

**Міністерство освіти і науки України  
Луцький національний технічний університет  
Факультет комп'ютерних та інформаційних технологій  
Кафедра інженерії програмного забезпечення**

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

**РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОРГАНІЗАЦІЇ  
МАРШРУТИЗАЦІЇ ДАНИХ У MESH-МЕРЕЖІ  
З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ LORA**

**SOFTWARE DEVELOPMENT FOR ORGANIZING DATA  
ROUTING IN A MESH NETWORK  
USING LORA TECHNOLOGY**

спеціальність 121 «Інженерія програмного забезпечення»  
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти  
групи ІПЗ-41  
**Савчук Андрій Валентинович**

Керівник:  
**Потейчук Михайло Іванович**

Кваліфікаційну роботу  
допущено до захисту  
«\_\_» \_\_\_\_\_ 20\_\_ р.  
Гарант освітньої програми:  
к.т.н., доцент  
Ліщина Наталія Миколаївна

---

Луцьк – 2026 року

# ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

«\_\_» \_\_\_\_\_ 20\_\_ р.

## ЗАВДАННЯ

### НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Савчуку Андрію Валентиновичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка програмного забезпечення для організації маршрутизації даних у mesh-мережі з використанням технології LoRa»

Керівник роботи: асистент, Потейчук М. І.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра «29» травня 2026 р.

3. Вихідні дані до роботи: технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз і огляд сучасного стану проблеми побудови mesh-мереж з використанням технології LoRa та формулювання завдань кваліфікаційної роботи бакалавра; визначення вимог до програмного забезпечення вузла LoRa mesh-мережі; обґрунтування вибору апаратних і програмних засобів, зокрема мікроконтролера ESP32, радіомодуля E32-400M30S, середовища розробки та алгоритмів маршрутизації; проектування структури програмного забезпечення з використанням UML-діаграм; розробка службового пакета, маршрутної таблиці, механізмів перевірки CRC, фільтрації повторів, контролю TTL і ретрансляції повідомлень; тестування та налагодження програмної логіки; розгляд заходів захисту інформаційно-комп'ютерної системи; визначення перспектив подальшого розвитку програмного забезпечення.

5. Перелік графічного матеріалу: 12 рисунків, 10 лістингів

## 6. Консультанти розділів роботи

| Розділ                                    | Прізвище, ініціали та посада консультанта | Підпис         |                  |
|-------------------------------------------|-------------------------------------------|----------------|------------------|
|                                           |                                           | завдання видав | завдання прийняв |
| Аналіз предметної області                 | <i>Потейчук М. І.</i>                     |                |                  |
| Специфікація вимог до розробленої системи | <i>Потейчук М. І.</i>                     |                |                  |
| Розробка об'єкта проєктування             | <i>Потейчук М. І.</i>                     |                |                  |
| Нормоконтроль                             | <i>Суринович О. М.</i>                    |                |                  |
| Гарант ОП                                 | <i>Ліщина Н. М.</i>                       |                |                  |
| Показник запозичень тексту                |                                           | %              |                  |
| Академічна доброчесність                  | <i>Потейчук М. І.</i>                     |                |                  |

## 7. Дата видачі завдання «3» лютого 2026 р.

| № | Назва етапів кваліфікаційної роботи бакалавра                                                   | Строк виконання етапів роботи | Примітка |
|---|-------------------------------------------------------------------------------------------------|-------------------------------|----------|
| 1 | Огляд літературних джерел по темі кваліфікаційної роботи бакалавра                              | до 14.02.2026 р.              |          |
| 2 | Аналіз проблеми розробки та впровадження об'єкту проєктуванням                                  | до 03.03.2026 р.              |          |
| 3 | Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання               | до 14.03.2026 р.              |          |
| 4 | Розробка функціонально-структурної схеми роботи об'єкта проєктування та проєктування бази даних | до 14.04.2026 р.              |          |
| 5 | Практична реалізація об'єкта проєктування та захист системи                                     | до 20.04.2026 р.              |          |
| 6 | Тестування та налагодження об'єкта проєктуванням                                                | до 04.05.2026 р.              |          |
| 7 | Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі                            | до 29.05.2026 р.              |          |

Здобувач вищої освіти \_\_\_\_\_

Андрій САВЧУК

Керівник кваліфікаційної роботи \_\_\_\_\_

Михайло ПОТЕЙЧУК

## АНОТАЦІЯ

Савчук А. В. Розробка програмного забезпечення для організації маршрутизації даних у mesh-мережі з використанням технології LoRa. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається з вступу, трьох розділів, висновків, списку використаних джерел.

У першому розділі виконано аналіз сучасного стану технологій mesh-мереж та маршрутизації даних з використанням LoRa. Розглянуто особливості побудови автономних бездротових мереж, принципи багатопотокового передавання пакетів, можливості використання ESP32 та радіомодуля E32-400M30S. Сформульовано завдання кваліфікаційної роботи бакалавра.

У другому розділі визначено функціональні та нефункціональні вимоги до програмного забезпечення вузла LoRa mesh-мережі. Спроектовано структуру програмного забезпечення, розроблено UML-діаграми основних модулів і послідовності обробки пакета. Обґрунтовано вибір апаратних засобів, середовища розробки, структури службового пакета та алгоритмічних механізмів маршрутизації.

У третьому розділі описано практичну реалізацію програмного забезпечення: структуру службового пакета, механізм перевірки контрольної суми, маршрутну таблицю, журнал оброблених пакетів, контроль TTL та ретрансляцію повідомлень. Проведено тестування та налагодження програмної логіки, розглянуто захист інформаційно-комп'ютерної системи, наведено перспективи подальшого розвитку програмного забезпечення.

Ключові слова: LoRa, mesh-мережа, ESP32, E32-400M30S, маршрутизація даних, службовий пакет, TTL, CRC.

## **ABSTRACT**

Savchuk Andrii. Software Development for Organizing Data Routing in a Mesh Network Using LoRa Technology. Manuscript. Bachelor's Qualification Thesis of the Educational Program "Software Engineering" in the Specialty "Software Engineering". Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, and a list of references.

The first chapter analyzes the current state of mesh network technologies and data routing using LoRa. The features of autonomous wireless network construction, the principles of multi-hop packet transmission, and the possibilities of using the ESP32 microcontroller and the E32-400M30S radio module are considered. The tasks of the bachelor's qualification thesis are formulated.

The second chapter defines the functional and non-functional requirements for the software of a LoRa mesh network node. The software structure is designed, and UML diagrams of the main modules and the packet processing sequence are developed. The choice of hardware tools, development environment, service packet structure, and algorithmic routing mechanisms is substantiated.

The third chapter describes the practical implementation of the software, including the service packet structure, checksum verification mechanism, routing table, processed packet log, TTL control, and message retransmission. Testing and debugging of the software logic are carried out, the protection of the information and computer system is considered, and prospects for further software development are presented.

Keywords: LoRa, mesh network, ESP32, E32-400M30S, data routing, service packet, TTL, CRC.

## ЗМІСТ

|                                                                                                                                             |    |
|---------------------------------------------------------------------------------------------------------------------------------------------|----|
| ВСТУП .....                                                                                                                                 | 7  |
| РОЗДІЛ 1 АНАЛІЗ ТЕХНОЛОГІЙ MESH-МЕРЕЖ ТА МАРШРУТИЗАЦІЇ<br>ДАНИХ З ВИКОРИСТАННЯМ LORA І ПОСТАНОВКА ЗАВДАНЬ НА<br>КВАЛІФІКАЦІЙНУ РОБОТУ ..... | 10 |
| 1.1 Аналіз сучасного стану проблеми .....                                                                                                   | 10 |
| 1.2 Постановка завдання на кваліфікаційну роботу бакалавра .....                                                                            | 17 |
| РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ .....                                                                                  | 18 |
| 2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та<br>проектування програмного забезпечення .....                   | 18 |
| 2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання .....                                                               | 29 |
| РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ<br>МАРШРУТИЗАЦІЇ ДАНИХ У MESH-МЕРЕЖІ З ВИКОРИСТАННЯМ<br>LORA .....                           | 35 |
| 3.1 Практична реалізація об'єкта проектування .....                                                                                         | 35 |
| 3.2 Тестування та налагодження програмно забезпечення.....                                                                                  | 40 |
| 3.3 Захист інформаційно-комп'ютерної системи.....                                                                                           | 45 |
| 3.4 Перспективи подальшого розвитку програмно забезпечення.....                                                                             | 50 |
| ВИСНОВКИ.....                                                                                                                               | 54 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....                                                                                                             | 58 |

## ВСТУП

Актуальність теми. Сучасний розвиток інформаційно-комунікаційних технологій супроводжується зростанням потреби у створенні автономних бездротових систем передавання даних. Такі системи є актуальними для об'єктів, розміщених на значній відстані один від одного, у складних умовах рельєфу, за відсутності стабільного мобільного зв'язку, Wi-Fi-покриття або дротової інфраструктури. Особливого значення набувають мережі, здатні функціонувати без постійної залежності від централізованого вузла, оскільки в багатьох практичних умовах встановлення шлюзу або підтримка постійного каналу зв'язку є ускладненими.

Однією з технологій, що може використовуватися для побудови таких систем, є LoRa. Вона орієнтована на передавання невеликих обсягів даних на значні відстані за умови відносно низького енергоспоживання. Водночас сама технологія LoRa забезпечує переважно фізичний рівень радіообміну і не визначає повноцінної логіки маршрутизації між вузлами. У звичайних системах зв'язку повідомлення часто передається безпосередньо від одного пристрою до іншого або через центральний шлюз. Однак у mesh-мережі кожен вузол може бути не лише джерелом або отримувачем даних, а й проміжним ретранслятором, який передає пакет далі за маршрутом. Саме тому для таких мереж необхідне спеціалізоване програмне забезпечення, яке забезпечує адресацію, обробку службових пакетів, вибір наступного вузла, контроль повторів і обмеження кількості ретрансляцій.

Актуальність теми кваліфікаційної роботи бакалавра полягає в необхідності розробки програмного забезпечення для організації маршрутизації даних у LoRa mesh-мережі. Використання mesh-підходу дозволяє розширити зону покриття мережі, підвищити її гнучкість і забезпечити передавання даних через проміжні вузли. Водночас така архітектура потребує програмної реалізації механізмів, які не допускають неконтрольованого поширення пакетів, повторної обробки повідомлень,

використання пошкоджених даних або помилкового вибору маршруту. Тому розробка програмної логіки вузла LoRa mesh-мережі є важливим завданням для побудови автономних бездротових систем передавання даних.

Метою кваліфікаційної роботи бакалавра є розробка програмного забезпечення для організації маршрутизації даних у mesh-мережі з використанням технології LoRa.

Об'єктом дослідження є процес передавання та маршрутизації даних у бездротовій mesh-мережі з використанням технології LoRa.

Предметом дослідження є програмні методи формування, перевірки, маршрутизації та ретрансляції службових пакетів між вузлами LoRa mesh-мережі.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати сучасний стан технологій LoRa, принципи побудови mesh-мереж і особливості маршрутизації даних у бездротових низькошвидкісних мережах;
- дослідити можливості використання мікроконтролера ESP32 та радіомодуля E32-400M30S як апаратної основи вузла LoRa mesh-мережі;
- визначити функціональні та нефункціональні вимоги до програмного забезпечення вузла mesh-мережі;
- спроектувати структуру програмного забезпечення з поділом на модулі приймання, перевірки пакета, маршрутизації, ретрансляції, роботи з маршрутною таблицею та діагностики;
- сформувати структуру службового пакета для передавання даних між вузлами мережі;
- реалізувати основні програмні механізми обробки пакетів, зокрема перевірку контрольної суми, фільтрацію повторів, пошук наступного вузла, контроль TTL і ретрансляцію;
- виконати тестування та налагодження розробленої програмної логіки;
- розглянути механізми захисту інформаційно-комп'ютерної системи;

– визначити перспективи подальшого розвитку програмного забезпечення.

Практичне значення роботи полягає в розробці програмної основи вузла LoRa mesh-мережі, яка може бути використана для організації передавання даних через проміжні вузли. Запропонована логіка передбачає формування службового пакета, перевірку його цілісності, фільтрацію повторних повідомлень, обмеження кількості переходів за допомогою TTL, пошук наступного вузла та прийняття рішення щодо локальної обробки або ретрансляції. Такий підхід може бути використаний як основа для подальшого вдосконалення автономних бездротових систем передавання даних на базі ESP32 та LoRa-модулів.

## **РОЗДІЛ 1**

### **АНАЛІЗ ТЕХНОЛОГІЙ MESH-МЕРЕЖ ТА МАРШРУТИЗАЦІЇ ДАНИХ З ВИКОРИСТАННЯМ LORA І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ**

#### **1.1 Аналіз сучасного стану проблеми**

Сучасний розвиток бездротових інформаційно-комунікаційних систем характеризується зростанням потреби у створенні автономних мереж передавання даних, здатних функціонувати без постійної залежності від централізованої інфраструктури. Такі системи є особливо актуальними для об'єктів, розташованих на значній відстані один від одного, у складних умовах рельєфу, за відсутності стабільного мобільного зв'язку, Wi-Fi-покриття або дротових каналів передавання інформації. До сфер практичного застосування таких рішень належать промисловий моніторинг, аграрні інформаційні системи, екологічний контроль, охоронні комплекси, автономна телеметрія, системи спостереження та розподілені сенсорні мережі. У межах таких систем важливим завданням є не лише забезпечення фізичного радіозв'язку між окремими пристроями, але й організація стійкої маршрутизації даних між вузлами мережі.

Однією з технологій, що активно використовується для побудови енергоефективних мереж великого радіуса дії, є LoRa. Дана технологія базується на принципах широкосмугової модуляції з розширенням спектра та орієнтована на передавання невеликих обсягів даних на значні відстані при відносно низькому енергоспоживанні. На відміну від високошвидкісних бездротових технологій, LoRa не призначена для передавання великих потоків інформації, однак її перевагою є можливість роботи в умовах складного радіосередовища, підвищена завадостійкість і здатність забезпечувати зв'язок там, де використання Wi-Fi або класичних короткодіапазонних радіомодулів є недоцільним. Саме тому LoRa розглядається як одна з ефективних технологій для побудови автономних мереж обміну короткими повідомленнями [1].

У даній кваліфікаційній роботі апаратною основою для побудови вузлів mesh-мережі є мікроконтролер ESP32 та радіомодуль E32-400M30S, що працює у частотному діапазоні 433 МГц. ESP32 є продуктивною вбудованою платформою, яка має достатні обчислювальні ресурси для реалізації логіки приймання, обробки, маршрутизації та передавання пакетів. Крім того, мікроконтролер підтримує набір периферійних інтерфейсів, зокрема UART, SPI, I2C, GPIO, ADC та інші апаратні засоби, що дозволяє інтегрувати його із зовнішніми радіомодулями та додатковими пристроями. Завдяки цьому ESP32 може використовуватись не лише як простий контролер передавання даних, а як повноцінний обчислювальний вузол mesh-мережі [2].

Радіомодуль E32-400M30S є спеціалізованим LoRa-модулем, побудованим на базі мікросхеми Semtech SX1278. Він підтримує роботу в діапазонах 433/470 МГц і має максимальну потужність передавача 30 dBm, що відповідає 1 Вт. Такі характеристики є важливими для систем, у яких необхідно забезпечити передавання даних на значні відстані. Водночас збільшення потужності передавача не є універсальним вирішенням проблеми покриття, оскільки реальна дальність зв'язку залежить від рельєфу місцевості, висоти розташування антен, параметрів модуляції, рівня електромагнітних завад, якості живлення та взаємного розміщення вузлів. Саме тому доцільним є застосування не лише прямого каналу зв'язку між двома пристроями, а й багатовузлової структури, у якій проміжні вузли можуть виконувати роль ретрансляторів [3].

Узагальнена структура LoRa mesh-мережі наведена на рисунку 1.1.

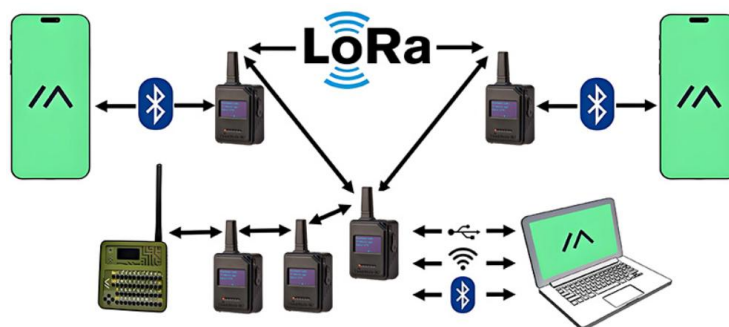


Рисунок 1.1 – Узагальнена структура LoRa mesh-мережі [4]

Класична архітектура LoRaWAN переважно базується на топології типу «зірка зірок», у якій кінцеві пристрої передають дані на шлюзи, а шлюзи пересилають ці дані на мережевий сервер через IP-інфраструктуру. У такій моделі кінцеві вузли не виконують функцію ретрансляції один для одного, а вся логіка обміну даними значною мірою залежить від наявності шлюзів і серверної частини. Така архітектура є ефективною для багатьох IoT-застосувань, однак вона має обмеження у випадках, коли вузол перебуває поза зоною приймання шлюзу або коли якість радіоканалу погіршується через значну відстань, перешкоди, забудову чи складний рельєф. У таких умовах виникає необхідність у використанні альтернативних топологій, де повідомлення може передаватися через проміжні вузли [5].

На відміну від зіркоподібної структури, mesh-мережа передбачає, що кожен вузол може виконувати не лише роль джерела або отримувача інформації, але й роль ретранслятора. Це дає змогу організувати багатохопове передавання даних, коли повідомлення проходить від початкового вузла до кінцевого адресата через один або декілька проміжних пристроїв. У результаті з'являється можливість збільшити ефективну зону покриття мережі без обов'язкового встановлення потужного центрального шлюзу. Крім того, mesh-архітектура підвищує гнучкість системи, оскільки в разі втрати одного маршруту потенційно може бути використано інший шлях передавання даних [6].

Основні топології комп'ютерних мереж, які можуть використовуватися для пояснення принципів організації LoRa-мереж, наведено на рисунку 1.2.

Основна проблема побудови mesh-мережі на основі LoRa полягає в тому, що фізичний радіоканал не забезпечує сам по собі повноцінної логіки маршрутизації. LoRa визначає спосіб передавання радіосигналу, однак не задає механізмів побудови маршрутних таблиць, вибору наступного вузла, контролю повторів, запобігання зацикленню пакетів і підтвердження доставки. У традиційних IP-мережах ці функції реалізуються спеціалізованими мережевими протоколами, проте у LoRa-системах їх потрібно адаптувати до обмежень

низькошвидкісного радіоканалу. Саме тому програмне забезпечення для LoRa mesh-мережі повинно містити власний рівень адресації, службового обміну, маршрутизації та контролю коректності передавання [6].

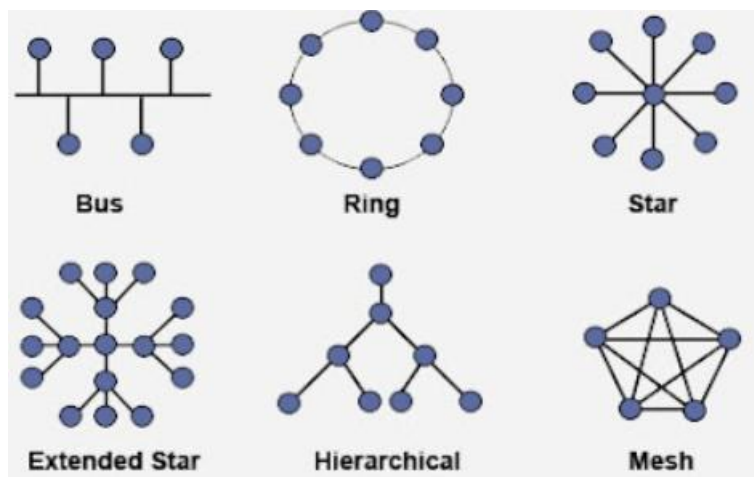


Рисунок 1.2 – Топології мереж в LoRa-мережах [7]

Одним із важливих підходів до організації маршрутизації у mesh-мережах є використання спрощених маршрутних таблиць, у яких вузли зберігають інформацію про доступність інших вузлів і можливий наступний перехід для передавання пакета. Проте пряме перенесення класичних протоколів маршрутизації у LoRa-середовище є неефективним, оскільки LoRa має обмежену пропускну здатність, значний час передавання окремого пакета та підвищену чутливість до надмірного службового трафіку. Кожне службове повідомлення займає радіоканал, збільшує енергоспоживання та може створювати колізії з іншими пакетами. Тому алгоритм маршрутизації для LoRa mesh-мережі має бути достатньо простим для реалізації на мікроконтролері, але водночас повинен забезпечувати коректне передавання даних через проміжні вузли [6].

Для побудови програмного забезпечення LoRa mesh-мережі важливим є формування структури службового пакета. На відміну від простої передачі даних між двома пристроями, у mesh-мережі пакет повинен містити не лише корисне навантаження, але й службові поля, необхідні для маршрутизації. До

таких полів доцільно віднести унікальний ідентифікатор пакета, адресу вузла-відправника, адресу вузла-отримувача, адресу попереднього вузла, тип повідомлення, номер послідовності, лічильник TTL, контрольну суму та власне корисні дані. Наявність TTL дозволяє обмежити кількість ретрансляцій і запобігти нескінченному поширенню пакета в мережі. Унікальний ідентифікатор пакета дає змогу відфільтровувати дублікати, а адресні поля забезпечують можливість визначити, чи повинен вузол обробити пакет як кінцевий отримувач, чи передати його далі.

Приклад передавання пакета через проміжні вузли у LoRa mesh-мережі наведено на рисунку 1.3.

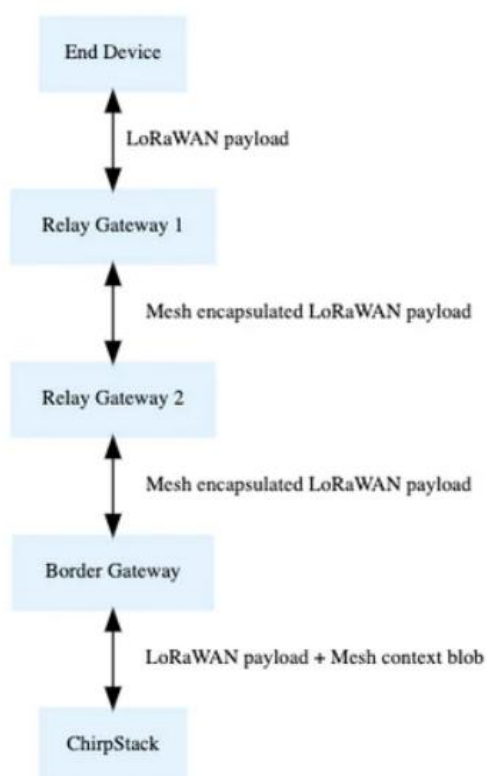


Рисунок 1.3 – Приклад передавання пакета у LoRaWAN mesh-мережі [8]

Одним із ключових завдань маршрутизації є вибір критерію, за яким вузол приймає рішення про подальшу передачу пакета. У класичних комп'ютерних мережах як метрики можуть використовуватися кількість переходів, пропускна здатність, затримка, завантаження каналу або

адміністративна вартість маршруту. У LoRa mesh-мережі до цих параметрів доцільно додати показники якості радіозв'язку, зокрема RSSI та SNR, стабільність доставки пакетів, час останнього контакту із сусіднім вузлом і кількість втрачених повідомлень. Водночас надмірне ускладнення метрик може призвести до збільшення службового трафіку, що є небажаним для LoRa-мереж. Тому для практичної реалізації доцільно застосовувати компромісний підхід, базовою метрикою може бути кількість переходів до отримувача, а допоміжними параметрами, якістю сигналу та актуальність маршруту.

Окремий інтерес становлять gateway-free LoRa mesh-системи, тобто мережі, які не потребують постійного центрального шлюзу для обміну повідомленнями між вузлами. У таких системах кожен пристрій може працювати як автономний вузол, що приймає, аналізує, обробляє та пересилає повідомлення іншим вузлам. Такий підхід є близьким до тематики даної кваліфікаційної роботи, оскільки ESP32 у поєднанні з LoRa-модулем E32-400M30S може функціонувати як самостійний елемент mesh-мережі. Особливо важливою властивістю таких систем є здатність адаптуватися до зміни топології, наприклад у разі вимкнення окремого вузла, появи нового вузла або погіршення якості радіоканалу [9].

Побудова mesh-мережі з використанням LoRa супроводжується низкою технічних і програмних обмежень. По-перше, LoRa має обмежену швидкість передавання даних, тому збільшення кількості службових пакетів може суттєво знизити ефективну пропускну здатність мережі. По-друге, кожна ретрансляція збільшує затримку доставки повідомлення. По-третє, у мережі можуть виникати дублікати пакетів, якщо декілька вузлів одночасно ретранслюють одне й те саме повідомлення. По-четверте, за відсутності механізмів контролю маршрутів можливе зациклення пакетів. По-п'яте, висока потужність передавання модуля E32-400M30S збільшує споживання енергії, тому програмне забезпечення повинно мінімізувати зайві передачі [9].

Порівняно із централізованими мережами, mesh-архітектура забезпечує вищу гнучкість і потенційну стійкість до відмов окремих вузлів. Якщо один

маршрут стає недоступним, мережа може використовувати альтернативний шлях передавання повідомлення. Однак ця перевага не виникає автоматично, а повинна бути реалізована на рівні програмної логіки. Для цього вузли мають зберігати інформацію про сусідів, оновлювати маршрутні таблиці, видаляти застарілі записи, контролювати повтори та обмежувати кількість ретрансляцій. Такі механізми дають змогу зменшити кількість некоректних пересилань і підвищити стійкість мережі до зміни її структури [9].

Для розроблюваної системи доцільним є використання спрощеної моделі mesh-маршрутизації, адаптованої до обмежень мікроконтролерних пристроїв і LoRa-радіоканалу. Така модель повинна передбачати унікальну адресу кожного вузла, таблицю сусідів, таблицю маршрутів, журнал останніх прийнятих пакетів, службові типи повідомлень і механізм обмеження кількості переходів. У найпростішому випадку вузол може зберігати інформацію про те, через який сусідній вузол доцільно передавати повідомлення до певного отримувача. Якщо маршрут відсутній, система може використовувати обмежене широкомовне поширення із контролем TTL та фільтрацією повторів.

Отже, аналіз сучасного стану проблеми показує, що використання LoRa для побудови mesh-мереж є актуальним напрямом у сфері автономних бездротових систем передавання даних. Класична LoRaWAN-архітектура є ефективною для мереж зі шлюзами та серверною інфраструктурою, однак вона не вирішує повністю проблему безпосередньої взаємодії між вузлами. Mesh-підхід дозволяє розширити зону покриття, підвищити гнучкість мережі та забезпечити передавання даних через проміжні вузли. Водночас така архітектура потребує спеціального програмного забезпечення для адресації, маршрутизації, контролю повторів, обмеження кількості ретрансляцій і оновлення маршрутної інформації. Використання ESP32 у поєднанні з модулем E32-400M30S створює технічну основу для реалізації такої системи, а основним завданням розробки є створення програмного рівня, який забезпечить стабільну взаємодію вузлів у LoRa mesh-мережі.

## 1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Проведений аналіз сучасного стану проблеми показав, що для побудови LoRa mesh-мережі недостатньо лише організувати радіообмін між окремими пристроями. Оскільки технологія LoRa не реалізує повноцінну логіку mesh-маршрутизації на рівні прикладного програмного забезпечення, у межах кваліфікаційної роботи бакалавра необхідно розробити програмну логіку вузла, здатного працювати як джерело повідомлення, проміжний ретранслятор або кінцевий отримувач даних.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- проаналізувати сучасний стан технологій LoRa, принципи побудови mesh-мереж і особливості маршрутизації даних у бездротових низькошвидкісних мережах;
- дослідити можливості використання мікроконтролера ESP32 та радіомодуля E32-400M30S як апаратної основи вузла LoRa mesh-мережі;
- визначити функціональні та нефункціональні вимоги до програмного забезпечення вузла mesh-мережі;
- спроектувати структуру програмного забезпечення з поділом на модулі приймання, перевірки пакета, маршрутизації, ретрансляції, роботи з маршрутною таблицею та діагностики;
- сформувати структуру службового пакета для передавання даних між вузлами мережі;
- реалізувати основні програмні механізми обробки пакетів, зокрема перевірку контрольної суми, фільтрацію повторів, пошук наступного вузла, контроль TTL і ретрансляцію;
- виконати тестування та налагодження розробленої програмної логіки;
- розглянути механізми захисту інформаційно-комп'ютерної системи;
- визначити перспективи подальшого розвитку програмного забезпечення.

## РОЗДІЛ 2

### СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

#### 2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Розробка програмного забезпечення для організації маршрутизації даних у mesh-мережі з використанням технології LoRa передбачає створення програмної логіки, яка забезпечує не лише передавання пакетів між окремими пристроями, а й їхню обробку, фільтрацію, маршрутизацію та ретрансляцію. На відміну від простої системи зв'язку між двома вузлами, mesh-мережа потребує самостійного прийняття рішень кожним вузлом щодо подальшої долі прийнятого пакета. Такий вузол повинен визначати, чи призначене повідомлення саме йому, чи потрібно передати його іншому вузлу, чи пакет необхідно відкинути через помилку, повторення або вичерпання допустимої кількості переходів. Тому основним завданням проєктування є побудова програмної структури, яка поєднує роботу з LoRa-модулем, перевірку службових полів пакета, ведення маршрутної інформації та підтримку механізмів захисту від дублювання повідомлень.

Програмне забезпечення розроблюваного вузла орієнтоване на використання мікроконтролера ESP32, який виконує функції керування, обробки даних і взаємодії з радіомодулем E32-400M30S. На рівні програмної реалізації ESP32 має забезпечувати ініціалізацію апаратних інтерфейсів, обмін із LoRa-модулем, збереження службових структур, аналіз прийнятих пакетів і формування рішень щодо передавання даних. Для програмування таких вбудованих систем можуть використовуватися засоби розробки для ESP32, зокрема офіційна документація ESP-IDF, у якій описано принципи налаштування середовища, конфігурації проєкту, складання та завантаження програмного забезпечення на мікроконтролер [10].

Взаємодія з радіомодулем у такій системі повинна бути відокремлена від логіки маршрутизації. Це означає, що модуль LoRa-зв'язку відповідає за

приймання та передавання пакетів, тоді як рішення про подальшу обробку прийнятого повідомлення приймається окремими програмними компонентами. Такий підхід робить структуру програмного забезпечення зрозумілішою та спрощує її подальше тестування. Для роботи з радіомодулями типу SX127x можуть застосовуватися бібліотеки бездротового зв'язку для вбудованих пристроїв, зокрема RadioLib, яка підтримує LoRa-модулі та надає програмні засоби для організації приймання й передавання повідомлень [11].

Кожен вузол LoRa mesh-мережі розглядається як автономний елемент системи, здатний виконувати три основні ролі: джерела повідомлення, кінцевого отримувача або проміжного ретранслятора. Якщо пакет адресований поточному вузлу, він обробляється локально. Якщо повідомлення призначене іншому вузлу, програмне забезпечення перевіряє можливість його подальшої передачі. У разі відсутності прямого зв'язку між відправником і отримувачем саме проміжні вузли забезпечують передавання пакета мережею. Такий принцип роботи підвищує гнучкість системи, але водночас потребує чіткої організації службових даних, маршрутної таблиці та правил відкидання некоректних або повторних пакетів.

На рисунку 2.1 подано основні складові програмного забезпечення вузла. Центральну частину структури утворюють модулі приймання пакетів, перевірки службових полів, фільтрації повторів, маршрутизації, ретрансляції, формування власних повідомлень, роботи з маршрутною таблицею та діагностики. Модуль ініціалізації забезпечує початкове налаштування ESP32 та LoRa-модуля E32-400M30S. Після приймання пакета система перевіряє його службові поля, визначає повторність, аналізує адресу отримувача та передає пакет на локальну обробку або ретрансляцію. Наявність модуля діагностики дозволяє фіксувати службову інформацію, що використовується під час налагодження й перевірки роботи вузла.

Вимоги до розроблюваного програмного забезпечення доцільно розглядати як сукупність функціональних і нефункціональних характеристик. Функціональні вимоги визначають дії, які має виконувати система, а

нефункціональні описують якість її роботи, стійкість, масштабованість, ефективність використання ресурсів і зручність налагодження. Для LoRa mesh-мережі такий поділ є важливим, оскільки система повинна не лише передавати повідомлення, а й забезпечувати коректну роботу в умовах нестабільного радіоканалу, обмеженої швидкості передавання та можливої втрати окремих пакетів [12].

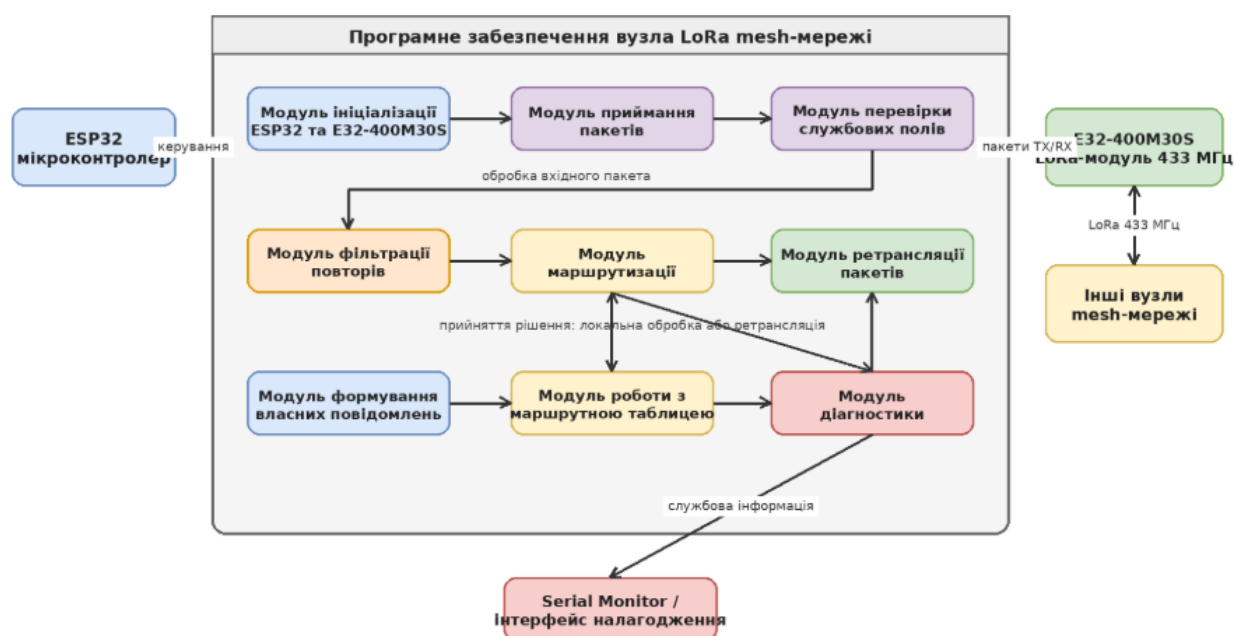


Рисунок 2.1 – Основні складові програмного забезпечення вузла LoRa mesh-мережі

Однією з основних функціональних вимог є ініціалізація вузла. Після запуску програмне забезпечення повинно налаштувати апаратні інтерфейси ESP32, підготувати LoRa-модуль до роботи, встановити параметри радіообміну, створити буфери приймання та передавання, а також ініціалізувати службові структури даних. До таких структур належать ідентифікатор вузла, маршрутна таблиця, журнал оброблених пакетів і параметри обмеження ретрансляції. Помилки на етапі ініціалізації можуть призвести до некоректної адресації, втрати пакетів або неможливості обміну між вузлами.

Наступною функціональною вимогою є приймання пакетів від інших вузлів мережі. Програмне забезпечення повинно контролювати наявність вхідних повідомлень, зчитувати дані з LoRa-модуля та передавати їх на перевірку. На цьому етапі важливо відокремити корисне навантаження від службових полів, оскільки саме службова інформація визначає маршрут пакета, його призначення, допустиму кількість переходів і можливість подальшої ретрансляції.

Важливою частиною обробки є перевірка адресної інформації. Кожен пакет повинен містити адресу відправника та адресу отримувача. Після приймання повідомлення вузол порівнює адресу отримувача зі своєю адресою. Якщо адреса збігається, пакет передається на локальну обробку. Якщо адреса не збігається, система перевіряє можливість передавання пакета далі. Такий механізм дозволяє уникнути обробки чужих повідомлень як локальних і забезпечує логіку багатохопової передачі.

Окремою функціональною вимогою є фільтрація повторних пакетів. У mesh-мережі одне й те саме повідомлення може надходити до вузла різними шляхами, особливо якщо застосовується ширококомовне або напівширокомовне поширення. Для уникнення повторної обробки кожен пакет повинен мати унікальний ідентифікатор. Вузол зберігає інформацію про вже оброблені повідомлення в журналі пакетів і відкидає дублікати. Це зменшує навантаження на радіоканал, обмежує кількість зайвих передач і знижує ризик зациклення повідомлень.

Ще однією необхідною функцією є контроль TTL. Значення TTL визначає максимально допустиму кількість переходів пакета між вузлами. Після кожної ретрансляції це значення зменшується на одиницю. Якщо TTL дорівнює нулю, пакет відкидається і не передається далі. Такий механізм є важливим для LoRa mesh-мережі, оскільки кожна зайва передача займає радіоканал, збільшує затримку доставки та підвищує енергоспоживання вузлів.

Маршрутна таблиця є основною службовою структурою, яка використовується під час вибору наступного вузла. Вона може містити адресу

вузла призначення, наступний вузол для передавання, кількість переходів до отримувача, час останнього оновлення запису та додаткові показники якості зв'язку. У спрощеній реалізації достатньо зберігати адресу призначення, адресу наступного переходу та кількість переходів. Така структура дозволяє зменшити обсяг службових даних і водночас забезпечити базову маршрутизацію пакетів у mesh-мережі.

На рисунку 2.2 показано послідовність проходження пакета через основні модулі програмного забезпечення. Пакет надходить через LoRa-модуль E32-400M30S до модуля приймання, після чого передається на перевірку. Далі модуль фільтрації повторів порівнює ідентифікатор пакета з журналом оброблених повідомлень. Якщо пакет є новим і коректним, він передається до модуля маршрутизації. Модуль маршрутизації використовує маршрутну таблицю, визначає подальший напрям передавання та приймає рішення про локальну обробку або ретрансляцію. Діагностичний журнал фіксує службові події, зокрема результати перевірки, рішення про маршрут і факт ретрансляції.

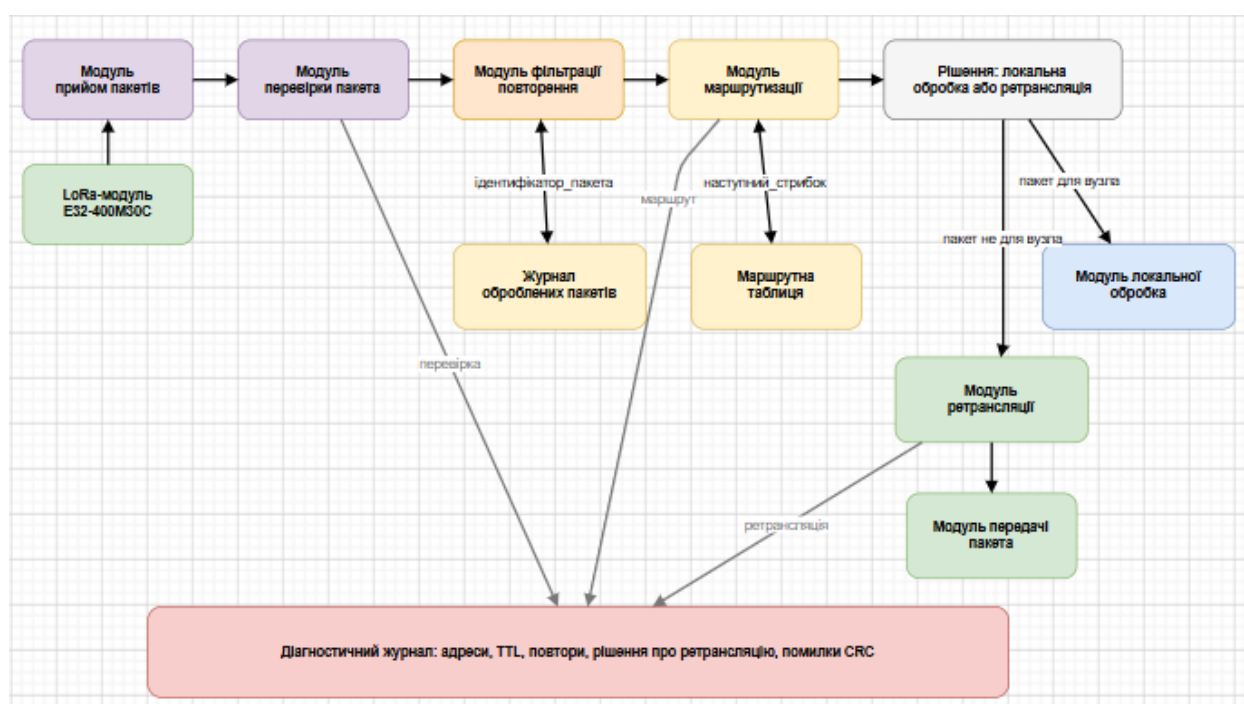


Рисунок 2.2 – Послідовність проходження пакета через основні модулі програмного забезпечення

Нефункціональні вимоги визначають умови, за яких програмне забезпечення має зберігати працездатність і коректність роботи. До таких вимог належать надійність, економне використання радіоканалу, раціональне використання пам'яті, масштабованість і зручність налагодження. Надійність передбачає, що вузол повинен продовжувати роботу за умов втрати окремих пакетів, тимчасового зникнення сусіднього вузла або погіршення якості радіоканалу. Система не повинна залежати від одного маршруту, а її логіка має допускати повторний пошук шляху або використання альтернативного переходу.

Економне використання радіоканалу є важливим через обмежену швидкість передавання LoRa. Надмірна кількість службових повідомлень або повторних ретрансляцій може погіршити роботу всієї мережі. Тому програмне забезпечення повинно мінімізувати широкомовне поширення, не передавати повторні пакети, використовувати TTL і не створювати зайвого службового трафіку. Раціональне використання пам'яті також має значення, оскільки журнал пакетів, маршрутна таблиця та буфери повідомлень займають ресурси мікроконтролера. Для цього доцільно застосовувати обмежені за розміром таблиці та видаляти застарілі записи.

Масштабованість системи означає можливість збільшення кількості вузлів без зміни основної логіки програмного забезпечення. Адресація, структура пакета та алгоритм ретрансляції повинні дозволяти додавати нові вузли без повного переписування програми. Простота налагодження забезпечується виведенням службової інформації через послідовний порт або інший інтерфейс. До такої інформації належать адреси вузлів, значення TTL, ідентифікатори пакетів, рішення про ретрансляцію, записи маршрутної таблиці та помилки перевірки контрольної суми.

На рисунку 2.3 подано алгоритм обробки пакета після запуску вузла. Спочатку виконується ініціалізація ESP32 та LoRa-модуля E32-400M30S, після чого вузол переходить у режим очікування пакета. Якщо пакет отримано, система приймає його та перевіряє контрольну суму. Некоректний пакет

відкидається. Якщо контрольна сума правильна, перевіряється, чи оброблявся цей пакет раніше. Повторне повідомлення також відкидається. Новий пакет передається на перевірку адреси призначення. Якщо адреса відповідає поточному вузлу, пакет обробляється локально. Якщо повідомлення призначене іншому вузлу, перевіряється TTL. За наявності допустимого значення TTL пакет передається наступному вузлу або поширюється ширококомовно з обмеженням кількості переходів.

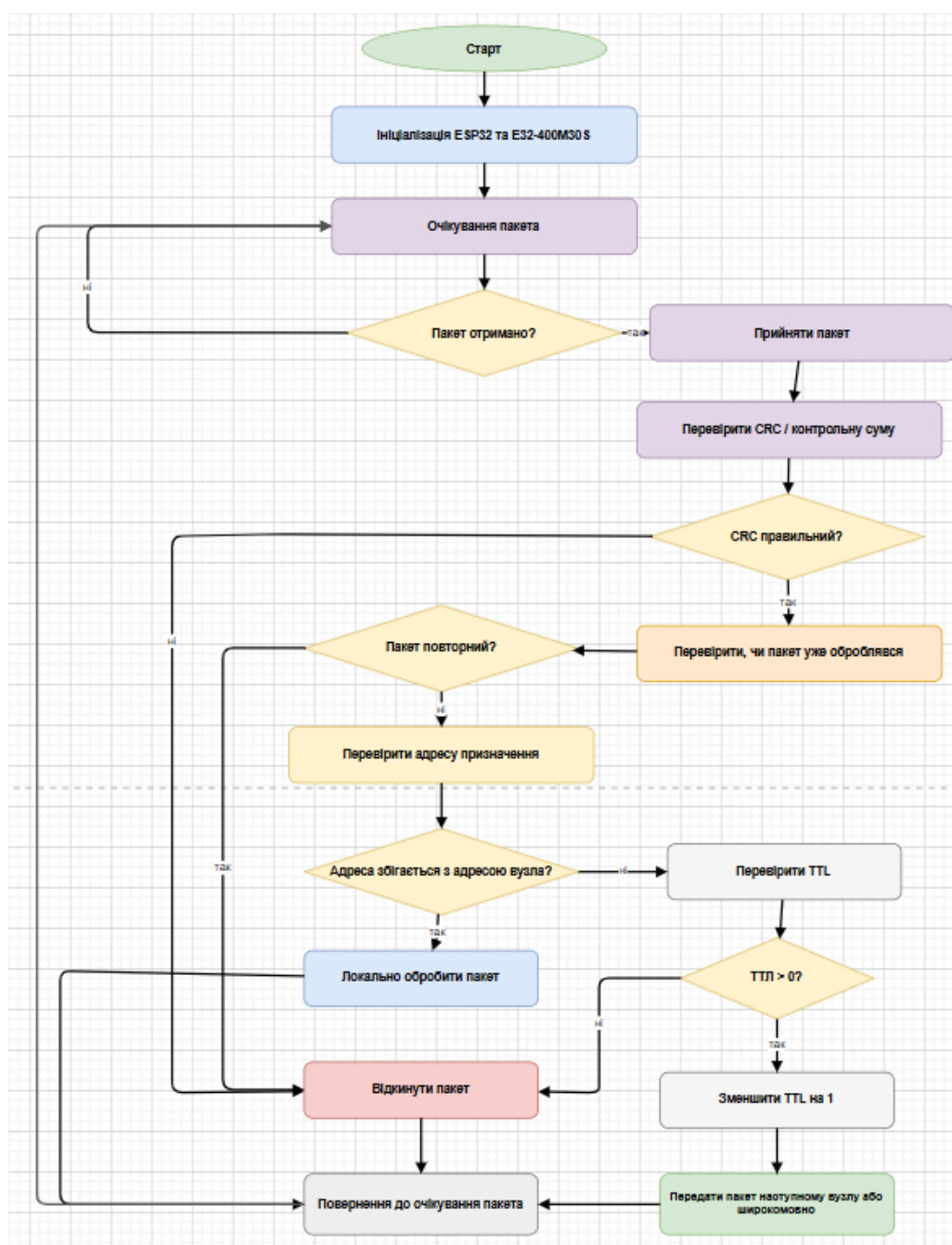


Рисунок 2.3 – Типовий сценарій обробки пакета вузлом LoRa mesh-мережі

Під час проектування важливо визначити структуру службового пакета. Такий пакет повинен містити набір полів, достатній для його ідентифікації, перевірки та маршрутизації. До базових полів належать ідентифікатор пакета, адреса відправника, адреса отримувача, адреса попереднього вузла, тип повідомлення, TTL, довжина корисних даних, корисне навантаження та контрольна сума. Наявність цих полів дозволяє визначати маршрут повідомлення, обмежувати кількість ретрансляцій, фільтрувати дублікати та перевіряти цілісність прийнятих даних.

Маршрутна таблиця використовується для збереження інформації про доступні напрями передавання. У ній можуть міститися адреса вузла призначення, адреса наступного вузла, кількість переходів, час останнього оновлення запису та додаткові показники якості зв'язку. Якщо запис у таблиці довго не оновлюється, він повинен вважатися застарілим. Такий підхід дозволяє зменшити ймовірність використання неактуальних маршрутів і підвищити стійкість мережі до зміни топології.

Для формалізації проєктних рішень у роботі використано UML-діаграми. Вони дозволяють подати систему не лише як набір програмних модулів, а і як сукупність сценаріїв взаємодії, компонентів та послідовностей виконання операцій. Використання UML є доцільним у процесі проектування програмного забезпечення, оскільки такі діаграми допомагають узгодити функції системи, структуру компонентів і логіку обробки даних до переходу до практичної реалізації [13].

На рисунку 2.4 подано взаємодію двох основних акторів із системою: користувача/адміністратора та іншого вузла mesh-мережі. Користувач або адміністратор виконує налаштування параметрів вузла, запускає його роботу та переглядає службову інформацію. Інший вузол mesh-мережі взаємодіє із системою через передавання або приймання пакета. Усередині системи відображено ключові сценарії: приймання пакета, перевірка контрольної суми, фільтрація повторів, перевірка адреси отримувача, локальна обробка

повідомлення або виконання ретрансляції. Діаграма показує зовнішні ролі та основні функції, які забезпечують роботу вузла в mesh-мережі.

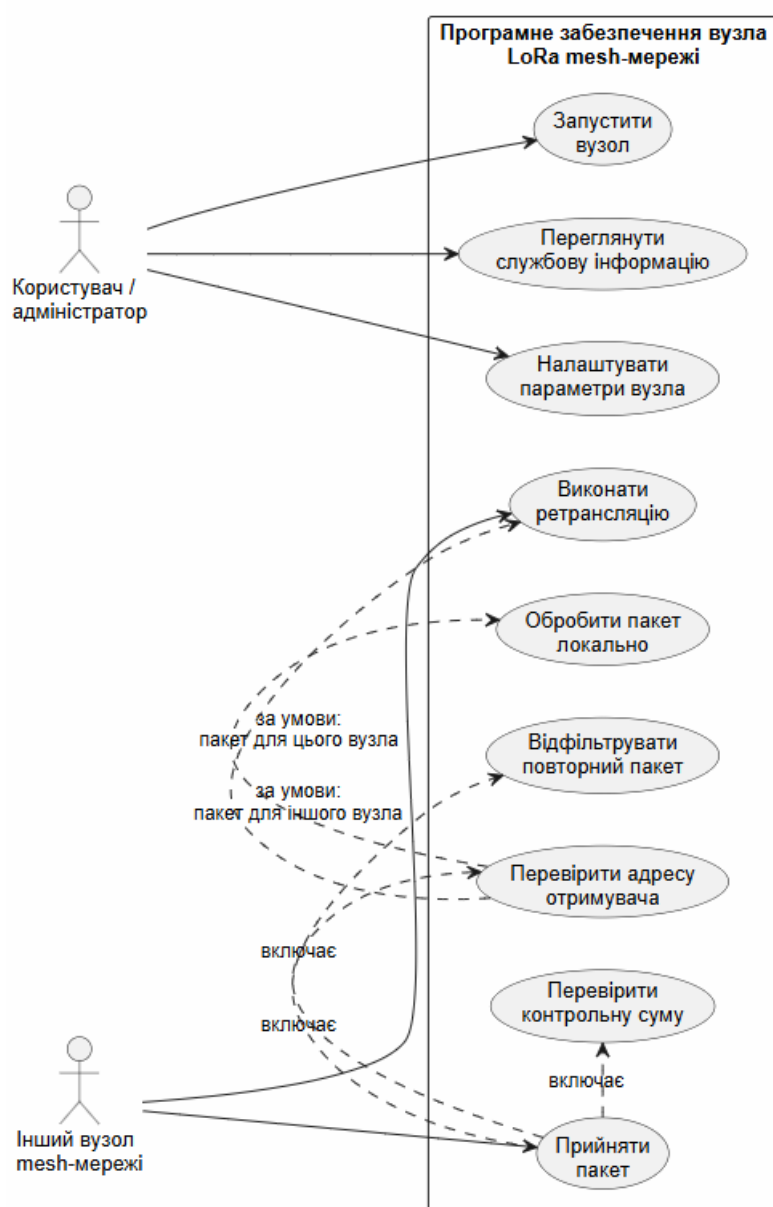


Рисунок 2.4 – UML-діаграма варіантів використання програмного забезпечення LoRa mesh-мережі

На рисунку 2.5 відображено узагальнену компонентну структуру програмного забезпечення. Центральним елементом є ядро програмного забезпечення вузла, яке координує роботу основних модулів. Модуль LoRa-зв'язку забезпечує взаємодію з радіомодулем, модуль перевірки пакета аналізує прийняті дані, а модуль маршрутизації визначає подальший напрям обробки

повідомлення. Для збереження службової інформації використовуються журнал пакетів і маршрутна таблиця. Модуль ретрансляції забезпечує передавання пакета іншому вузлу, а модуль діагностики отримує службові дані про стан вузла та процес маршрутизації.

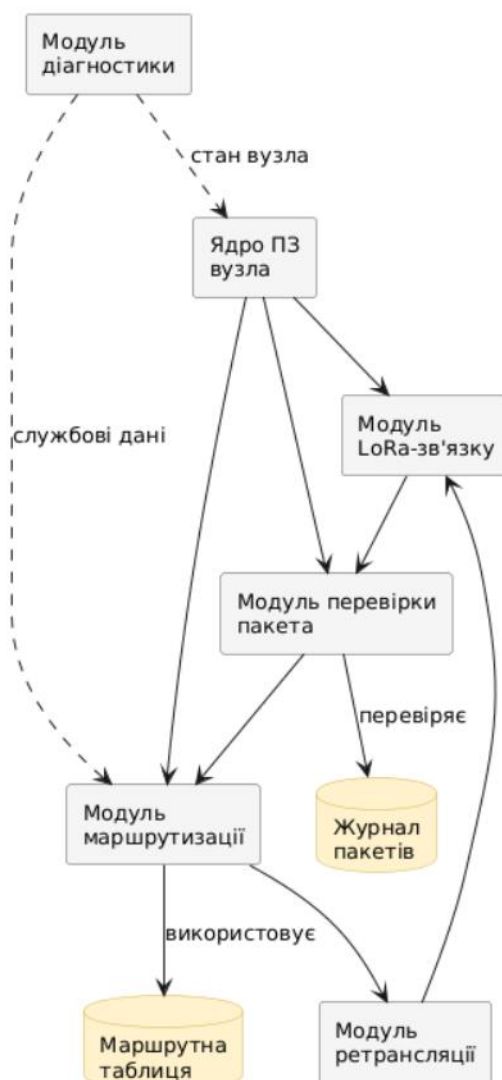


Рисунок 2.5 – UML-діаграма компонентів програмного забезпечення вузла LoRa mesh-мережі

На рисунку 2.6 показано порядок проходження пакета між основними учасниками процесу. Інший вузол mesh-мережі надсилає пакет через LoRa-модуль E32-400M30S. Далі прийняті дані передаються до модуля приймання, після чого пакет проходить перевірку коректності та повторності. Якщо повідомлення є некоректним або вже було оброблене, воно відкидається. Якщо

пакет коректний, він передається до модуля маршрутизації, який звертається до маршрутної таблиці. У випадку, коли пакет адресований поточному вузлу, він обробляється локально. Якщо повідомлення призначене іншому вузлу, воно передається до модуля ретрансляції та надсилається далі через LoRa-модуль.

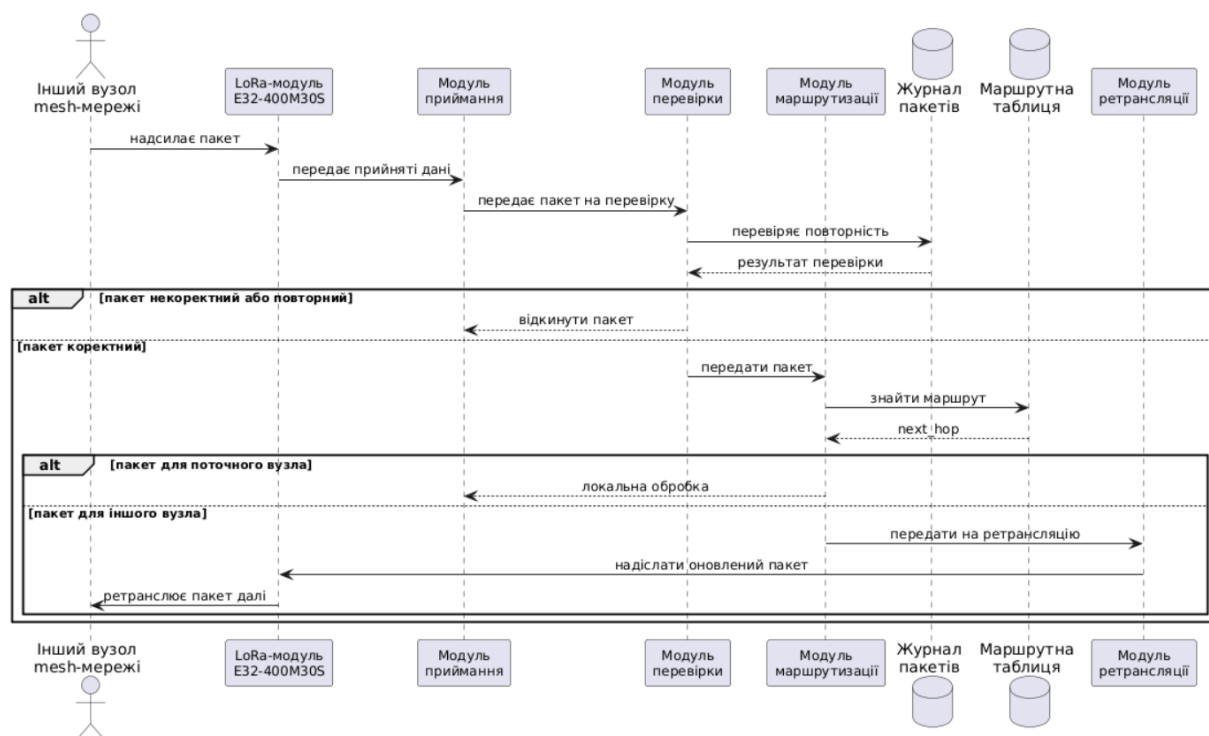


Рисунок 2.6 – UML-діаграма послідовності обробки та ретрансляції пакета

Таким чином, у підрозділі визначено основні вимоги до програмного забезпечення вузла LoRa mesh-мережі та подано його проектну структуру. Розроблювана система повинна забезпечувати ініціалізацію вузла, приймання пакетів, перевірку службових полів, фільтрацію повторів, контроль TTL, ведення маршрутної таблиці, локальну обробку повідомлень і ретрансляцію пакетів. Запропонована структура програмного забезпечення дозволяє розділити функції між окремими модулями, що спрощує реалізацію, тестування та подальше вдосконалення системи. UML-діаграми доповнюють опис проектування, оскільки формально показують сценарії використання, компонентну організацію та послідовність обробки пакета в LoRa mesh-мережі.

## **2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання**

Для реалізації програмного забезпечення маршрутизації даних у mesh-мережі з використанням технології LoRa необхідно обґрунтувати вибір апаратних засобів, програмного середовища, способу взаємодії між компонентами та алгоритмічн их механізмів, які забезпечуватимуть приймання, обробку, маршрутизацію і ретрансляцію пакетів. Особливістю такої системи є те, що вона повинна працювати в умовах обмеженої пропускної здатності радіоканалу, можливої втрати пакетів, зміни якості зв'язку між вузлами та відсутності постійної централізованої інфраструктури. Тому вибір засобів реалізації має враховувати не лише технічні характеристики окремих компонентів, але й можливість побудови на їх основі автономного вузла, здатного виконувати функції передавання, приймання та проміжної ретрансляції даних.

У межах даної кваліфікаційної роботи бакалавра як апаратну платформу для вузла mesh-мережі обрано мікроконтролер ESP32. Такий вибір є доцільним, оскільки ESP32 має достатній рівень продуктивності для реалізації логіки маршрутизації, обробки службових полів пакета, збереження маршрутної таблиці та ведення журналу прийнятих повідомлень. На відміну від простіших мікроконтролерів, ESP32 має більший обсяг оперативної пам'яті, вищу тактову частоту та розвинений набір периферійних інтерфейсів, що дає змогу використовувати його не лише як керуючий елемент для передавання даних, а як повноцінний обчислювальний вузол мережі. Для задачі LoRa mesh-маршрутизації це є важливим, оскільки мікроконтролер повинен виконувати кілька операцій послідовно: приймати пакет, перевіряти його структуру, аналізувати адресу отримувача, визначати повторність повідомлення, контролювати значення TTL і приймати рішення щодо локальної обробки або ретрансляції [14].

Важливою перевагою ESP32 є наявність апаратних інтерфейсів, які можуть бути використані для підключення зовнішнього радіомодуля та організації діагностичного виведення інформації. У розроблюваній системі мікроконтролер виконує роль центрального керуючого компонента, який координує роботу всіх програмних модулів. До його завдань належить ініціалізація радіомодуля, формування службових пакетів, збереження записів маршрутної таблиці, оновлення журналу оброблених повідомлень та передавання діагностичних даних під час тестування. Такий функціональний розподіл дозволяє розглядати ESP32 як основу вузла mesh-мережі, на якій може бути реалізована як базова, так і розширена логіка маршрутизації.

Як радіомодуль для організації бездротового зв'язку обрано E32-400M30S. Цей модуль працює у діапазоні 433/470 МГц, побудований на базі LoRa-рішення SX1278 і має максимальну потужність передавача 30 dBm, що відповідає 1 Вт. Такі характеристики є важливими для побудови мережі, у якій окремі вузли можуть розміщуватися на значній відстані один від одного. Використання частотного діапазону 433 МГц є доцільним для систем, де необхідно забезпечити зв'язок на відносно великих відстанях із передаванням невеликих обсягів даних. При цьому LoRa-модуль не виконує функції маршрутизації самостійно, а забезпечує фізичний рівень радіообміну, тоді як логіка обробки пакетів реалізується на ESP32 [14].

Застосування E32-400M30S у поєднанні з ESP32 дозволяє розділити функції між апаратними компонентами. Радіомодуль відповідає за передавання та приймання повідомлень у радіоканалі, а мікроконтролер виконує аналіз отриманих даних і приймає рішення щодо їх подальшої обробки. Такий підхід є зручним для реалізації mesh-мережі, оскільки структура пакета, правила ретрансляції, механізм фільтрації повторів і спосіб ведення маршрутної таблиці можуть змінюватися на рівні програмного забезпечення без необхідності заміни апаратної частини. Це також створює умови для подальшого вдосконалення системи, наприклад додавання нових типів службових повідомлень,

розширення маршрутної таблиці або впровадження складніших критеріїв вибору маршруту.

Для розробки програмного забезпечення доцільно використовувати Arduino IDE або PlatformIO. Arduino IDE є зручним середовищем для початкової реалізації, оскільки має просту структуру проєкту, зрозумілий процес компіляції та завантаження прошивки, а також широку підтримку бібліотек для роботи з мікроконтролерами. Такий варіант є доцільним на етапі первинного тестування, коли потрібно швидко перевірити працездатність окремих функцій: ініціалізацію плати, обмін із радіомодулем, приймання та передавання пакетів, виведення діагностичної інформації через послідовний порт. PlatformIO, у свою чергу, є зручнішим для структурованої розробки складнішого проєкту, оскільки дозволяє краще організувати файли, бібліотеки, параметри компіляції та конфігурацію цільової платформи.

З огляду на структуру розроблюваного програмного забезпечення, середовище розробки не повинно жорстко визначати архітектуру програми. Основну увагу необхідно приділити поділу програмної логіки на окремі модулі: модуль ініціалізації, модуль LoRa-зв'язку, модуль перевірки пакета, модуль маршрутизації, модуль ретрансляції, журнал оброблених пакетів і модуль діагностики. За такого підходу програмний код може бути спочатку реалізований у середовищі Arduino IDE, а за потреби перенесений до PlatformIO без зміни основної логіки. Це є важливим для кваліфікаційної роботи бакалавра, оскільки дозволяє поєднати простоту практичної реалізації з можливістю подальшого розвитку програмного продукту.

Обмін між ESP32 і LoRa-модулем доцільно організувати через апаратний інтерфейс, який підтримується обраною конфігурацією радіомодуля. У межах програмної архітектури роботу з радіомодулем необхідно винести в окремий модуль LoRa-зв'язку. Такий модуль повинен відповідати за ініціалізацію радіочастини, передавання сформованих пакетів, приймання вхідних повідомлень і передавання їх до модуля перевірки. Відокремлення цього рівня від логіки маршрутизації є важливим, оскільки дозволяє уникнути змішування

апаратно-залежного коду з алгоритмічною частиною. У результаті модуль маршрутизації працює не безпосередньо з радіомодулем, а з уже сформованим або прийнятим пакетом.

Важливим етапом проєктування є вибір структури службового пакета. У звичайній системі передавання між двома пристроями достатньо передати лише корисні дані, однак у mesh-мережі цього недостатньо. Пакет повинен містити службові поля, за допомогою яких вузол може визначити джерело повідомлення, кінцевого отримувача, попередній вузол, допустиму кількість переходів, тип повідомлення та коректність прийнятих даних. Орієнтовна структура пакета може включати такі поля: `packet_id`, `source_id`, `destination_id`, `previous_hop`, `ttn`, `message_type`, `payload` та `src`. Така структура забезпечує мінімально необхідний набір даних для реалізації маршрутизації та водночас не створює надмірного службового навантаження на LoRa-канал.

Для реалізації маршрутизації в розроблюваній системі доцільно використати спрощений алгоритм, що поєднує маршрутну таблицю та обмежене ширококомове поширення. Якщо вузол має запис у маршрутній таблиці для певного отримувача, пакет передається наступному вузлу, зазначеному в таблиці. Якщо маршрут відсутній, система може виконати обмежене ширококомове передавання з урахуванням TTL. Такий підхід є компромісом між простотою реалізації та функціональністю. Він не потребує складного службового обміну, характерного для повноцінних протоколів маршрутизації, але дозволяє організувати передавання даних через проміжні вузли.

Маршрутна таблиця в базовій реалізації повинна містити лише найнеобхідніші поля: адресу вузла призначення, адресу наступного вузла, кількість переходів і час останнього оновлення запису. За потреби вона може бути доповнена показниками якості зв'язку, зокрема RSSI або SNR. Проте для першої версії системи надмірне ускладнення таблиці є недоцільним, оскільки воно збільшує обсяг службових даних і складність програмної реалізації. Основним завданням є забезпечення коректного вибору наступного вузла та

можливість передачі пакета в умовах, коли повний маршрут до отримувача заздалегідь невідомий.

Механізм захисту від повторів реалізується за допомогою журналу оброблених пакетів. Після приймання повідомлення вузол перевіряє значення `packet_id`. Якщо такий ідентифікатор уже міститься в журналі, пакет вважається повторним і відкидається. Якщо повідомлення є новим, його ідентифікатор додається до журналу, а сам пакет передається на подальшу перевірку та маршрутизацію. Для економії пам'яті журнал не повинен зберігати необмежену кількість записів. Доцільно використовувати фіксований розмір журналу або механізм видалення застарілих записів.

Захист від зациклення повідомлень забезпечується використанням лічильника TTL. Кожного разу, коли вузол виконує ретрансляцію, значення TTL зменшується на одиницю. Якщо після перевірки виявлено, що TTL дорівнює нулю, пакет не передається далі. Такий механізм обмежує кількість переходів пакета мережею і запобігає ситуації, коли повідомлення нескінченно циркулює між вузлами через помилку маршрутизації або дублювання маршрутів. Поєднання журналу пакетів і TTL формує базовий рівень захисту від повторів і зациклення, що є необхідною умовою стабільної роботи LoRa mesh-мережі.

Окреме значення має діагностика роботи програмного забезпечення. Під час тестування вузол повинен виводити службову інформацію про прийняті та передані пакети, адресу відправника й отримувача, значення TTL, результат перевірки контрольної суми, наявність повтору, вибраний наступний вузол і рішення про локальну обробку або ретрансляцію. Така інформація дозволяє перевірити правильність роботи алгоритму без використання складного додаткового обладнання. На етапі практичної реалізації це особливо важливо, оскільки дає змогу швидко виявити помилки у структурі пакета, логіці маршрутизації або роботі з радіомодулем.

Таким чином, для вирішення поставленого завдання обрано апаратно-програмний підхід, у якому ESP32 використовується як керуючий і

обчислювальний елемент вузла, а E32-400M30S забезпечує бездротовий LoRa-зв'язок між вузлами. Програмна реалізація може виконуватися в Arduino IDE або PlatformIO, причому структура коду повинна залишатися модульною та придатною до подальшого розширення. Основу алгоритмічної частини становлять службовий пакет із полями адресації, маршрутна таблиця, журнал оброблених повідомлень і лічильник TTL. Обраний підхід є достатнім для реалізації базової LoRa mesh-мережі та створює основу для практичної реалізації програмного забезпечення в наступному розділі роботи.

## РОЗДІЛ 3

### РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ МАРШРУТИЗАЦІЇ ДАНИХ У MESH-МЕРЕЖІ З ВИКОРИСТАННЯМ LORA

#### 3.1 Практична реалізація об'єкта проєктування

Практична реалізація об'єкта проєктування була спрямована на створення програмної логіки вузла LoRa mesh-мережі, яка може виконувати обробку службових пакетів, перевірку їхньої коректності, вибір подальшого напрямку передавання та налагоджувальне виведення результатів роботи. Основна увага приділялася реалізації тих частин алгоритму, які визначають поведінку вузла під час приймання повідомлення: перевірки контрольної суми, аналізу адреси отримувача, контролю повторів, роботі з маршрутною таблицею та обмеженню кількості переходів за допомогою TTL.

Програмна логіка була побудована за модульним принципом. Окремо реалізовано структуру службового пакета, функцію обчислення контрольної суми, пошук наступного вузла в маршрутній таблиці, журнал оброблених пакетів, функцію обробки прийнятого повідомлення та механізм ретрансляції. Такий підхід дозволяє змінювати окремі частини алгоритму без повного переписування програми, що є важливим для подальшого розширення розроблюваного програмного забезпечення.

Єдиним форматом даних, з яким працюють основні функції програми, є службовий пакет. Він містить мінімальний набір полів, необхідних для перевірки, маршрутизації та подальшої обробки повідомлення. Реалізацію структури службового пакета наведено у лістингу 3.1.

Лістинг 3.1 – Структура службового пакета LoRa mesh-мережі

---

```
struct MeshPacket {
    uint16_t packetId;
    uint8_t sourceNode;
    uint8_t destinationNode;
    uint8_t previousNode;
    uint8_t ttl;
```

```

    String payload;
    uint8_t crc;
};
MeshPacket createPacket(uint16_t id, uint8_t src, uint8_t dst,
String data) {
    MeshPacket packet = { id, src, dst, src, 4, data, 0 };
    return packet;
}

```

---

кінець лістингу 3.1

Використання окремої структури MeshPacket дає змогу передавати пакет між функціями як єдиний об'єкт. Це спрощує подальшу обробку, оскільки службові поля та корисні дані не розділяються на окремі незалежні змінні.

Для виявлення пошкоджених повідомлень у програмі реалізовано функцію обчислення контрольної суми. Вона використовується як під час формування пакета, так і під час його приймання. Фрагмент реалізації наведено у лістингу 3.2.

### Лістинг 3.2 – Обчислення контрольної суми службового пакета

---

```

uint8_t calculatePacketCrc (const MeshPacket &packet)
{
    uint8_t crcValue = 0;
    crcValue ^= lowByte(packet.packetId);
    crcValue ^= highByte(packet.packetId);
    crcValue ^= packet.sourceNode;
    crcValue ^= packet.destinationNode;
    crcValue ^= packet.previousNode;
    crcValue ^= packet.ttl;
    for (uint16_t i = 0; i < packet.payload.length(); i++) {
        crcValue ^= packet.payload[i];
    }
    return crcValue;
}

```

---

Кінець лістингу 3.2

Ця функція використовується перед прийняттям рішення про подальшу обробку пакета. Якщо контрольне значення не збігається з очікуванням, повідомлення не передається до блоку маршрутизації.

Для вибору подальшого напрямку передавання використовується маршрутна таблиця. У програмній реалізації вона подана як масив записів, де

кожен запис містить кінцевий вузол, наступний вузол і кількість переходів. Реалізацію маршрутної таблиці та пошуку наступного вузла наведено у лістингу 3.3.

### Лістинг 3.3 – Пошук наступного вузла в маршрутній таблиці

---

```

struct RouteRecord {
    uint8_t destination;
    uint8_t nextNode;
    uint8_t hopCount;
};
RouteRecord routeTable[] = {
    {3, 2, 2},
    {4, 3, 3},
    {1, 4, 1}
};
uint8_t findNextNode(uint8_t destination) {
    for (uint8_t i = 0; i < 3; i++) {
        if (routeTable[i].destination == destination) return
routeTable[i].nextNode;
    }
    return 0;
}

```

---

кінець лістингу 3.3

Винесення пошуку маршруту в окрему функцію дозволяє не змішувати логіку обробки пакета з логікою вибору наступного вузла. Надалі ця функція може бути розширена, наприклад, з урахуванням якості сигналу або часу останнього оновлення маршруту.

Для уникнення повторної обробки повідомлень використовується журнал пакетів. У ньому зберігаються ідентифікатори пакетів, які вже були прийняті вузлом. Реалізацію цього механізму наведено у лістингу 3.4.

### Лістинг 3.4 – Журнал оброблених пакетів

---

```

const uint8_t logSize = 10;
uint16_t packetLog[logSize];
uint8_t logPosition = 0;
bool isProcessed(uint16_t packetId) {
    for (uint8_t i = 0; i < logSize; i++) {
        if (packetLog[i] == packetId) return true;
    }
}

```

---

```

}
return false;
}
void addToLog(uint16_t packetId) {
packetLog[logPosition] = packetId;
logPosition = (logPosition + 1) % logSize;
}

```

---

кінець лістингу 3.4

Журнал реалізовано з фіксованим розміром, що дозволяє обмежити використання пам'яті мікроконтролера. Циклічне оновлення позиції запису дає змогу замінювати старі ідентифікатори новими без додаткового очищення масиву.

Основна функція обробки пакета об'єднує перевірку контрольної суми, перевірку повторності, аналіз адреси отримувача та виклик ретрансляції. Її скорочений варіант наведено у лістингу 3.5.

### Лістинг 3.5 – Обробка прийнятого пакета вузлом мережі

---

```

void handlePacket(MeshPacket packet, uint8_t currentNode) {
    if (calculatePacketCrc(packet) != packet.crc) {
        Serial.println("Пакет пошкоджено");
        return;
    }
    if (isProcessed(packet.packetId)) {
        Serial.println("Повторний пакет");
        return;
    }
    addToLog(packet.packetId);
    if (packet.destinationNode == currentNode) {
        Serial.println("Пакет оброблено локально");
        Serial.println(packet.payload);
        return;
    }
    forwardPacket(packet, currentNode);
}

```

---

кінець лістингу 3.5

Ця функція визначає загальний порядок роботи вузла після приймання повідомлення. Пакет або завершує обробку на поточному вузлі, або передається до функції ретрансляції.

Ретрансляція реалізована окремою функцією. У ній перевіряється значення TTL, виконується пошук наступного вузла, оновлюється адреса попереднього вузла та повторно обчислюється контрольна сума. Фрагмент реалізації наведено у лістингу 3.6.

### Лістинг 3.6 – Ретрансляція пакета з оновленням TTL

---

```
void forwardPacket(MeshPacket packet, uint8_t currentNode) {
    if (packet.ttl == 0) {
        Serial.println("TTL вичерпано");
        return;
    }
    uint8_t nextNode = findNextNode(packet.destinationNode);
    if (nextNode == 0) {
        Serial.println("Маршрут не знайдено");
        return;
    }
    packet.ttl--;
    packet.previousNode = currentNode;
    packet.crc = calculatePacketCrc(packet);
    Serial.print("Передавання до вузла ");
    Serial.println(nextNode);
}
```

---

кінець лістингу 3.6

Винесення ретрансляції в окремий блок спрощує подальшу зміну алгоритму вибору наступного вузла. Наприклад, у майбутньому замість статичної маршрутної таблиці може бути використано динамічне оновлення маршрутів.

Для налагодження роботи програмного забезпечення використовується виведення службових параметрів у Serial Monitor. Такий механізм дає змогу контролювати стан пакета на різних етапах його обробки. Функцію діагностичного виведення наведено у лістингу 3.7.

### Лістинг 3.7 – Діагностичне виведення параметрів пакета

---

```
void printPacketInfo(const MeshPacket &packet) {
    Serial.print("ID=");
    Serial.print(packet.packetId);
    Serial.print(", SRC=");
```

```

Serial.print(packet.sourceNode);
Serial.print(", DST=");
Serial.print(packet.destinationNode);
Serial.print(", PREV=");
Serial.print(packet.previousNode);
Serial.print(", TTL=");
Serial.print(packet.ttl);
Serial.print(", DATA=");
Serial.print(packet.payload);
Serial.print(", CRC=");
Serial.println(packet.crc);
}

```

---

кінець лістингу 3.7

Діагностичний вивід використовується під час перевірки роботи алгоритму, оскільки дозволяє фіксувати не лише кінцевий результат, а й проміжні стани пакета. Саме на основі такого виведення виконано тестування, подане в підрозділі 3.2.

Таким чином, практична реалізація об'єкта проєктування складається з набору взаємопов'язаних програмних механізмів, які забезпечують базову логіку роботи вузла LoRa mesh-мережі. Реалізовані фрагменти охоплюють формування службового пакета, перевірку контрольної суми, роботу з маршрутною таблицею, фільтрацію повторних повідомлень, обробку прийнятого пакета, ретрансляцію та діагностичне виведення. Така структура створює основу для подальшого розширення програмного забезпечення та його інтеграції з апаратним радіомодулем E32-400M30S.

### 3.2 Тестування та налагодження програмно забезпечення

Після розробки основних програмних механізмів вузла LoRa mesh-мережі виконано тестування алгоритмічної частини програмного забезпечення. Основна увага під час перевірки приділялася правильності роботи логіки маршрутизації, обробці службових полів пакета, контролю кількості переходів, перевірці адреси отримувача та відкиданню некоректних повідомлень. Тестування виконувалося на мікроконтролері ESP32 у середовищі Arduino IDE

з виведенням результатів у Serial Monitor. Такий підхід дав змогу простежити послідовність проходження пакета між вузлами mesh-мережі та перевірити реакцію програмного забезпечення на типові ситуації, які можуть виникати під час роботи системи.

У тестовому режимі було задано декілька вузлів mesh-мережі, службові параметри пакета, маршрутні зв'язки між вузлами та механізми перевірки коректності повідомлення. До службових параметрів пакета належать ідентифікатор пакета ID, адреса відправника SRC, адреса отримувача DST, адреса попереднього вузла PREV, лічильник переходів TTL та контрольна сума CRC. Наявність таких полів дозволяє визначити походження пакета, його кінцевого отримувача, попередній вузол маршруту, допустиму кількість ретрансляцій і коректність прийнятих даних.

Перший тест був спрямований на перевірку проходження пакета в кільцевій топології mesh-мережі. У цьому сценарії використано послідовність передавання (1-2-3-4-1), а прямий зв'язок між вузлом 1 і вузлом 3 відсутній. Пакет створюється на вузлі 1 з параметрами ID=101, SRC=1, DST=5, TTL=4. Вузол 5 у цьому тесті задано як умовний недоступний отримувач, що дозволяє перевірити поведінку програмного забезпечення у випадку, коли пакет не може бути доставлений до кінцевого вузла за наявним маршрутом.

Під час проходження пакета кожен проміжний вузол виконує перевірку контрольної суми, визначає, що пакет є новим, зменшує значення TTL і передає його наступному вузлу. У результаті пакет проходить через вузли 2, 3 і 4, після чого повертається до початкового вузла 1. Програмне забезпечення визначає факт повернення пакета до початкового вузла як ознаку можливого зациклення та відкидає пакет. Результат перевірки проходження пакета в кільцевій LoRa mesh-мережі зображено на рисунку 3.1.

```

Output  gdb-server  Serial Monitor  X
Message (Enter to send message to 'ESP32 D... New Line 115200 baud

-----
[ІНІЦІАЛІЗАЦІЯ] Система готова до роботи
[ІНІЦІАЛІЗАЦІЯ] Журнал пакетів очищено
[ІНІЦІАЛІЗАЦІЯ] Маршрутну таблицю завантажено

Топологія mesh-мережі:
ВУЗОЛ 1 -> ВУЗОЛ 2 -> ВУЗОЛ 3 -> ВУЗОЛ 4 -> ВУЗОЛ 1
Прямий зв'язок ВУЗОЛ 1 -> ВУЗОЛ 3 відсутній

===== Перевірка проходження пакета по кільцю =====
[ПАКЕТ] ВУЗОЛ 1 створив пакет: ID=101, SRC=1, DST=5, TTL=4
[ПРИМІТКА] Кінцевий ВУЗОЛ 5 недоступний у маршрутній таблиці
[ПЕРЕДАВАННЯ] ВУЗОЛ 1 -> ВУЗОЛ 2
[ПРИЙМАННЯ] ВУЗОЛ 2 отримав пакет ID=101
[ПЕРЕВІРКА] ВУЗОЛ 2: CRC коректна, пакет новий
[МАРШРУТИЗАЦІЯ] ВУЗОЛ 2: TTL 4 -> 3
[МАРШРУТИЗАЦІЯ] ВУЗОЛ 2: наступний вузол 3
[ПЕРЕДАВАННЯ] ВУЗОЛ 2 -> ВУЗОЛ 3
[ПРИЙМАННЯ] ВУЗОЛ 3 отримав пакет ID=101
[ПЕРЕВІРКА] ВУЗОЛ 3: CRC коректна, пакет новий
[МАРШРУТИЗАЦІЯ] ВУЗОЛ 3: TTL 3 -> 2
[МАРШРУТИЗАЦІЯ] ВУЗОЛ 3: наступний вузол 4
[ПЕРЕДАВАННЯ] ВУЗОЛ 3 -> ВУЗОЛ 4
[ПРИЙМАННЯ] ВУЗОЛ 4 отримав пакет ID=101
[ПЕРЕВІРКА] ВУЗОЛ 4: CRC коректна, пакет новий
[МАРШРУТИЗАЦІЯ] ВУЗОЛ 4: TTL 2 -> 1
[МАРШРУТИЗАЦІЯ] ВУЗОЛ 4: наступний вузол 1
[ПЕРЕДАВАННЯ] ВУЗОЛ 4 -> ВУЗОЛ 1
[ПРИЙМАННЯ] ВУЗОЛ 1 повторно отримав пакет ID=101
[ЗАХИСТ] Пакет повернувся до початкового вузла
[ЗАХИСТ] Виявлено можливе зациклення, пакет відкинуто

===== Додаткові результати перевірки =====
[TTL] Лічильник TTL обмежує кількість переходів пакета
[ЖУРНАЛ] ID пакета запобігає повторній обробці
[РЕЗУЛЬТАТ] Логіку маршрутизації перевірено успішно
=====
Ln 67, Col 2  ESP32 Dev Module on COM7 5

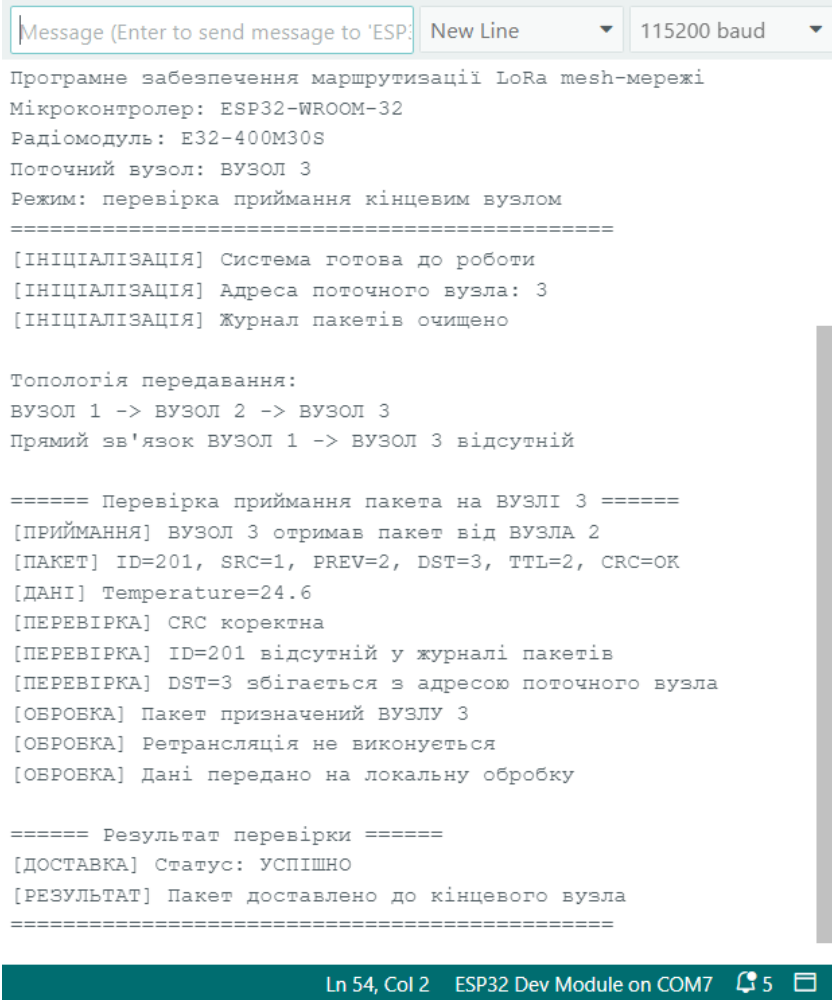
```

Рисунок 3.1 – Результат перевірки проходження пакета в кільцевій LoRa mesh-мережі

Другий тест був спрямований на перевірку приймання пакета кінцевим вузлом. У цьому сценарії використано топологію (1-2-3), де вузол 1 не має прямого зв'язку з вузлом 3. Поточним вузлом під час перевірки є вузол 3. У службових полях пакета зазначено ID=201, SRC=1, PREV=2, DST=3, TTL=2, CRC=OK. Це означає, що початковим відправником є вузол 1, попереднім вузлом маршруту є вузол 2, а кінцевим отримувачем є вузол 3.

Після приймання пакета вузол 3 виконує перевірку контрольної суми та встановлює, що CRC є коректною. Далі перевіряється журнал пакетів, у якому ідентифікатор ID=201 відсутній, тому пакет не є повторним. Наступним етапом

перевіряється адреса отримувача. Оскільки значення DST=3 збігається з адресою поточного вузла, програмне забезпечення визначає, що пакет призначений саме цьому вузлу. У такому випадку ретрансляція не виконується, а дані передаються на локальну обробку. Результат перевірки приймання пакета кінцевим вузлом LoRa mesh-мережі зображено на рисунку 3.2.



```

Message (Enter to send message to 'ESP32') New Line 115200 baud
Програмне забезпечення маршрутизації LoRa mesh-мережі
Мікроконтролер: ESP32-WROOM-32
Радіомодуль: E32-400M30S
Поточний вузол: ВУЗОЛ 3
Режим: перевірка приймання кінцевим вузлом
=====
[ІНІЦІАЛІЗАЦІЯ] Система готова до роботи
[ІНІЦІАЛІЗАЦІЯ] Адреса поточного вузла: 3
[ІНІЦІАЛІЗАЦІЯ] Журнал пакетів очищено

Топологія передавання:
ВУЗОЛ 1 -> ВУЗОЛ 2 -> ВУЗОЛ 3
Прямий зв'язок ВУЗОЛ 1 -> ВУЗОЛ 3 відсутній

===== Перевірка приймання пакета на ВУЗЛІ 3 =====
[ПРИЙМАННЯ] ВУЗОЛ 3 отримав пакет від ВУЗЛА 2
[ПАКЕТ] ID=201, SRC=1, PREV=2, DST=3, TTL=2, CRC=OK
[ДАНІ] Temperature=24.6
[ПЕРЕВІРКА] CRC коректна
[ПЕРЕВІРКА] ID=201 відсутній у журналі пакетів
[ПЕРЕВІРКА] DST=3 збігається з адресою поточного вузла
[ОБРОБКА] Пакет призначений ВУЗЛУ 3
[ОБРОБКА] Ретрансляція не виконується
[ОБРОБКА] Дані передано на локальну обробку

===== Результат перевірки =====
[ДОСТАВКА] Статус: УСПІШНО
[РЕЗУЛЬТАТ] Пакет доставлено до кінцевого вузла
=====
Ln 54, Col 2 ESP32 Dev Module on COM7 5

```

Рисунок 3.2 – Результат перевірки приймання пакета кінцевим вузлом  
LoRa mesh-мережі

Третій тест був спрямований на перевірку реакції програмного забезпечення на пошкоджений пакет. Поточним вузлом у цьому сценарії є вузол 2, який отримує пакет від вузла 1. У службових полях пакета зазначено ID=301, SRC=1, PREV=1, DST=3, TTL=3, а стан контрольної суми визначено як

CRC=ERROR. Такий сценарій дозволяє перевірити, чи буде некоректний пакет передано до модуля маршрутизації або ретрансльовано далі.

Після приймання повідомлення вузол 2 виконує перевірку контрольної суми. Оскільки очікуване значення CRC не збігається з отриманим, програмне забезпечення визначає пакет як пошкоджений. У результаті пакет не передається до модуля маршрутизації, не ретрансльовується до наступного вузла та відкидається. Результат перевірки відкидання пакета з помилкою CRC зображено на рисунку 3.3.

```

Output  gdb-server  Serial Monitor X
Message (Enter to send message to 'ESP: New Line 115200 baud

=====
[ІНІЦІАЛІЗАЦІЯ] Система готова до роботи
[ІНІЦІАЛІЗАЦІЯ] Адреса поточного вузла:2
[ІНІЦІАЛІЗАЦІЯ] Журнал пакетів очищено

Топологія передавання:
ВУЗОЛ 1 -> ВУЗОЛ 2 -> ВУЗОЛ 3

===== Перевірка пакета з помилкою CRC =====
[ПРИЙМАННЯ] ВУЗОЛ 2 отримав пакет від ВУЗЛА 1
[ПАКЕТ] ID=301, SRC=1, PREV=1, DST=3, TTL=3, CRC=ERROR
[ДАНИ] Temperature=24.6
[ПЕРЕВІРКА] Виконується перевірка контрольної суми CRC
[ПЕРЕВІРКА] Очікуване значення CRC не збігається з отриманим
[ПОМИЛКА] Виявлено пошкодження пакета
[ОБРОВКА] Пакет не передається до модуля маршрутизації
[ОБРОВКА] Ретрансляція не виконується
[ОБРОВКА] Пакет відкинута

===== Результат перевірки =====
[РЕЗУЛЬТАТ] Пошкоджений пакет успішно виявлено
[РЕЗУЛЬТАТ] Некоректний пакет не був ретрансльований
=====

Ln 50, Col 2  ESP32 Dev Module on COM7 5

```

Рисунок 3.3 – Результат перевірки відкидання пакета з помилкою CRC

За результатами тестування встановлено, що розроблена алгоритмічна частина програмного забезпечення коректно обробляє основні ситуації, характерні для LoRa mesh-мережі. Програмна логіка забезпечує проходження пакета через проміжні вузли, перевірку адреси кінцевого отримувача, локальну обробку повідомлення, контроль кількості переходів за допомогою TTL,

виявлення можливого зациклення та відкидання пошкоджених пакетів. Отримані результати підтверджують працездатність базових механізмів маршрутизації та можуть бути використані як основа для подальшого апаратного випробування системи з використанням радіомодуля E32-400M30S.

### **3.3 Захист інформаційно-комп'ютерної системи**

Захист інформаційно-комп'ютерної системи є важливою складовою розробки програмного забезпечення для LoRa mesh-мережі, оскільки така система функціонує в бездротовому середовищі та передбачає передавання пакетів між декількома вузлами. На відміну від дротових систем, у яких канал передавання фізично обмежений, бездротова мережа є більш відкритою до впливу зовнішніх факторів. До таких факторів належать перехоплення повідомлень, повторне передавання раніше прийнятих пакетів, пошкодження службових полів, навмисне або випадкове створення некоректних повідомлень, а також надмірна кількість ретрансляцій, що може призвести до перевантаження радіоканалу.

У межах розроблюваної LoRa mesh-мережі захист розглядається не лише як криптографічне приховування даних, а як комплекс програмних механізмів, спрямованих на забезпечення коректності обробки пакетів, обмеження повторів, контроль цілісності та запобігання циклічному поширенню повідомлень. Такий підхід є доцільним для першої версії системи, оскільки основним завданням є забезпечення стійкої роботи маршрутизації та недопущення обробки некоректних або повторних пакетів. Тому базовий рівень захисту реалізовано через перевірку контрольної суми, аналіз службових полів, журнал оброблених пакетів і контроль TTL.

Однією з основних загроз для LoRa mesh-мережі є пошкодження пакета під час передавання. Причинами цього можуть бути завади в радіоканалі, низький рівень сигналу, нестабільне живлення вузла або помилки під час формування повідомлення. Якщо пошкоджений пакет буде переданий до

модуля маршрутизації, це може призвести до неправильного вибору наступного вузла, некоректної локальної обробки або повторної ретрансляції хибних даних. Для зменшення такого ризику в програмному забезпеченні передбачено перевірку контрольної суми перед виконанням подальших дій із пакетом.

Перевірка CRC виконується на початковому етапі обробки повідомлення. Якщо отримане контрольне значення не збігається з розрахованим, пакет не передається до модуля маршрутизації та не ретранслюється іншим вузлом. Таким чином, пошкоджене повідомлення зупиняється на вузлі, який його прийняв. Фрагмент програмної перевірки контрольної суми наведено у лістингу 3.8.

---

#### Лістинг 3.8 – Перевірка цілісності пакета за контрольною сумою

---

```
bool checkPacketIntegrity(const MeshPacket &packet) {
    uint8_t expectedCrc = calculatePacketCrc(packet);
    if (expectedCrc != packet.crc) {
        Serial.println("[ЗАХИСТ] Помилка CRC");
        Serial.println("[ЗАХИСТ] Пакет відкинуто");
        return false;
    }
    Serial.println("[ЗАХИСТ] CRC коректна");
    return true;
}
```

---

кінець лістингу 3.8

Наведений фрагмент використовується як перший логічний бар'єр під час обробки пакета. Це дозволяє не виконувати зайві операції з повідомленням, цілісність якого не підтверджена. Такий механізм не забезпечує захисту від цілеспрямованої підміни пакета, однак він є необхідним базовим засобом контролю помилок у бездротовому каналі.

Другою важливою загрозою є повторне надходження одного й того самого пакета. У mesh-мережі така ситуація є достатньо ймовірною, оскільки повідомлення може проходити різними маршрутами або повертатися до вже пройденого вузла через кільцеву топологію. Без додаткового контролю вузол може повторно обробити одне й те саме повідомлення або знову передати його

далі. Це збільшує навантаження на радіоканал, створює зайві передачі та може спричинити циклічне поширення пакета.

Для обмеження повторної обробки у програмному забезпеченні використовується журнал ідентифікаторів пакетів. Кожне повідомлення має власне поле `packetId`, яке записується до журналу після успішної первинної перевірки. Якщо пакет із таким самим ідентифікатором надходить повторно, він відкидається без ретрансляції. Приклад реалізації такого захисту наведено у лістингу 3.9.

### Лістинг 3.9 – Фільтрація повторних пакетів

---

```
bool checkReplayProtection(uint16_t packetId) {
    if (isProcessed(packetId)) {
        Serial.println("[ЗАХИСТ] Повторний пакет");
        Serial.println("[ЗАХИСТ] Обробку зупинено");
        return false;
    }
    addToLog(packetId);
    Serial.println("[ЗАХИСТ] Пакет додано до журналу");
    return true;
}
```

---

кінець лістингу 3.9

Застосування журналу пакетів дозволяє зменшити кількість повторних повідомлень у мережі та підвищити стійкість маршрутизації. Для мікроконтролерної реалізації важливо, щоб такий журнал мав обмежений розмір, оскільки оперативна пам'ять ESP32 не повинна витрачатися на необмежене зберігання службових даних. Тому доцільним є використання циклічного журналу, у якому старі записи поступово замінюються новими.

Окрему роль у захисті системи відіграє лічильник TTL. Це службове поле визначає максимально допустиму кількість переходів пакета між вузлами. Після кожної ретрансляції значення TTL зменшується на одиницю. Якщо значення дорівнює нулю, пакет відкидається. Такий механізм не дозволяє повідомленню нескінченно переміщуватися мережею у випадку помилки маршрутизації або повернення до початкового вузла.

У розроблюваній системі контроль TTL поєднується з перевіркою маршруту. Перед ретрансляцією вузол визначає, чи дозволено передавати пакет далі, а також чи існує запис про наступний вузол у маршрутній таблиці. Якщо хоча б одна з умов не виконується, ретрансляція не здійснюється. Фрагмент перевірки умов безпечної ретрансляції наведено у лістингу 3.10.

### Лістинг 3.10 – Контроль TTL перед ретрансляцією пакета

---

```
bool allowPacketForwarding(MeshPacket &packet, uint8_t
currentNode) {
    if (packet.ttl == 0) {
        Serial.println("[ЗАХИСТ] TTL вичерпано");
        return false;
    }
    uint8_t nextNode = findNextNode(packet.destinationNode);
    if (nextNode == 0) {
        Serial.println("[ЗАХИСТ] Маршрут відсутній");
        return false;
    }
    packet.ttl--;
    packet.previousNode = currentNode;
    packet.crc = calculatePacketCrc(packet);
    return true;
}
```

---

кінець лістингу 3.10

Такий механізм забезпечує контрольоване поширення пакета в mesh-мережі. Після зміни TTL та адреси попереднього вузла контрольна сума перераховується повторно, оскільки службові поля пакета були змінені. Це дозволяє наступному вузлу перевірити вже оновлений стан повідомлення та коректно прийняти рішення щодо подальшої обробки.

Крім перевірки CRC, журналу пакетів і TTL, важливим елементом захисту є аналіз адресних полів. У службовому пакеті використовуються адреси відправника, отримувача та попереднього вузла. Це дозволяє визначити походження повідомлення, його напрям руху та призначення. Якщо адреса отримувача збігається з адресою поточного вузла, пакет передається на локальну обробку. Якщо адреса не збігається, вузол перевіряє можливість

ретрансляції. Такий підхід запобігає ситуації, коли вузол помилково обробляє чуже повідомлення як власне.

Захист інформаційно-комп'ютерної системи також передбачає контроль налагоджувального виведення. Під час тестування Serial Monitor використовується для відображення службової інформації про проходження пакетів, значення TTL, результат перевірки CRC та рішення щодо ретрансляції. Однак у реальних умовах експлуатації надмірне виведення службових даних може розкривати структуру мережі, адреси вузлів і логіку маршрутизації. Тому діагностичний режим доцільно використовувати лише під час розробки, тестування або технічного обслуговування. У робочому режимі обсяг службового виведення повинен бути обмежений.

Важливим аспектом є також захист конфігураційних параметрів вузла. До таких параметрів належать адреса вузла, параметри радіообміну, допустиме значення TTL, таблиця маршрутів і налаштування діагностики. Некоректна або несанкціонована зміна цих параметрів може порушити роботу мережі. Тому під час практичного використання системи необхідно передбачити контроль доступу до зміни конфігурації, а також перевірку коректності параметрів перед запуском вузла.

Сукупність реалізованих механізмів формує базовий рівень захисту програмної логіки LoRa mesh-мережі. Контрольна сума запобігає подальшій обробці пошкоджених пакетів, журнал ідентифікаторів зменшує ризик повторної обробки повідомлень, TTL обмежує кількість переходів, а перевірка адресних полів забезпечує правильне визначення призначення пакета. У результаті вузол не лише приймає та передає повідомлення, а й виконує попередню оцінку їхньої коректності та допустимості подальшої ретрансляції.

Отже, у межах розробленого програмного забезпечення захист інформаційно-комп'ютерної системи реалізовано на рівні контролю цілісності, фільтрації повторів, обмеження ретрансляції та перевірки службових параметрів пакета. Такі механізми є необхідними для стабільної роботи LoRa mesh-мережі, оскільки вони зменшують кількість некоректних передач,

запобігають повторному проходженню пакетів і дозволяють обмежити поширення помилкових повідомлень. Подальше вдосконалення захисту може бути пов'язане з розширенням механізмів автентифікації, шифрування та контролю доступу, що буде розглянуто в наступному підрозділі роботи.

### **3.4 Перспективи подальшого розвитку програмно забезпечення**

Подальший розвиток програмного забезпечення для маршрутизації даних у LoRa mesh-мережі може бути спрямований на підвищення стійкості, гнучкості, захищеності та масштабованості системи. Розроблена програмна логіка вже передбачає базові механізми обробки службових пакетів, перевірку контрольної суми, фільтрацію повторів, контроль TTL, аналіз адреси отримувача та прийняття рішення щодо локальної обробки або ретрансляції. Водночас практичне застосування mesh-мережі потребує подальшого вдосконалення окремих функціональних можливостей, оскільки умови роботи бездротових вузлів можуть змінюватися залежно від відстані, рельєфу, рівня завад, кількості вузлів і навантаження на радіоканал.

Одним із перспективних напрямів розвитку є удосконалення алгоритму маршрутизації. У базовому варіанті вибір наступного вузла може виконуватися на основі заздалегідь сформованої маршрутної таблиці. Такий підхід є достатнім для початкової реалізації, однак у подальшому програмне забезпечення доцільно доповнити механізмом динамічного оновлення маршрутів. У цьому випадку вузли зможуть періодично обмінюватися службовою інформацією про доступних сусідів, актуальність маршрутів і кількість переходів до кінцевого отримувача. Це дозволить системі адаптуватися до зміни топології мережі, наприклад у разі тимчасового зникнення окремого вузла або появи нового пристрою в зоні зв'язку.

Доцільним є також розширення маршрутної таблиці додатковими параметрами якості зв'язку. Крім адреси отримувача, наступного вузла та кількості переходів, у таблиці можна зберігати значення RSSI, SNR, час

останнього успішного зв'язку, кількість невдалих передач і статистику доставки пакетів. Використання таких показників дасть змогу обирати не лише найкоротший маршрут, а й більш стабільний шлях передавання даних. Наприклад, маршрут із меншою кількістю переходів не завжди буде найкращим, якщо один із вузлів має нестабільний радіоканал. Тому в подальшому доцільно реалізувати комбіновану метрику маршруту, яка враховуватиме як кількість переходів, так і якість зв'язку між вузлами.

Окремим напрямом удосконалення може бути використання двох LoRa-радіомодулів на одному вузлі. Такий підхід дозволить розділити функції приймання та передавання або організувати роботу на різних частотних каналах. Оскільки LoRa-модуль у звичайному режимі не може одночасно приймати й передавати дані, використання двох радіомодулів може підвищити ефективність ретрансляційного вузла. Один модуль може працювати переважно на приймання пакетів від сусідніх вузлів, а другий на передавання або ретрансляцію повідомлень далі за маршрутом. Це особливо доцільно для вузлів, які виконують роль проміжних ретрансляторів і обробляють більшу кількість пакетів порівняно з кінцевими пристроями.

Перспективним є впровадження автоматичного вибору частотного каналу. У реальних умовах експлуатації окремі канали можуть бути перевантажені або зашумлені, що впливає на якість передавання. Програмне забезпечення можна доповнити механізмом періодичної оцінки рівня сигналу та завад, після чого вузол зможе обирати канал із кращими параметрами. Такий підхід дозволить підвищити стійкість мережі до зовнішніх завад і зменшити кількість невдалих передач. Для уникнення хаотичної зміни частот доцільно передбачити службові повідомлення, за допомогою яких вузли узгоджуватимуть перехід на інший канал.

Важливим напрямом розвитку є реалізація механізму підтвердження доставки пакетів. У базовій логіці вузол може передати пакет наступному пристрою, однак для підвищення надійності доцільно отримувати підтвердження від кінцевого вузла або від наступного вузла маршруту. Такий

механізм може бути реалізований за допомогою коротких службових АСК-повідомлень, які містять ідентифікатор прийнятого пакета та статус його обробки. Якщо підтвердження не отримано протягом визначеного часу, вузол може виконати повторну передачу або обрати інший маршрут. При цьому кількість повторних передач має бути обмеженою, щоб не створювати надмірного навантаження на LoRa-канал.

Подальший розвиток програмного забезпечення також може бути пов'язаний із підвищенням рівня інформаційної безпеки. У попередньому підрозділі було розглянуто базові механізми захисту, зокрема контроль CRC, журнал пакетів, TTL і перевірку адресних полів. У майбутньому ці механізми можна доповнити шифруванням корисного навантаження, автентифікацією службових пакетів і використанням ключів для окремих груп вузлів. Це дозволить зменшити ризик несанкціонованого читання повідомлень, підміни службових полів або надсилання фальшивих пакетів у мережу. Окрему увагу доцільно приділити захисту від replay-атак, коли зломисник повторно передає раніше перехоплений пакет.

Ще одним важливим напрямом є розширення засобів діагностики та моніторингу. У подальших версіях програмного забезпечення можна реалізувати журнал подій, у якому фіксуватимуться прийняті та передані пакети, помилки CRC, повторні повідомлення, зміни маршрутної таблиці, значення TTL, RSSI, SNR і рішення про ретрансляцію. Такі дані можуть використовуватися для аналізу стабільності мережі, пошуку проблемних вузлів і оцінювання якості окремих маршрутів. Для зручності налагодження доцільно передбачити можливість виведення скороченого та розширеного діагностичного режиму.

Перспективним є також створення механізму віддаленого оновлення конфігурації вузлів. До таких параметрів можуть належати адреса вузла, максимальне значення TTL, список дозволених сусідів, частотний канал, швидкість передавання, режим діагностики та параметри повторної передачі. У разі збільшення кількості вузлів ручне налаштування кожного пристрою стає

менш зручним, тому централізоване або напівцентралізоване оновлення параметрів може значно спростити обслуговування мережі. При цьому зміна конфігурації повинна бути захищеною, щоб уникнути несанкціонованого втручання в роботу системи.

Окремої уваги потребує оптимізація енергоспоживання. Вузли LoRa mesh-мережі можуть працювати від автономних джерел живлення, тому програмне забезпечення має враховувати режими зниженого споживання. У подальшому можна реалізувати перехід мікроконтролера в режим сну, періодичне пробудження для приймання службових повідомлень, обмеження частоти діагностичного виведення та адаптивне керування активністю радіомодуля. Для ретрансляційних вузлів, які повинні постійно бути доступними для приймання пакетів, енергозбереження має бути збалансоване з вимогами до стабільності зв'язку.

Таким чином, перспективи подальшого розвитку програмного забезпечення пов'язані з удосконаленням маршрутизації, використанням показників якості радіозв'язку, підтримкою двох LoRa-модулів, автоматичним вибором частотного каналу, підтвердженням доставки пакетів, посиленням інформаційної безпеки, розширенням діагностики, віддаленим налаштуванням вузлів і зменшенням енергоспоживання. Реалізація цих напрямів дозволить підвищити надійність LoRa mesh-мережі, покращити її адаптивність до зміни умов зв'язку та розширити можливості практичного застосування розробленої системи.

## ВИСНОВКИ

У кваліфікаційній роботі бакалавра було розроблено програмне забезпечення для організації маршрутизації даних у mesh-мережі з використанням технології LoRa. У процесі виконання роботи розглянуто особливості побудови автономних бездротових мереж, визначено роль проміжних вузлів у передаванні пакетів і обґрунтовано необхідність створення програмної логіки, яка забезпечує приймання, перевірку, маршрутизацію та ретрансляцію повідомлень. Розроблене рішення орієнтоване на використання мікроконтролера ESP32 та радіомодуля E32-400M30S, що дозволяє сформувати основу для побудови вузла LoRa mesh-мережі.

У результаті виконання роботи було сформовано структуру службового пакета, визначено основні програмні модулі, реалізовано механізми перевірки контрольної суми, фільтрації повторів, контролю TTL, пошуку наступного вузла та діагностичного виведення. Проведене тестування підтвердило коректність базової логіки обробки пакетів: проходження повідомлення через проміжні вузли, приймання пакета кінцевим вузлом і відкидання пошкодженого повідомлення. Отримані результати свідчать про досягнення поставленої мети та створення програмної основи для подальшого розвитку LoRa mesh-системи.

Проаналізовано сучасний стан технологій LoRa, принципи побудови mesh-мереж і особливості маршрутизації даних у бездротових низькошвидкісних мережах. Встановлено, що LoRa є доцільною технологією для передавання невеликих обсягів даних на значні відстані за умови обмеженого енергоспоживання. Водночас визначено, що сама технологія LoRa не забезпечує повноцінної логіки mesh-маршрутизації, тому відповідні механізми мають бути реалізовані на рівні програмного забезпечення. Аналіз показав, що для таких систем важливими є адресація вузлів, контроль повторів, обмеження кількості переходів і перевірка коректності службових пакетів.

Досліджено можливості використання мікроконтролера ESP32 та радіомодуля E32-400M30S як апаратної основи вузла LoRa mesh-мережі. Визначено, що ESP32 має достатні обчислювальні ресурси та набір периферійних інтерфейсів для реалізації логіки приймання, обробки й маршрутизації пакетів. Радіомодуль E32-400M30S є придатним для організації LoRa-зв'язку в діапазоні 433 МГц і може використовуватися як фізичний рівень передавання даних між вузлами. Поєднання цих компонентів дозволяє створити автономний вузол, здатний виконувати функції відправника, отримувача або проміжного ретранслятора.

Визначено функціональні та нефункціональні вимоги до програмного забезпечення вузла mesh-мережі. До функціональних вимог віднесено ініціалізацію вузла, приймання пакетів, перевірку службових полів, фільтрацію повторів, контроль TTL, ведення маршрутної таблиці та ретрансляцію повідомлень. Нефункціональні вимоги охоплюють надійність, економне використання радіоканалу, раціональне використання пам'яті, масштабованість і зручність налагодження. Сформульовані вимоги стали основою для подальшого проєктування програмної структури вузла.

Спроектовано структуру програмного забезпечення з поділом на окремі модулі приймання, перевірки пакета, маршрутизації, ретрансляції, роботи з маршрутною таблицею та діагностики. Такий поділ дозволив розмежувати функції окремих частин програми та спростити подальшу реалізацію алгоритму. Для формалізації проєктних рішень було використано UML-діаграми, які відображають сценарії взаємодії, компонентну структуру та послідовність обробки пакета. Запропонована архітектура забезпечує логічну узгодженість між прийманням повідомлення, його перевіркою, маршрутизацією та передаванням наступному вузлу.

Сформовано структуру службового пакета для передавання даних між вузлами мережі. До складу пакета включено ідентифікатор повідомлення, адресу відправника, адресу отримувача, адресу попереднього вузла, лічильник TTL, контрольну суму та корисне навантаження. Така структура дозволяє

визначати напрям проходження пакета, перевіряти його цілісність, виявляти повторні повідомлення та обмежувати кількість ретрансляцій. Використання єдиного формату службового пакета спрощує взаємодію між модулями програмного забезпечення.

Реалізовано основні програмні механізми обробки пакетів. До них належать перевірка контрольної суми, фільтрація повторів, пошук наступного вузла в маршрутній таблиці, контроль TTL і ретрансляція повідомлення. Реалізовані фрагменти програмного коду показують послідовність прийняття рішення вузлом після приймання пакета. У результаті було створено базову програмну логіку, яка дозволяє вузлу визначати, чи потрібно обробити повідомлення локально, відкинути його або передати далі за маршрутом.

Проведено тестування та налагодження розробленої програмної логіки. Перевірено проходження пакета в кільцевій топології, приймання повідомлення кінцевим вузлом і відкидання пошкодженого пакета з помилкою CRC. Результати тестування засвідчили, що програмне забезпечення коректно обробляє службові поля пакета, контролює значення TTL і не допускає подальшого поширення некоректних повідомлень. Налагоджувальне виведення через Serial Monitor дало змогу простежити послідовність роботи алгоритму та підтвердити правильність прийнятих програмою рішень.

Розглянуто механізми захисту інформаційно-комп'ютерної системи. Основну увагу приділено базовим програмним засобам захисту, зокрема контролю цілісності пакета, фільтрації повторних повідомлень, перевірці адресних полів і обмеженню ретрансляції за допомогою TTL. Встановлено, що такі механізми зменшують ризик поширення пошкоджених або повторних пакетів у mesh-мережі. Розглянуті підходи формують початковий рівень захисту програмної логіки та можуть бути доповнені криптографічними механізмами в подальших версіях системи.

Визначено перспективи подальшого розвитку програмного забезпечення. До основних напрямів удосконалення віднесено динамічне оновлення маршрутної таблиці, використання показників якості радіозв'язку, підтримку

двох LoRa-модулів, автоматичний вибір частотного каналу та реалізацію підтвердження доставки пакетів. Також перспективними є посилення інформаційної безпеки, розширення діагностичних можливостей, віддалене налаштування вузлів і оптимізація енергоспоживання. Реалізація цих напрямів дозволить підвищити надійність, масштабованість і практичну придатність LoRa mesh-мережі.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Semtech. SX1278 LoRa Connect Transceiver. URL: <https://www.semtech.com/products/wireless-rf/lora-connect/sx1278> (дата звернення: 14.02.2026).
2. Espressif Systems. ESP32 Series Datasheet. URL: [https://www.espressif.com/sites/default/files/documentation/esp32\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf) (дата звернення: 14.02.2026).
3. EBYTE. E32-400M30S Based on the SX1278 produced by Semtech 433/470MHz LoRa SMD wireless module with the maximum transmission power of 1W. URL: <https://www.cdebyte.com/products/E32-400M30S> (дата звернення: 14.02.2026).
4. ThinkNode M2 Meshtastic (LoRa). URL: [https://arduino.ua/prod8037-thinknode-m2-meshtastic-lora-transiver-na-esp32-s3-z-1-3-oled-displem-y-korpysi?srsltidAfmBOOpW\\_qU4Dvo5Je1YP5p9aOAzcmuF2i0HOYtR9PeO\\_HHCd8Axc4C](https://arduino.ua/prod8037-thinknode-m2-meshtastic-lora-transiver-na-esp32-s3-z-1-3-oled-displem-y-korpysi?srsltidAfmBOOpW_qU4Dvo5Je1YP5p9aOAzcmuF2i0HOYtR9PeO_HHCd8Axc4C) (дата звернення: 14.02.2026).
5. LoRa Alliance. What is LoRaWAN® Specification / LoRaWAN® Network Architecture. URL: <https://loro-alliance.org/about-lorawan-old/> (дата звернення: 14.02.2026).
6. Wong A. W. L. et al. Multi-Hop and Mesh for LoRa Networks: Recent Advancements, Issues, and Recommended Applications. ACM Computing Surveys. URL: <https://dl.acm.org/doi/full/10.1145/3638241> (дата звернення: 03.03.2026).
7. Організація комп'ютерних мереж. URL: <https://kremenetskyu.blogspot.com/2017/10/blog-post.html> (дата звернення: 03.03.2026).
8. LoRaWAN Mesh: The Revolutionary Solution to Connect Your IoT Objects, Without the Internet. URL: [https://sparwan.com/en/blogs/news/lorawan-mesh-la-solution-revolutionnaire-pour-connecter-vos-objets-iot-sans-internet?srsltid=AfmBOOp-P66zomk\\_t3EM0wZ08jnh5qBJyXGdw-0oTq\\_YWzI8nG\\_Pt0ti](https://sparwan.com/en/blogs/news/lorawan-mesh-la-solution-revolutionnaire-pour-connecter-vos-objets-iot-sans-internet?srsltid=AfmBOOp-P66zomk_t3EM0wZ08jnh5qBJyXGdw-0oTq_YWzI8nG_Pt0ti) (дата звернення: 03.03.2026).

9. Arregui Almeida D. et al. Gateway-Free LoRa Mesh on ESP32: Design, Self-Healing Mechanisms, and Empirical Performance. Sensors. URL: <https://www.mdpi.com/1424-8220/25/19/6036> (дата звернення: 03.03.2026).

10. Espressif Systems. ESP-IDF Programming Guide. URL: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html> (дата звернення: 14.03.2026).

11. RadioLib. Universal wireless communication library for embedded devices. URL: <https://jgromes.github.io/RadioLib/> (дата звернення: 14.03.2026).

12. Ironhack. Functional vs Non-Functional Requirements: Understanding the Core Differences and Examples. URL: <https://www.ironhack.com/gb/blog/functional-vs-non-functional-requirements-understanding-the-core-differences-and> (дата звернення: 14.03.2026).

13. UML overview – where and why each UML diagram is used. URL: <https://www.drawio.com/blog/uml-overview> (дата звернення: 14.03.2026).

14. Chengdu Ebyte Electronic Technology Co., Ltd. E32-400M30S User Manual. URL: <https://www.cdebyte.com/pdf-down.aspx?id=1924> (дата звернення: 20.04.2026).