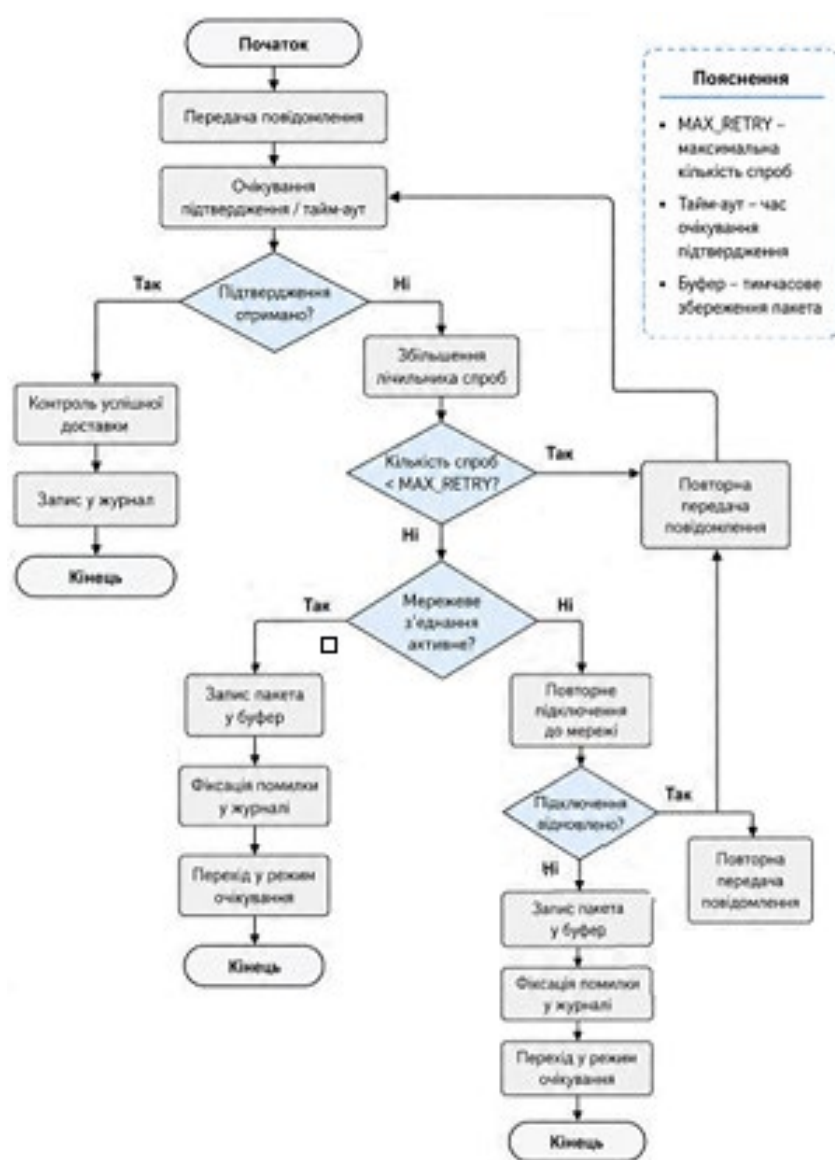


Рисунок 3.5 – Алгоритм повторної передачі повідомлення

Проведене тестування програмного забезпечення показало:

- стабільну роботу MQTT-передачі;
- підтримку одночасної передачі UDP та TCP;
- коректне функціонування механізмів повторної передачі;
- стабільну роботу FreeRTOS при багатозадачному режимі [14].

У таблиці 3.5 подано результати тестування програмного забезпечення.

Таблиця 3.5 – Результати тестування програмного забезпечення

Параметр	Значення
Середня затримка MQTT	18 мс
Середня затримка UDP	7 мс
Середня затримка TCP	29 мс
Втрати пакетів MQTT	<1 %
Втрати пакетів TCP	<0.1 %
Час reconnect MQTT	2...4 с

Таким чином, розроблене програмне забезпечення забезпечує:

- стабільну передачу даних;
- підтримку MQTT / UDP / TCP;
- контроль помилок передачі;
- підтримку режиму реального часу;
- енергоефективне функціонування embedded-вузлів.

Отримані результати підтверджують ефективність розроблених алгоритмів передачі даних для використання у сучасних IoT та телекомунікаційних системах [1].

РОЗДІЛ 4

ОХОРОНА ПРАЦІ ТА БЕЗПЕКА

4.1 Аналіз небезпечних факторів при роботі з телекомунікаційним обладнанням

Під час експлуатації вбудованих телекомунікаційних систем передачі даних з використанням протоколів MQTT, UDP та TCP виникає низка небезпечних і шкідливих факторів, які можуть негативно впливати на персонал, обладнання та надійність функціонування системи. У роботі розглядається телекомунікаційна система, що містить мікроконтролери, мережеві інтерфейси Wi-Fi/Ethernet, MQTT-брокер, серверні вузли, сенсори та модулі бездротового зв'язку [1-5].

Основними джерелами безпеки при роботі з телекомунікаційним обладнанням є:

- електричний струм;
- електромагнітне випромінювання;
- перегрів елементів системи;
- статична електрика;
- ризики короткого замикання;
- психофізіологічне навантаження оператора;– вплив комп'ютерної техніки та мережевого обладнання.

На рисунку 4.1 наведено основні небезпечні фактори під час експлуатації телекомунікаційної системи.

Під час роботи телекомунікаційного обладнання існує небезпека ураження електричним струмом. Особливо це актуально для систем, що використовують мережеві блоки живлення, комутаційне обладнання, Ethernet-інтерфейси та серверні модулі [16-17].

Небезпека виникає при:

- пошкодженні ізоляції провідників;
- неправильному заземленні;

- перевантаженні мережі живлення;
- короткому замиканні;
- роботі у вологому середовищі.

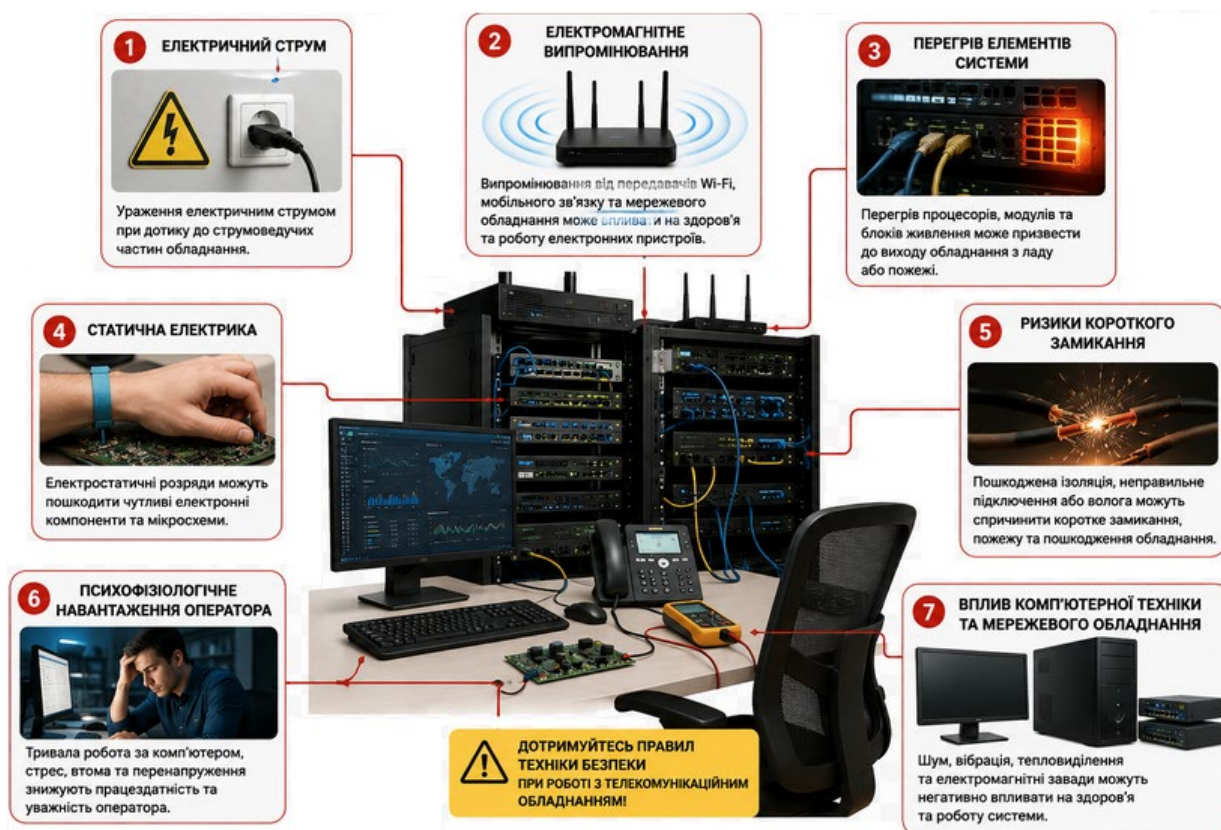


Рисунок 4.1 – Основні небезпечні фактори при роботі з телекомунікаційним обладнанням

Вбудовані IoT-системи часто працюють у режимі безперервного передавання даних через MQTT-брокер та TCP/IP мережі, що призводить до тривалого функціонування електронних компонентів. Унаслідок цього виникає додаткове теплове навантаження на процесори, мережеві контролери та комунікаційні модулі [1], [3], [9].

Одним із важливих факторів є електромагнітне випромінювання, яке створюється:

- Wi-Fi модулями;
- GSM/LTE передавачами;
- мережевими адаптерами;

– високочастотними цифровими інтерфейсами.

Тривалий вплив електромагнітних полів може викликати втому оператора, погіршення концентрації уваги та негативно впливати на роботу електронних компонентів [17], [20].

Для зменшення впливу ЕМП необхідно:

- використовувати екрановані кабелі;
- забезпечувати заземлення;
- дотримуватись нормативних відстаней
- застосовувати сертифіковане обладнання.

У таблиці 4.1 подано основні небезпечні фактори телекомунікаційної системи

Таблиця 4.1 – Основні небезпечні фактори телекомунікаційної системи

Небезпечний фактор	Джерело виникнення	Можливі наслідки
Електричний струм	Блоки живлення, мережеве обладнання	Ураження персоналу
Перегрів обладнання	Процесори, контролери, сервери	Вихід обладнання з ладу
Електромагнітне випромінювання	Wi-Fi, GSM, Ethernet	Погіршення умов праці
Статична електрика	Монтаж та обслуговування плат	Пошкодження мікросхем
Коротке замикання	Несправність електромережі	Пожежа, пошкодження системи
Психофізіологічне навантаження	Робота за ПК	Втома, зниження уваги

Особливу увагу під час роботи з embedded-системами необхідно приділяти захисту від статичної електрики. Сучасні мікроконтролери STM32, ESP32 та мережеві модулі мають високу чутливість до електростатичних розрядів [1], [2], [10].

Для захисту обладнання застосовуються:

- антистатичні браслети;
- заземлені робочі поверхні;
- антистатичні килимки;
- контроль вологості повітря.

Під час роботи з серверним обладнанням та мережевими пристроями необхідно забезпечити належну вентиляцію та охолодження. У системах передачі даних, що використовують MQTT-брокери та TCP/IP-мережі, серверне навантаження може бути значним через постійну обробку пакетів та маршрутизацію повідомлень [3], [9], [4].

Таким чином, аналіз небезпечних факторів показує, що під час роботи з телекомунікаційним обладнанням необхідно забезпечити комплексний підхід до охорони праці. Основними напрямками підвищення безпеки є електрозахист, контроль електромагнітного випромінювання, забезпечення теплового режиму обладнання, ергономіка робочого місця та захист embedded-компонентів від електростатичних розрядів. Це дозволяє підвищити надійність функціонування телекомунікаційної системи та забезпечити безпечні умови праці персоналу [17], [18, 19].

4.2 Ергономіка та безпечна організація робочого місця

Під час проектування, налагодження та експлуатації вбудованих телекомунікаційних систем передачі даних важливе значення має правильна організація робочого місця оператора та розробника системи. Робота з мікроконтролерами STM32, ESP32, мережевим обладнанням, серверними платформами та комп'ютерною технікою супроводжується тривалим перебуванням користувача за персональним комп'ютером, що може призводити до зниження працездатності, підвищеної втоми та професійних захворювань [18, 19].

Основними вимогами до ергономіки робочого місця при роботі з

телекомунікаційним обладнанням є:

- забезпечення комфортних умов праці;
- зменшення навантаження на зір;
- правильне розташування обладнання;
- дотримання норм освітлення;
- підтримання оптимального мікроклімату;
- мінімізація електромагнітного впливу;
- забезпечення електробезпеки [18, 19].

На рисунку 4.2 наведено приклад ергономічної організації робочого місця оператора телекомунікаційної системи.



Рисунок 4.2 – Ергономічна організація робочого місця оператор

Під час роботи з embedded-системами оператор використовує:

- персональний комп'ютер;
- монітор;
- мережеве обладнання;

- мікроконтролерні плати;
- вимірювальні прилади;
- джерела живлення;
- серверні або мережеві модулі [16], [18].

Робоче місце повинно відповідати вимогам ДСанПіН та міжнародних стандартів ергономіки. Монітор необхідно розташовувати на відстані 60...70 см від очей користувача, при цьому верхня межа екрана повинна знаходитися на рівні очей або трохи нижче. Це дозволяє зменшити навантаження на шийний відділ хребта та органи зору [19], [20].

Тривала робота з програмним забезпеченням для моніторингу MQTT-повідомлень, аналізу TCP/UDP-трафіку та налагодження IoT-пристроїв може спричиняти перевтому оператора. Для зменшення навантаження рекомендується:

- робити технологічні перерви;
- використовувати антиблікові екрани;
- забезпечувати природне та штучне освітлення;
- підтримувати правильну поставу під час роботи [17], [20].

Для забезпечення комфортних умов праці важливе значення має освітлення приміщення. Освітленість робочої поверхні повинна становити [19]:

- для роботи з документацією – 300...500 лк;
- для роботи з комп'ютерною технікою – 200...300 лк [16], [18].

Не допускається:

- поява відблисків на екрані монітора;
- різкий контраст між освітленням приміщення та яскравістю дисплея;
- використання несправних освітлювальних приладів.

Під час роботи телекомунікаційного обладнання важливо забезпечити ефективну вентиляцію приміщення, оскільки сервери, маршрутизатори, мережеві шлюзи та мікроконтролерні системи виділяють тепло. Перегрів обладнання може призвести до:

- нестабільної роботи системи;

- помилок передачі даних;
- виходу електронних компонентів з ладу [17], [20].

В таблиці 4.2 наведено основні ергономічні параметри робочого місця.

Таблиця 4.2 – Основні ергономічні параметри робочого місця

Параметр	Рекомендоване значення
Відстань до монітора	60...70 см
Висота столу	72...75 см
Кут огляду монітора	15...20° вниз
Освітленість робочої поверхні	300...500 лк
Температура повітря	20...24 °C
Вологість повітря	40...60 %

Для зменшення негативного впливу електромагнітного випромінювання необхідно:

- використовувати сертифіковане обладнання;
- дотримуватися нормативних відстаней;
- застосовувати екрановані кабелі;
- забезпечувати заземлення обладнання [16], [18].

Особливу увагу під час роботи з embedded-вузлами необхідно приділяти захисту від статичної електрики. Робота з платами ESP32, STM32 та мережевими модулями повинна виконуватись із використанням:

- антистатичних браслетів;
- антистатичних килимків;
- заземлених поверхонь [16], [18].

Таким чином, ергономічна та безпечна організація робочого місця є важливою складовою під час роботи з телекомунікаційними системами передачі даних. Дотримання вимог ергономіки дозволяє підвищити ефективність роботи оператора, зменшити ризик професійних захворювань, забезпечити стабільне функціонування обладнання та підвищити надійність телекомунікаційної системи [19], [20].

ВИСНОВКИ

1. Проведено аналіз сучасних телекомунікаційних систем, вбудованих IoT-пристроїв та принципів передачі даних у мережах реального часу. Розглянуто використання Wi-Fi, Ethernet, хмарних платформ, MQTT-брокерів та edge computing.

2. Виконано порівняльний аналіз протоколів MQTT, UDP та TCP. Визначено, що MQTT є доцільним для IoT-систем завдяки моделі publish / subscribe, низькому навантаженню на мережу та масштабованості. UDP забезпечує мінімальні затримки й підходить для задач реального часу, а TCP доцільний для передавання важливої інформації, оскільки забезпечує контроль доставки, цілісність даних і повторну передачу пакетів.

3. Обґрунтовано вибір комбінованої архітектури системи, яка поєднує broker-based підхід для MQTT, клієнт-серверну модель для TCP та прямий пакетний обмін для UDP.

4. Розроблено структурну, функціональну та принципову електричну схеми системи передачі даних. Описано логіку роботи системи від зчитування даних із сенсорів до передавання інформації через вибраний протокол.

5. Виконано розрахунок основних параметрів передачі даних: затримки, джиттера, пропускної здатності, втрат пакетів та ефективності використання MQTT, UDP і TCP.

6. Розроблено алгоритми передачі даних та програмне забезпечення для MQTT, UDP і TCP-обміну.

7. Проведено оцінку ефективності розробленої системи. Встановлено, що вона забезпечує стабільну передачу даних, підтримує роботу з протоколами MQTT, UDP та TCP.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC 30141:2024. Internet of Things (IoT). Reference architecture. Geneva: International Organization for Standardization, 2024. 53 p. URL: <https://www.iso.org/standard/88800.html> (дата звернення: 03.02.2026).
2. Fagan M., Megas K. N., Cuthill B., Marron J., Hoehn B. Foundational Cybersecurity Activities for IoT Product Manufacturers: NIST IR 8259r1. Gaithersburg: National Institute of Standards and Technology, 2026. 64 p. URL: <https://doi.org/10.6028/NIST.IR.8259r1> (дата звернення: 03.06.2026).
3. HiveMQ Team. MQTT 5 Essentials: Advanced MQTT for IoT Communication. Landshut: HiveMQ, 2026. URL: <https://www.hivemq.com/mqtt/mqtt-5/> (дата звернення: 13.02.2026).
4. Eddy W. M. Transmission Control Protocol (TCP): RFC 9293. Fremont: RFC Editor, 2022. URL: <https://www.rfc-editor.org/info/rfc9293> (дата звернення: 17.02.2026).
5. Touch J., Heard C. Transport Options for UDP: RFC 9868. Fremont: RFC Editor, 2025. URL: <https://www.rfc-editor.org/info/rfc9868> (дата звернення: 03.03.2026).
6. AWS. MQTT: AWS IoT Core Developer Guide. Seattle: Amazon Web Services, 2026. URL: <https://surl.li/rrevdf> (дата звернення: 17.03.2026).
7. Microsoft. Use MQTT to communicate with Azure IoT Hub. Redmond: Microsoft Learn, 2026. URL: <https://learn.microsoft.com/en-us/azure/iot-hub/iot-mqtt-connect-to-iot-hub> (дата звернення: 17.03.2026).
8. Google Cloud. IoT platform product architecture on Google Cloud. Mountain View: Google Cloud Architecture Center, 2024. URL: <https://surl.li/xwntdp> (дата звернення: 15.04.2026).
9. Eclipse Foundation. Eclipse Mosquitto: An open source MQTT broker. Brussels: Eclipse Foundation, 2026. URL: <https://mosquitto.org/> (дата звернення: 15.04.2026).

10. Espressif Systems. ESP32 Series Datasheet. Version 5.2. Shanghai: Espressif Systems, 2025. 76 p. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (дата звернення: 15.04.2026).

11. WIZnet. W6300 Datasheet. Version 1.0.0. Seongnam: WIZnet Co., Ltd., 2025. 146 p. URL: https://docs.wiznet.io/assets/files/W6300%20DS_V100E-b7faabfc62400edb7ae942f77c424dca.pdf (дата звернення: 03.06.2026).

12. Raspberry Pi Ltd. Raspberry Pi 4 Model B Specifications. Cambridge: Raspberry Pi Ltd., 2026. URL: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/> (дата звернення: 15.04.2026).

13. STMicroelectronics. STM32CubeIDE User Guide: UM2609. Version 17.0. Geneva: STMicroelectronics, 2025. URL: <https://surl.li/cvfifd> (дата звернення: 03.06.2026).

14. Amazon Web Services. FreeRTOS 202604 LTS Libraries. Seattle: AWS, 2026. URL: <https://freertos.org/Documentation/03-Libraries/01-Library-overview/03-LTS-libraries/01-LTS-libraries> (дата звернення: 03.06.2026).

15. Rose S., Borchert O., Kerman A., Souppaya M. et al. Implementing a Zero Trust Architecture: NIST Special Publication 1800-35. Gaithersburg: National Institute of Standards and Technology, 2025. 54 p. URL: <https://doi.org/10.6028/NIST.SP.1800-35> (дата звернення: 03.06.2026).

16. ДСТУ EN IEC 62368-1:2022. Обладнання аудіо/відео, інформаційних та комунікаційних технологій. Частина 1. Вимоги щодо безпеки. Київ: ДП «УкрНДНЦ», 2022. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=105881 (дата звернення: 25.04.2026).

17. ДСТУ EN ISO 9241-302:2022. Ергономіка взаємодії людина-система. Частина 302. Термінологія для електронних візуальних дисплеїв. Київ: ДП «УкрНДНЦ», 2022. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=112528 (дата звернення: 03.06.2026).

18. ДБН В.2.5-23:2025. Проектування електроустановок об'єктів цивільного призначення. Київ: Мінрегіон України, 2025. URL:

https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=117699 (дата звернення: 03.06.2026).

19. ДСТУ EN ISO 9241-307:2022. Ергономіка взаємодії людина-система. Частина 307. Методи аналізування та перевірки на відповідність електронних візуальних дисплеїв. Київ: ДП «УкрНДНЦ», 2022. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=112608 (дата звернення: 03.06.2026).

20. ISO 9241-5:2024. Ergonomics of human-system interaction. Part 5: Workstation layout and postural requirements. Geneva: International Organization for Standardization, 2024. 27 p. URL: <https://www.iso.org/standard/86222.html> (дата звернення: 27.05.2026).