МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри,
д-р техн. наук, проф.

_____ Ірина ЖУРАВСЬКА

« __ » _____ 202__ р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

СИСТЕМА ДАЛЬНОГО РАДІОЗВ'ЯЗКУ ДЛЯ КОНТРОЛЮ ДАТЧИКІВ НА ОСНОВІ ОДНОПЛАТНИХ МІКРОКОМП'ЮТЕРІВ З LORA

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія»

Здобувач _____Денис ВАРАНКІН
підпис

«__» _____ 202__ р.

Керівник д-р техн. наук, проф. _____Ірина ЖУРАВСЬКА
підпис

«__» _____ 202__ р.

Миколаїв – 2024

Завдання на виконання кваліфікаційної магістерської роботи

Факультет	Комп’ютерних наук
Кафедра	Комп’ютерної інженерії
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	123 Комп’ютерна інженерія
Освітня програма	Комп’ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри комп’ютерної інженерії
_____ Ірина ЖУРАВСЬКА
«___»_____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

_____Варанкіна Дениса Володимировича_____
(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи _
Система дального радіозв’язку на основі одноплатних мікрокомп’ютерів з LoRa_
_____

Затверджена наказом ректора ЧНУ ім. Петра Могили від «16» вересня 2024 р. № 236.

2. Строк представлення кваліфікаційної роботи «____» _____ 202__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні Створення енергоефективної системи передачі даних на базі технології LoRa, яка забезпечуватиме стабільний зв’язок на великих відстанях з мінімальною затримкою та втратами пакетів. Система повинна бути легко масштабованою, надійною та придатною для використання в різних умовах, зокрема у сільській місцевості та для IoT-рішень.

4. Перелік питань, що підлягають розробці _____
1) огляд і аналіз інформаційних технологій;
2) порівняння малопотужних бездротових технології;
3) обрання протоколів маршрутизації;
4) розробка системи отримання повідомлення даних;
5) розробка моделі;
6) проведення та оформлення тестуваннь.

5. Перелік графічних матеріалів _____
1) блок схеми роботи алгоритмів маршрутизації
2) слайди презентації
3) результати тестів
4) місце проведення тестів

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ **Ірина ЖУРАВСЬКА**
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ **Денис ВАРАНКІН**
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання «____» _____ 202__ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної магістерської роботи

Тема: Система дального радіозв'язку на основі одноплатних мікрокомп'ютерів з LoRa

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	01.09.2024	08.09.2024	Виконано
2.	Огляд літератури за темою роботи	08.09.2024	16.09.2024	Виконано
3.	Складання календарного плану КМР	17.09.2024	20.09.2024	Виконано
4.	Аналіз предметної області	21.09.2024	25.09.2024	Виконано
5.	Розробка проєктних рішень	26.09.2024	07.10.2024	Виконано
6.	Моделювання апаратної частини	08.10.2024	20.10.2024	Виконано
7.	Оформлення КМР та презентації	21.10.2024	29.10.2024	Виконано
8.	Перший передзахист КМР	30.10.2024	31.10.2024	Виконано
9.	Розробка програмного забезпечення та тестування	01.11.2024	22.11.2024	Виконано
10.	Відгук керівника КМР	23.11.2024	26.11.2024	Виконано
11.	Другий передзахист КМР	28.11.2024	29.11.2024	Виконано
12.	Рецензування	30.11.2024	05.12.2024	Виконано
13.	Завершення оформлення КМР та презентації	06.12.2024	12.12.2024	Виконано
14.	Захист КМР	19.12.2024	19.12.2024	

Керівник роботи _____ **Ірина ЖУРАВСЬКА**
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ **Денис ВАРАНКІН**
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

АНОТАЦІЯ

до кваліфікаційної магістерської роботи
«Система дального радіозв'язку для контролю датчиків
на основі одноплатних мікрокомп'ютерів з LoRa»
Здобувач гр. 605: Варанкін Денис Володимирович
Керівник: д-р техн. наук, проф. Журавська І. М.

Актуальність теми кваліфікаційної роботи обумовлена стрімким розширенням зони покриття мережі Інтернет та зростанням технічних можливостей для впровадження сучасних технологій, зокрема LPWAN (Low Power Wide Area Network). Ці технології відкривають нові можливості для дистанційного управління пристроями, що робить актуальним вивчення специфіки LPWAN-мереж. LPWAN створює оригінальну парадигму комунікацій, що доповнює стільникові мережі та бездротові технології малого радіусу дії, що має важливе значення для розвитку Інтернету речей (IoT).

Об’єктом дослідження є процес моніторингу та передачі даних, що використовує технологію LoRa для контролю датчиків на основі одноплатних мікрокомп'ютерів. Особливу увагу приділено організації збору, передачі та обробки даних, яка потребує оптимізації для забезпечення таких характеристик системи, як надійність, енергоефективність та відмовостійкість.

Предметом дослідження є система дальнього радіозв'язку для контролю датчиків на основі технології LoRa, що реалізується з використанням одноплатних мікрокомп'ютерів.

Мета кваліфікаційної роботи: розробка системи передачі даних з використанням технології LoRa для забезпечення стабільного зв’язку на великих відстанях із мінімальним енергоспоживанням.

Кваліфікаційна магістерська робота містить: перелік скорочень, вступ, чотири розділи, висновки, перелік джерел посилання та два додатки.

У вступі обґрунтовано актуальність теми, сформульовано об’єкт і предмет дослідження, визначено мету та завдання. У першому розділі проведено аналіз існуючих бездротових технологій та обрано оптимальну для розробки системи. Другий розділ містить опис алгоритмів передачі даних та методів маршрутизації в мережах LoRa. У третьому розділі описано процес розробки та проєктування системи, включаючи вибір компонентів, архітектуру мережі та інтеграцію програмного забезпечення. Четвертий розділ присвячено налаштуванню середовища для тестування, проведенню експериментів та аналізу отриманих результатів. У висновку підведено підсумки виконаної роботи та надано рекомендації щодо подальшого розвитку системи. Додатки містять програмний код і технічну документацію.

Кваліфікаційна магістерська робота містить 68 сторінок (без додатків), 42 рис., 5 табл., 24 джерела посилання, 2 додатки.

Ключові слова: LoRa, LPWAN, бездротовий зв’язок, Інтернет речей, одноплатні мікрокомп’ютери, дальній радіозв’язок, маршрутизація, енергоефективність, сенсорна мережа, топологія «зірка», обробка даних, передача даних

ASTRACT

of the Master's Thesis
"Long-range radio communication system for sensor control
based on single-board microcomputers with LoRa"
Applicant: Varankin Denys
Supervisor: Dr. Sci. (Techn.), Professor Zhuravska I. M.

The relevance of the topic of the Master's Thesis is determined by the rapid expansion of Internet coverage and the growing technical capabilities for implementing modern technologies, particularly LPWAN (Low Power Wide Area Network). These technologies open up new opportunities for remote device management, making the study of LPWAN networks highly relevant. LPWAN introduces a unique communication paradigm that complements cellular networks and short-range wireless technologies, which is crucial for the development of the Internet of Things (IoT).

The subject of the research is a long-range radio communication system for sensor control based on LoRa technology, implemented using single-board microcomputers.

The object of the research is the process of monitoring and data transmission using LoRa technology for sensor control with single-board microcomputers. Special attention is given to the organization of data collection, transmission, and processing, which requires optimization to ensure system characteristics such as reliability, energy efficiency, and fault tolerance.

The purpose of the Master's Thesis is to develop a data transmission system using LoRa technology to ensure stable long-range communication with minimal energy consumption.

The Master's Thesis includes: a list of abbreviations, an introduction, four chapters, a conclusion, a list of references, and four appendices.

The introduction substantiates the relevance of the topic, formulates the object and subject of the research, and defines the purpose and objectives. The first chapter analyzes existing wireless technologies and selects the optimal one for system development. The second chapter describes data transmission algorithms and routing methods in LoRa networks. The third chapter outlines the process of system development and design, including the selection of components, network architecture, and software integration. The fourth chapter focuses on setting up the testing environment, conducting experiments, and analyzing the obtained results.

The conclusion summarizes the work performed and provides recommendations for further system development. The appendices contain the software code and technical documentation.

The qualification work consists of 68 pages (excluding appendices), 39 figures, 5 tables, 25 references, and 2 appendices.

Keywords: LoRa, LPWAN, wireless communication, Internet of Things (IoT), single-board microcomputers, long-range radio communication, routing, energy efficiency, sensor network, star topology, data processing, data transmission.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КСВ – коефіцієнт стоячої хвилі
НВЧ – надвисокі частоти
ОЗП – оперативний запам’ятовуючий пристрій
ЦАП – цифро-аналоговий перетворювач

ADC – Analog-to-Digital Converter
BLE – Bluetooth Low Energy
dBi – Decibel relative to isotropic radiator
EEPROM – Electrically Erasable Programmable Read-Only Memory
GPS – Global Positioning System
IoT – Internet of Things
LoRa – Long Range
LPWAN – Low Power Wide Area Network
MHz – Megahertz / Мегагерц
PWM – Pulse Width Modulation
RREP – Route Reply,
SPI – Serial Peripheral Interface
SRAM – Static Random Access Memory
UART – Universal Asynchronous Receiver-Transmitter

ВСТУП

У сучасному світі, з розвитком технологій Інтернету Речей (IoT), зростає потреба в ефективних та надійних системах радіозв'язку для моніторингу та контролю різноманітних датчиків. Системи дальнього радіозв'язку, зокрема на основі технології LoRa (Long Range), забезпечують широкий діапазон покриття, низьке енергоспоживання та високу надійність передачі даних, що робить їх ідеальними для застосувань у віддалених та важкодоступних місцях.

Завдяки архітектурі типу «зірка», у якій кінцеві пристрої безпосередньо спілкуються зі шлюзами, технології LoRa можуть зменшити енергетичні витрати на роботу датчиків, що, в свою чергу, продовжує термін їхньої служби. Це особливо важливо для систем, які потребують тривалого автономного функціонування без частого обслуговування.

Впровадження одноплатних мікрокомп'ютерів у дані системи дозволяє створити універсальні рішення для контролю та управління датчиками, забезпечуючи інтеграцію різноманітних функцій, таких як геолокація, обробка даних та управління пристроями.

Використання мікрокомп'ютерів, поряд з технологією LoRa, дозволяє створити гнучкі та масштабовані системи, які можуть задовольняти вимоги сучасних користувачів.

Актуальність теми даної роботи зумовлена тим, що з кожним роком зона покриття мережі Інтернет збільшується і разом з цим з'являються технічні можливості, що дозволяють за допомогою технології LPWAN керувати різними пристроями. У зв'язку з цим постає актуальним питання аналізу особливостей так званих LPWAN-мереж, що задають оригінальну парадигму комунікацій, яка доповнить стільниковий зв'язок і технології бездротового зв'язку малого радіусу дії для різних застосунків Інтернету речей.

Предметом дослідження є система дальнього радіозв'язку для контролю датчиків на основі технології LoRa, що реалізується з використанням одноплатних мікрокомп'ютерів.

Об’єктом дослідження є процес моніторингу та передачі даних, що використовує технологію LoRa для контролю датчиків на основі одноплатних мікрокомп'ютерів. В рамках цього дослідження об’єктом виступає процес організації віддаленого збору, передачі та обробки даних, який потребує оптимізації для забезпечення надійності, енергоефективності та відмовостійкості системи, що використовує бездротові сенсори та LoRa-технології.

Це дослідження охоплює різні аспекти системи, зокрема її архітектуру, яка грунтується на топології «зірка», що забезпечує ефективну та надійну передачу даних від кінцевих пристроїв до шлюзів. Аналіз технічних характеристик, таких як діапазон частот, швидкість передачі даних, енергоспоживання та автономність пристроїв, є ключовим елементом дослідження, оскільки ці фактори критично важливі для практичних застосувань у віддалених умовах.

Метою кваліфікаційної магістерської роботи є розробка та впровадження ефективної системи далекого радіозв'язку для контролю датчиків на основі одноплатних мікрокомп'ютерів з використанням технології LoRa. Така система дозволить забезпечити надійний моніторинг та передачу даних на великі відстані, мінімізуючи енергоспоживання і оптимізуючи витрати. Це важливий крок для вдосконалення систем контролю та автоматизації у різних галузях.

Для досягнення поставленої мети необхідно виконати наступні **завдання**:

1) провести аналіз існуючих технологій та протоколів бездротового зв'язку, що використовуються для контролю датчиків, з акцентом на LPWAN і LoRa;

2) дослідити можливості одноплатних мікрокомп'ютерів у контексті побудови мережі для моніторингу датчиків;

3) розробити апаратну та програмну архітектуру системи контролю датчиків з використанням LoRa;

4) реалізувати прототип системи та провести його тестування для оцінки стабільності передачі даних та енергоспоживання.

Практичне значення отриманих результатів полягає в розробці ефективної системи радіозв'язку для контролю датчиків на основі одноплатних мікрокомп'ютерів з використанням технології LoRa. Ця система може бути застосована для побудови масштабованих та енергоефективних мереж моніторингу в різних сферах, таких як сільське господарство, логістика, інфраструктура розумного міста, промисловий контроль та екологічний моніторинг.

Розроблене рішення дозволяє забезпечити тривалий час автономної роботи датчиків, що є важливим для зменшення експлуатаційних витрат та підвищення надійності системи в реальних умовах. Впровадження таких систем може значно полегшити управління великими масивами розподілених датчиків, забезпечуючи безперервний моніторинг у важкодоступних або віддалених місцях.

Робота пройшла **апробацію** під час XXI Міжнародної наукової конференції «Ольвійський форум» (Миколаїв, 20–23 червня 2024 р.).

Публікації. Основні положення кваліфікаційної магістерської роботи опубліковані у збірнику матеріалів XXI Міжнародної наукової конференції «Ольвійський форум – 2024» [14].

1 АНАЛІТИЧНИЙ ОГЛЯД СИСТЕМИ LORAWAN

1.1 Огляд і аналіз інформаційних технологій

LoRaWAN (Long-Range Wide Area Network) – це тип технології бездротового зв'язку для передачі даних на великі відстані. Вона створена для забезпечення низькошвидкісного збирання даних між підключеними сенсорами, що відстежують і передають інформацію [1]. LoRa здатна передавати дані на великі відстані, використовуючи малу потужність. Це працює завдяки фізичним законам, які стверджують, що для передачі даних на далекі відстані потрібно або збільшити потужність, або зменшити пропускну здатність.

Існує кілька інших типів подібних мереж, таких як LPWAN (мережа з низьким енергоспоживанням для широкої зони покриття), LPWA (мережа з низьким енергоспоживанням для широкої зони) або LPN (мережа з низьким енергоспоживанням).

LoRa працює на радіочастотах у діапазонах мегагерц, що не вимагають ліцензії, таких як 169 МГц, 433 МГц, 868 МГц (Європа) і 915 МГц (Північна Америка). LoRa забезпечує передачу на дуже великі відстані (більше 10 км у сільських районах) при низькому енергоспоживанні. Технологія складається з двох частин: LoRa – фізичний рівень, і протокол зв'язку, побудований на базі цього фізичного рівня LoRa. Комунікаційний рівень представлений WAN (Wide Area Network) – відкритим протоколом зв'язку, який визначається консорціумом LoRa Alliance.

Отже, LoRaWAN визначає протокол зв'язку та архітектуру системи для мережі, тоді як фізичний рівень LoRa забезпечує довготривалий зв'язок. Протокол зв'язку LoRaWAN гарантує надійну та безпечну передачу даних [2].

1.2 Фізичний і комунікаційний рівні мережі LoRaWAN

Мережі LoRaWAN можуть бути побудовані за різними топологіями. Найбільшого розповсюдження мають сіткова (англ. mesh) та зірка. Згідно з [2],

LoRaWAN має три класи кінцевих пристроїв, які відповідають різним вимогам (рис. 1.1):

– клас A: передача даних ініціюється кінцевим пристроєм й завжди асинхронна. Передача даних на пристрій завжди відбувається після відправки даних з пристрою. Є найбільш енергоефективним;

– клас B: відповідно з класом A, періодично виконуються запити з ціллю отримання даних на пристрій;

– клас C: завжди прослуховує лінію зв’язку. Підходить для пристроїв які мають постійне джерело енергії, оскільки має високий ступінь енергоживлення.

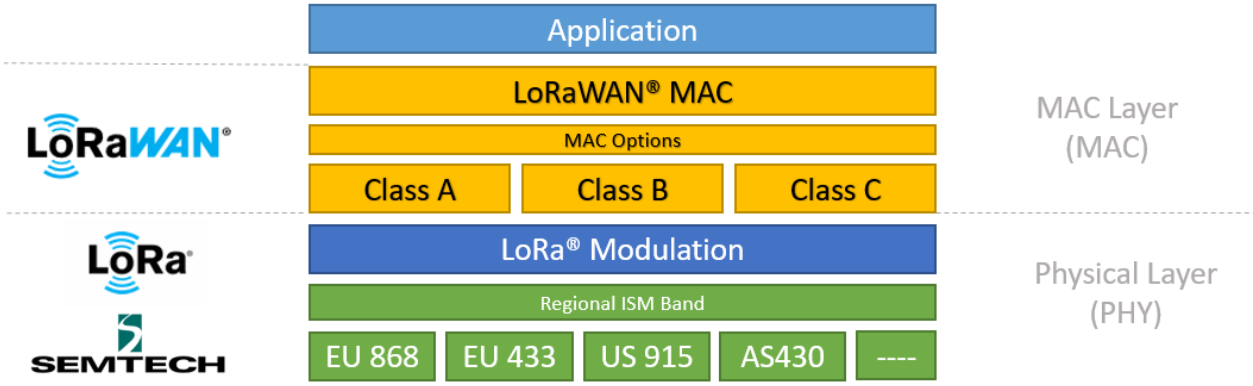


Рисунок 1.1 – Принципи роботи LoRaWAN за рівнями OSI

Питання безпеки підключення та обміну даними в LoRaWAN мережах вирішується використанням ключами шифрування за алгоритмом AES із довжиною в 128 біт. У мережі визначаються два типи ключів [4]:

– унікальний ключ сесії поміж кінцевим пристроєм та шлюзом (NwkSKey);

– унікальний ключ рівня застосунку, який може використовуватись усіма пристроями (AppSKey).

Мережі LoRaWAN підтримують різні топології, серед яких найбільш поширеними є сіткова та зіркоподібна. Кінцеві пристрої класифікуються на три класи (A, B, C), які відповідають різним вимогам енергоефективності та способу передачі даних. Найбільш енергоефективним є клас A, тоді як клас C призначений для пристроїв з постійним джерелом живлення [5].

1.3 Топологія мереж LoRa та LoRaWAN

Архітектура мережі LoRaWAN реалізується за топологією «star-of-stars». Це дозволяє базовим станціям передавати дані між сенсорними вузлами та сервером мережі. Зв'язок між сенсорними вузлами та базовими станціями відбувається через бездротовий канал, використовуючи фізичний рівень LoRa, а з'єднання між шлюзами і центральним сервером здійснюється через магістральну мережу на базі IP-протоколу [3]:

– кінцеві вузли передають дані безпосередньо всім шлюзам у межах досяжності, використовуючи технологію LoRa;

– LoRa-транспондер використовується для передачі сигналів за запатентованим методом радіопередачі LoRa, а також може мати мікроконтролер (з вбудованою пам'яттю) як додаткову опцію;

– кінцеві вузли LoRaWAN зазвичай є сенсорами, що живляться від батарей (класу A та класу B). Термін служби батареї у вузлі зазвичай становить від двох до п'яти років [1]. Сенсори LoRa використовують прямий радіозв'язок, а відстань передачі може варіюватися від однієї до десяти кілометрів (табл. 1.1).

Таблиця 1.1 – Типи вузлів

Параметри	**Клас A**	**Клас B**	**Клас C**
Живлення	Батарейка	Батарейка	Батарейка
Напрямок зв'язку	Двосторонній (1 UL + 2 DL слот)	Двосторонній зі запланованими слотами вниз	Двосторонній з переважно активним прослуховуванням
Типи повідомлень	Унікаст	Унікаст та Мультикаст	Унікаст та Мультикаст
Розмір навантаження	Малий розмір	Малий розмір	Малий розмір

Параметри	Клас A	Клас B	Клас C
Ініціація зв'язку від пристрою	Пристрій ініціює зв'язок (Uplink)	Додаткове вікно прийому	Сервер може ініціювати передачу у будь-який час
Комунікація сервера з пристроєм	Сервер спілкується з пристроєм під час запланованих вікон	Сервер може ініціювати передачу через фіксовані інтервали	Пристрій постійно отримує дані

Вузли можуть належати до трьох класів (A, B, C), залежно від потреб у енергоспоживанні та режиму передачі даних [4]. Клас A є найбільш енергоефективним, тоді як клас C забезпечує постійне прослуховування мережі. Термін служби батарей у вузлах зазвичай становить від 2 до 5 років, а дальність передачі варіюється від 1 до 10 кілометрів [2].

1.4 LoRa-шлюзи

LoRa-сенсори передають дані до LoRa-шлюзів. Ці шлюзи підключаються до інтернету через стандартний IP-протокол і передають отримані дані від вбудованих LoRa-сенсорів до Інтернету, тобто до мережі, сервера або хмари [6].

Пристрої-шлюзи завжди підключені до джерела живлення. З'єднання шлюзів працює як прозорий міст, просто перетворюючи RF-пакети на IP-пакети та навпаки.

До переваг LoRaWAN можна віднести наступне:

– низькопотужні сенсори, здатні покривати великі території, вимірювані в кілометрах;

– функціонує в промислових, наукових та медичних (англ. Industrial, Scientific and Medical, ISM) радіочастотних діапазонах. Це безкоштовні (неліцензовані) частоти, без початкових витрат на ліцензування технології;

– низьке енергоспоживання забезпечує довгий термін служби батарей пристроїв. Батареї сенсорів можуть працювати від двох до п’яти років (клас A і клас B) [7];

– один пристрій LoRa-шлюзу розроблений для обслуговування тисяч кінцевих пристроїв або вузлів;

– ідеально підходить для моніторингу активів, розміщених у полях;

– широко використовується для додатків M2M/IoT;

– далекобійна функціональність дозволяє реалізовувати розумні рішення, такі як додатки для розумних міст;

– бездротова технологія, легка у налаштуванні та швидка в розгортанні;

– безпека: передбачено два рівні захисту: один для мережі та один для застосування з AES-шифруванням;

– повністю двостороння комунікація.

Серед недоліків LoRaWAN можна виділити наступні [4; 5]:

– не призначено для великого обсягу даних, максимальний обсяг обмежено до 0,3–50,0 кбіт/с;

– не підтримує передачу аудіо чи відео;

– обмежено комунікацією прямої видимості (англ. Line-of-sight, LOS);

– не підходить для безперервного моніторингу (за винятком пристроїв класу C);

– не є ідеальним варіантом для додатків реального часу, які вимагають низької затримки та обмеженого джіттера.

LoRa працює на відкритих частотах, які не вимагають державної ліцензії. Це означає, що існує шанс зазнати перешкод на цій частоті, що може знизити швидкість передачі даних. Крім того, оскільки відкриті частоти відрізняються в різних країнах, продуктивність може варіюватися [8].

1.5 Різниця між 4G-сімкою та підключенням по LPWAN-мережі

Витрати на енергію для датчиків, які використовують 4G-картки, будуть значними, що вимагатиме їхньої підзарядки не рідше ніж один раз на місяць. При цьому, порівняння вартості послуг різних 4G-провайдерів для таких датчиків становитиме від 75 грн на місяць (провайдер № 1) до 400 грн (провайдер № 2). У той же час, використання мережі SigFox (LPWAN-провайдера) обійдеться від 7 до 50 грн на місяць, навіть за умови міжнародного використання датчика.

Самостійно можна побудувати лише мережу формату LoRaWAN, оскільки це єдиний формат мовлення в діапазонах, що не ліцензуються. Одна зона покриття складатиметься з LoRaWAN – шлюзу та маршрутизатора [2].

Таблиця 1.2 – Порівняння LPWAN-мереж представлених в Україні

Характеристика	**LoRa**	**Sigfox**	**NB-IoT**
Метод модуляції	Чірповий розподілений спектр (англ. Chirp Spread Spectrum, CSS)	ISM	OFDMA/DSSS (Ортогональне частотне мультиплексування з розподілом спектра)
Діапазон частот	ISM	ISM	Ліцензований
Швидкість передачі даних	0,3–50,0 кбіт/с	100,0 біт/с	Завантаження: 1,0–144,0 кбіт/с; Передача: 1,0–200,0 кбіт/с
Ширина смуги	До 500 кГц	100 кГц	200 кГц
Час автономної роботи (1 датчик, мінімальне використання)	Більше 10 років	До 10 років	До 10 років

Характеристика	LoRa	Sigfox	NB-IoT
Час автономної роботи (середнє навантаження)	3–5 років	3–5 років	3–5 років
Частота	868,8 МГц (Європа), 915 МГц (США), 433 МГц (Азія)	868,8 МГц (Європа), 915 МГц (США)	700/800/900 МГц
Захист даних	AES-64 і 128 біт	AES з HMACS	–
Дальність	До 2,5 км у місті, до 45,0 км за його межами	До 10,0 км у місті, до 50,0 км за його межами	Інформація відсутня
Підтримка	LoRa Alliance, IBM, Cisco, Actility, Semtech	Sigfox, Samsung	3GPP, Ericsson, Nokia, Huawei

LoRaWAN має кілька ключових переваг, які роблять його кращим вибором у порівнянні з SigFox та NB-IoT. По-перше, LoRaWAN працює в безкоштовних ISM-діапазонах (868 МГц в Європі та 915 МГц у США), що усуває витрати на ліцензування, які є характерними для NB-IoT. Це також надає більшу гнучкість в налаштуванні, оскільки не потрібно отримувати ліцензію для пристроїв [6].

Термін автономної роботи LoRaWAN перевищує 10 років, що є важливим фактором для IoT-датчиків, адже довговічність дозволяє зменшити частоту заміни батарей. Вартість підключення до LoRaWAN, як правило, нижча порівняно з 4G та NB-IoT, що робить його економічно вигідним вибором, особливо для LPWAN-рішень.

LoRaWAN також пропонує гнучкість у конфігурації мереж, що дозволяє адаптувати рішення під конкретні потреби проєктів. Системи LoRaWAN забезпечують AES-шифрування для захисту передачі даних, що важливо для збереження конфіденційної інформації.

Крім того, LoRaWAN підтримується багатьма відомими компаніями, такими як IBM, Cisco та Samsung, що сприяє інтеграції з існуючими системами. Усі ці фактори роблять LoRaWAN привабливим варіантом для різних IoT-застосувань, особливо тих, що вимагають тривалого терміну служби, низького енергоспоживання та високої надійності.

1.6 Топологія LPWAN-мереж та архітектура мережі LoRaWAN

Підхід, який застосовується для створення LPWAN-мереж, нагадує принцип функціонування мобільних мереж. LPWAN-мережа використовує топологію «зірка» (рис. 1.2), на відміну від деяких технологій короткозонного зв’язку, які реалізують мережу у формі mesh-структури, де кожен пристрій взаємодіє з базовою станцією безпосередньо.

Основні недоліки mesh-мереж пов'язані з обмеженим динамічним діапазоном з'єднань, що виникає через високу швидкість передачі та низьку чутливість приймачів. Наприклад, деякі з'єднання ZigBee можуть стикатися з труднощами у передачі даних на відстань понад 20–30 м, через низьку потужність передавача та, відповідно, значне загасання сигналу. Крім того, для забезпечення надійної mesh-мережі потрібно використовувати значну кількість елементів [8].

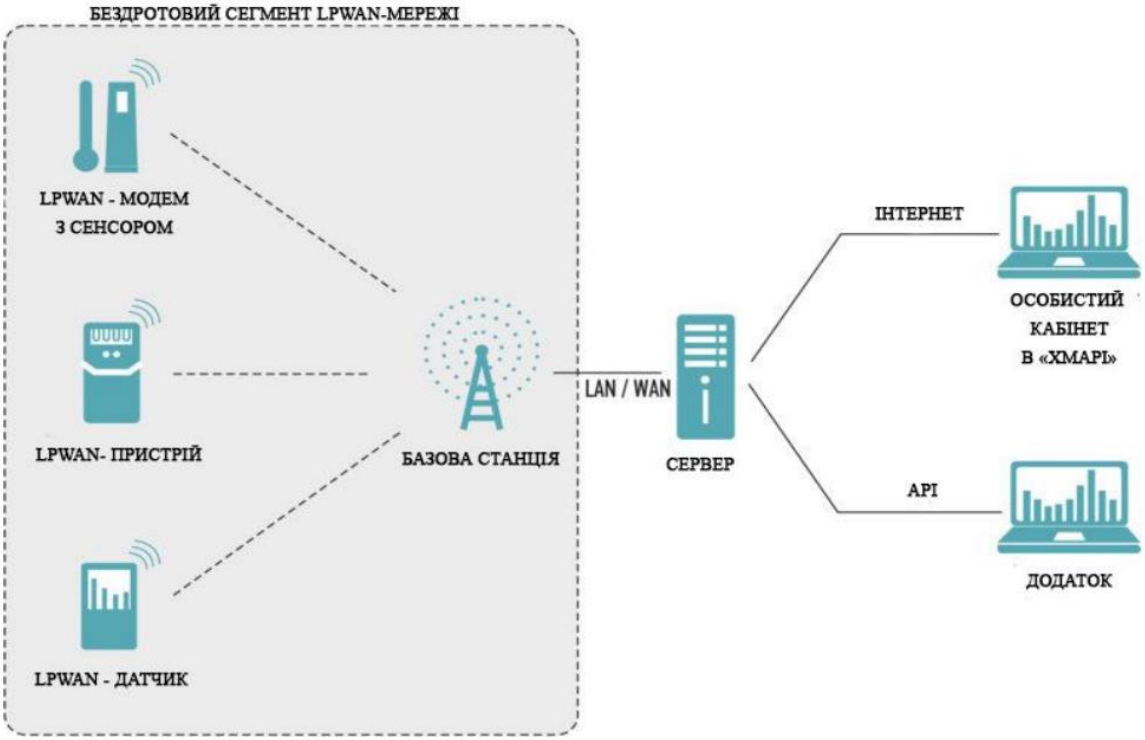


Рисунок 1.2 – Топологія LPWAN-мереж

Базові станції отримують та оцифровують сигнали від LPWAN-пристроїв, передаючи їх далі на сервер. На серверах дані з усіх станцій у мережі обробляються і подаються у зручному для користувача форматі. Зворотний канал зв’язку забезпечує можливість віддаленого керування пристроями [9].

Крім того, відмовостійкість усієї системи підвищується завдяки створенню єдиної точки відмови. Наприклад, у ZigBee-системах концентраційні пристрої встановлюються без запасу покриття, щоб уникнути збільшення вартості рішення. У разі відмови концентратора втрачається зв'язок з усіма датчиками в його радіусі дії. На відміну від цього, у LPWAN відмова одного кінцевого пристрою не впливає на роботу інших пристроїв у мережі.

Топологія бездротової технології LPWAN являє собою мережу «зірка з зірок» (рис. 1.3), де кожна базова станція отримує дані від різних датчиків і передає їх на сервер для подальшої обробки.

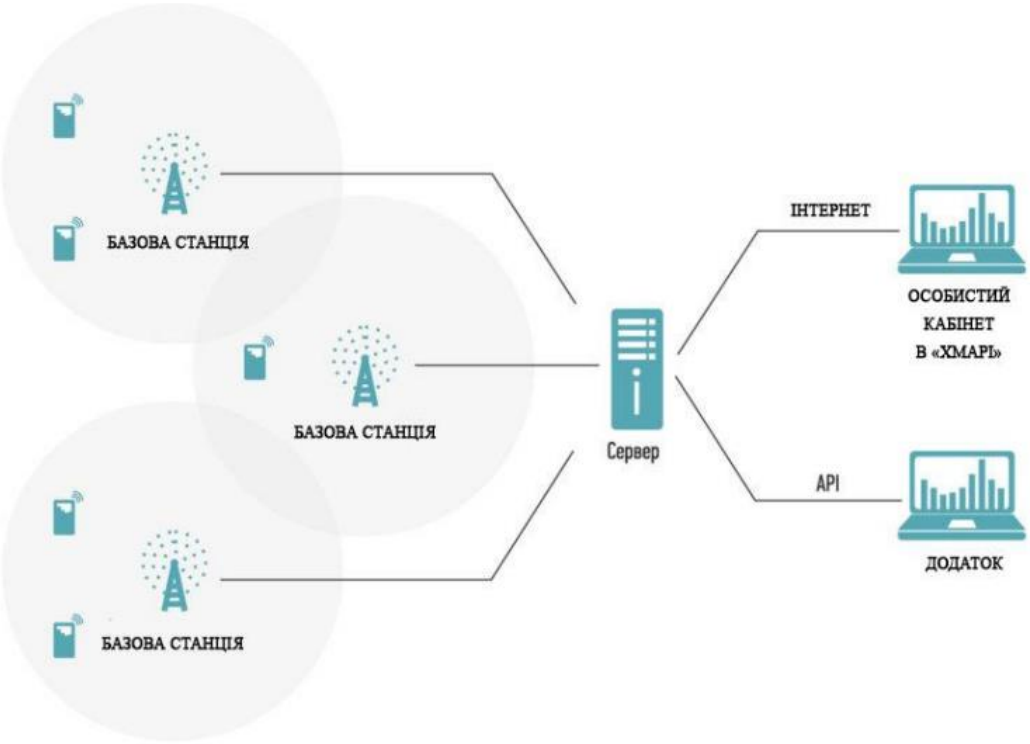


Рисунок 1.3 – LPWAN-мережа із топологією «зірка із зірок»

У мережах LoRaWAN кожен кінцевий пристрій безпосередньо підключається до шлюзу, що відрізняє їх від традиційних стільникових мереж. Це дозволяє оптимізувати енергоспоживання пристроїв та забезпечити тривалий термін їх служби. Мережевий сервер LoRaWAN керує процесом передачі даних, вибираючи оптимальний шлях для кожного повідомлення. Такий підхід забезпечує високу надійність передачі даних та можливість геолокації пристроїв (рис. 1.4).

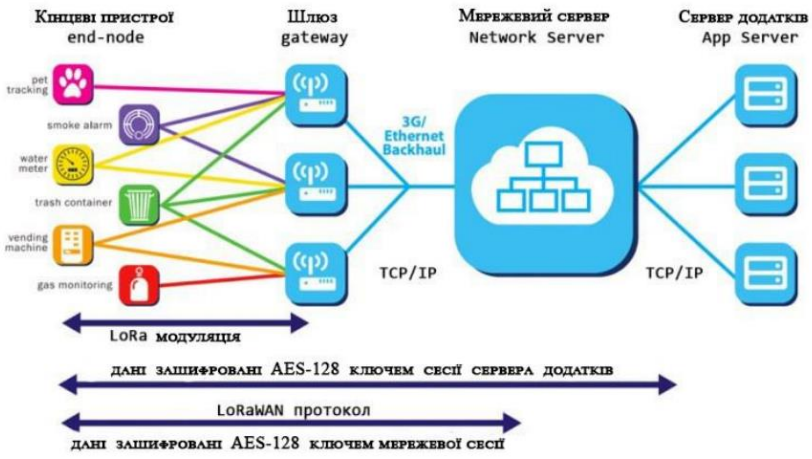


Рисунок 1.4 – Архітектура мережі LoRaWAN

Мережевий сервер LoRaWAN відіграє важливу роль у забезпеченні надійності та високої якості обслуговування. Він аналізує всі вхідні повідомлення, виявляє дублікати та вибирає оптимальний шлях для передачі даних. Це дозволяє підвищити пропускну здатність мережі та зменшити ймовірність втрати даних. Крім того, можливість приймати повідомлення від різних шлюзів підвищує стійкість мережі до перешкод.

1.7 Формування вимог до апаратно-програмного забезпечення

Під час формування специфікації вимог до апаратно-програмного забезпечення (АПЗ) для системи моніторингу на основі LoRa та одноплатних мікрокомп'ютерів необхідно враховувати такі аспекти:

Функціональні вимоги:

– забезпечення безперервного збору даних з сенсорів;

– підтримка багатоточкової передачі даних від кількох кінцевих пристроїв;

– забезпечення передачі даних у режимі реального часу або з мінімальними затримками;

– можливість віддаленого керування датчиками та отримання даних через центральний сервер.

Нефункціональні вимоги:

– енергоефективність: довгий термін роботи кінцевих пристроїв на одному заряді акумулятора;

– надійність: стійкість до збоїв у передачі даних, мінімізація втрат інформації;

– масштабованість: можливість розширення системи за рахунок підключення додаткових сенсорів без значних змін в архітектурі.

Сумісність

– взаємодія з одноплатними мікрокомп'ютерами, такими як Raspberry Pi;

– підтримка стандартів LoRaWAN та можливість інтеграції з іншими IoT-платформами.

Експлуатаційні вимоги:

– простота налаштування та управління системою, можливість швидкого розгортання в польових умовах;

– мінімальні вимоги до технічного обслуговування пристроїв.

Ці вимоги є основою для розробки АПЗ, що буде забезпечувати ефективний моніторинг та управління віддаленими датчиками з використанням LoRa-технології.

Висновки до розділу 1

У даному розділі було розглянуто основні характеристики та переваги технологій, що використовуються в мережах LPWAN, зокрема LoRaWAN. Аналіз топології, архітектури та принципів роботи цих мереж показав, що вони мають суттєві переваги в порівнянні з традиційними технологіями, такими як ZigBee та стільникові мережі.

По-перше, архітектура LoRaWAN, що базується на топології типу «зірка», забезпечує простоту та ефективність в управлінні зв'язком між кінцевими пристроями та шлюзами. Це дозволяє зменшити енергоспоживання, що, в свою чергу, продовжує термін служби акумуляторів пристроїв, що є критично важливим для багатьох застосувань IoT.

По-друге, унікальна функція прийому повідомлень будь-яким шлюзом, а також можливість використання кількох шлюзів для забезпечення безперервного зв'язку підвищують надійність і гнучкість мережі. Це робить LoRaWAN ідеальним вибором для різноманітних застосувань, таких як моніторинг віддалених об'єктів, управління інфраструктурою та реалізація рішень для смарт-міст.

Узагалі, з урахуванням усіх вищезазначених аспектів, можна зробити висновок, що мережі LPWAN є перспективним рішенням для побудови ефективних, економічних та надійних систем зв'язку для Інтернету речей.

2 МАТЕМАТИЧНА МОДЕЛЬ

2.1 Недорога бездротова технологія малої потужності

LoRa-мережа – це тип бездротової мережі, що споживає мало енергії та здатна передавати дані на великі відстані. Вона використовує асинхронну технологію зв’язку і алгоритми розширеного спектру, що дозволяє передавати дані на відстані до кількох кілометрів, навіть у густонаселених міських умовах. LoRa працює у безліцензійному діапазоні частот, призначеному для цілей ISM, і використовує декілька каналів для передачі даних, що забезпечує одночасне підключення багатьох пристроїв. Це робить LoRa підходящою для застосунків Інтернету речей і для завдань, що потребують далекої передачі даних, низького енергоспоживання та низьких витрат.

Однією з особливостей LoRa є невелика швидкість передачі даних, яка зазвичай становить від кількох кбіт/с до близько 50 кбіт/с. Проте можливість передачі даних на відстань до кількох кілометрів і низьке енергоспоживання є значними перевагами цієї мережі. Крім того, LoRa має функції безпеки, що забезпечують захист переданих даних. Протоколи передачі даних LoRa зашифровані, щоб запобігти злому та несанкціонованому доступу.

У більшості сучасних проєктів IoT для з'єднання та передачі інформації між двома точками A і B використовуються прямі з'єднання, які фокусуються на прямому з'єднанні двох кінцевих пристроїв через один канал [10]. Однак така мережева структура має кілька недоліків:

– відстань з'єднання обмежена, що ускладнює і обмежує передачу даних;

– оскільки структура має лише одне з'єднання, будь-яка несправність призводить до розриву зв'язку в системі.

З огляду на ці недоліки, доцільно замінити структуру «точка-точка» в LoRa на мережеву структуру. Мережева структура в LoRa використовує багато кінцевих пристроїв, і коли один з каналів виходить з ладу, мережа самостійно перенаправляє

дані по іншому шляху. Таким чином, між точками A і B завжди існуватиме резервний шлях, що гарантує стабільність з'єднання [11].

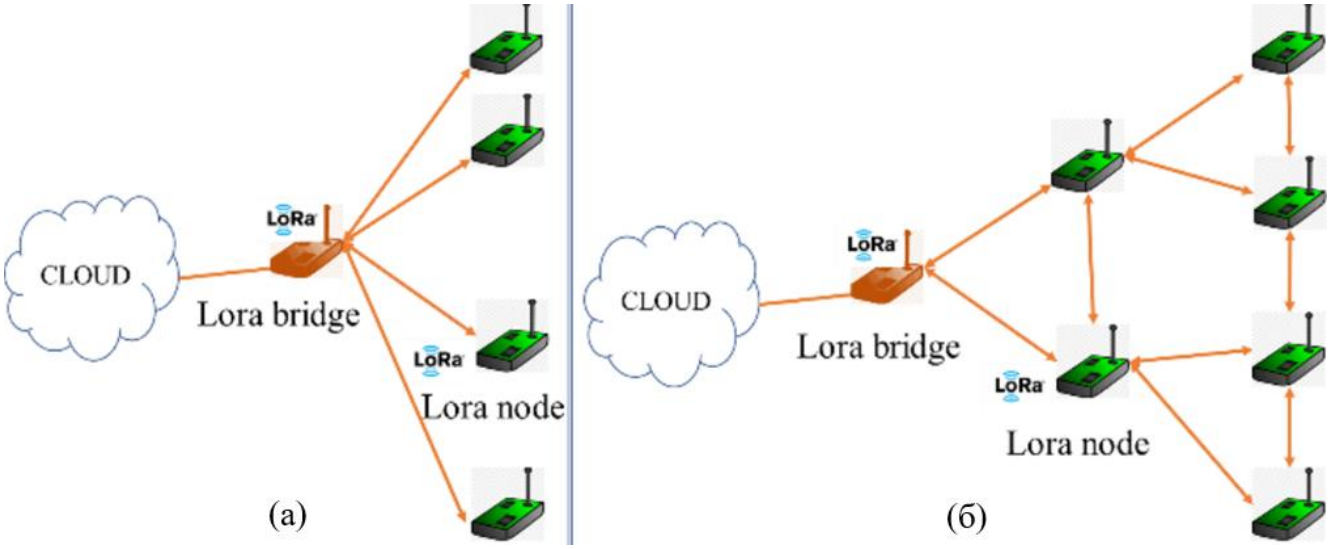


Рисунок 2.1 – Мережева структура в LoRa: а) мережа LoRaWAN;
б) мережа LoRa Mesh

LoRa Mesh – це розширена версія мережі LoRaWAN, де пристрої можуть не лише безпосередньо спілкуватися з шлюзом, але й об'єднуватися в мережу з топологією «mesh» (рис. 2.1). LoRa Mesh забезпечує масштабованість мережі, розширене покриття та підвищену надійність для мереж LoRaWAN.

У мережі LoRa Mesh пристрої можуть обмінюватися даними один з одним, що дозволяє мережі LoRaWAN функціонувати як система «mesh», збільшуючи покриття та підвищуючи надійність мережі. Пристрої в мережі LoRa Mesh можуть бути налаштовані як маршрутизуючі вузли, що сприяє оптимізації передачі даних і підвищенню масштабованості мережі. LoRa Mesh в режимі ad-hoc є самонавчальною безпровідною мережею, у якій пристрої автоматично формують і підтримують з'єднання між собою без участі контрольного центру.

У мережі LoRa Mesh кожен пристрій обладнано функцією маршрутизації, що дозволяє йому передавати інформацію між іншими пристроями в мережі, що сприяє підвищенню масштабованості та надійності мережі.

Кожен маршрутизатор у мережі LoRa Mesh використовує динамічні алгоритми маршрутизації, які автоматично налаштовуються для оптимізації передачі даних і зменшення затримок. Мережа LoRa Mesh також підтримує автоматичне з'єднання, що дозволяє новим пристроям приєднуватися до мережі без потреби в ручній конфігурації. Ця функція спрощує розгортання та управління мережею [13].

Мережа LoRa Mesh також с проєктована для економії енергії, оскільки пристрої можуть переходити в режим глибокого сну і автоматично прокидатися, коли є дані для передачі. Це допомагає подовжити термін служби батарей і знизити витрати на їх заміну. Мережа LoRa Mesh має безліч застосувань у різних сферах, включаючи моніторинг і управління тваринами, контроль за пристроями розумного дому, управління енергією в енергетичних системах тощо [12].

Чирпована хвиля – це форма хвильового сигналу, в якій частота змінюється з часом, створюючи вигляд, схожий на дзвін. У LoRaWAN чирповані хвилі використовуються для передачі даних через бездротові канали на великі відстані з низьким енергоспоживанням. Чирповані хвилі у LoRaWAN генеруються за допомогою техніки CSS. У CSS чирповані хвилі перетворюють радіочастотні (англ. Radio Frequency, RF) сигнали для створення особливого типу сигналу, відомого як «хвиля-літак». Під час передачі даних чирповані хвилі використовуються для кодування інформації та її передачі через RF-канали.

Чирпована хвиля в LoRaWAN може бути налаштована відповідно до вимог різних додатків, включаючи частоту передачі, довжину та відстань. Коли чирпована хвиля передається, сигнал підлягає впливу навколишнього середовища, що призводить до затухання, спотворення та шуму, внаслідок чого змінюється частота сигналу. Однак завдяки характеристикам чирпованї хвилі сигнал можна декодувати та відновити за допомогою відповідних алгоритмів обробки сигналів на приймачі.

Чирповані хвилі використовуються в багатьох комунікаційних застосуваннях, таких як радар, медична візуалізація та бездротові комунікації. Використання чирпованих хвиль у бездротових комунікаціях має багато переваг, зокрема стійкість до перешкод, зменшення втрат сигналу та підвищену здатність виявляти слабкі сигнали. Крім того, чирповані хвилі можуть також використовуватися для створення багатошляхового каналу, покращуючи зв'язок у середовищах з великою кількістю відбиваючих об'єктів [18].

Чирпована хвиля, або «chirp wave», зазвичай описується за допомогою математичної формули, яка відображає зміну частоти в часі. Загальна форма для чирпованої хвилі може бути представлена як:

$$s(t) = A * \sin(2\pi(f_0 t + \frac{k}{2}t^2)), \quad (2.1)$$

де s(t) – амплітуда чирпованої хвилі в момент часу t;

A – амплітуда хвилі (максимальне значення);

f_0 – початкова частота;

k – коефіцієнт, що визначає швидкість зміни частоти (зазвичай обчислюється як $k = \frac{f_1 - f_0}{T^2}$, де f^1 – кінцева частота, а T – час тривалості чирпованої хвилі).

Ця формула описує, як частота хвилі змінюється в часі, створюючи характерний «чирпований» сигнал [19].

Як показано на рисунку 2.2, мережа LoRa для передачі даних використовує метод зв'язку CSS. CSS є технікою бездротової комунікації, що забезпечує підвищену стійкість до перешкод та зменшує вплив сигналів перешкод на передавані хвилі. Сигнали, що випускаються вузлами, кодуються за допомогою CSS і передаються у вигляді радіохвиль на діапазоні ISM на частотах 433 МГц або 868 МГц. Оскільки ці сигнали мають дуже вузьку смугу пропускання (близько 125 кГц), вони можуть ефективно проходити крізь перешкоди, такі як стіни або інші об'єкти.

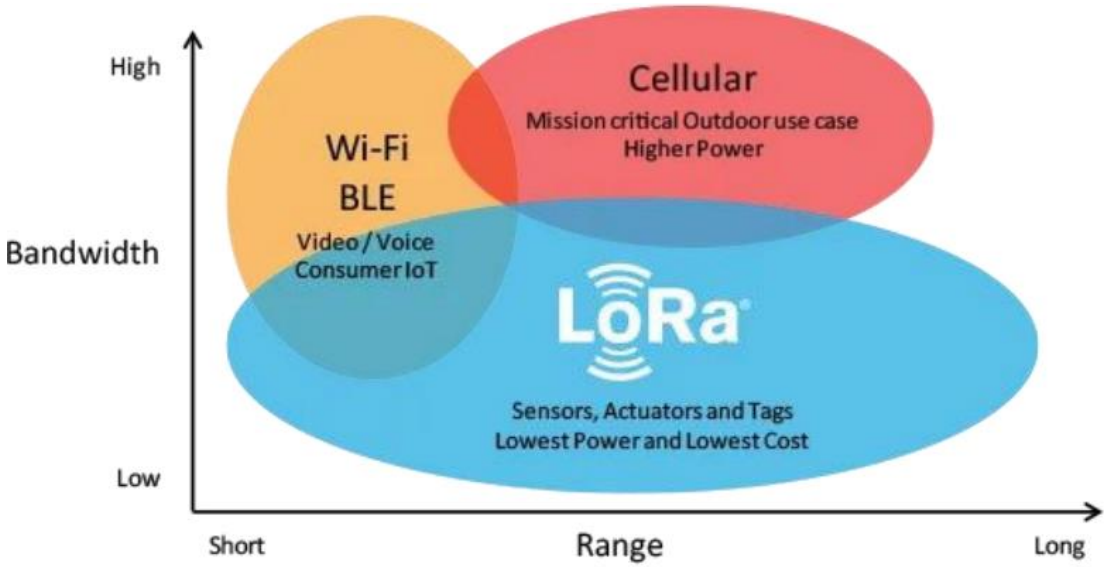


Рисунок 2.2 – CSS у LoRa в порівнянні з іншими бездротовими технологіями

Однак розповсюдження хвиль у мережі LoRa також може залежати від інших факторів, таких як висота антен, топографічні особливості та різна щільність об'єктів. У CSS дані кодуються за допомогою чирпованої хвилі, що збільшує ширину сигналу в порівнянні з початковою чирповано. хвилею, після чого він ділиться на кілька блоків і розподіляється на кількох частотах. Кількість частот, на які буде розподілено сигнал, залежить від смуги пропускання чирпованої хвилі та роздільної здатності сигналу. Кожна частота несе частину даних, а всі ці частоти об'єднуються для формування CSS сигналу [20].

Для передачі CSS сигналу частота сигналу змінюється безперервно у певному порядку, відомому як послідовність стрибків частоти. Ця послідовність генерується за допомогою заздалегідь визначеного алгоритму і ділиться на кілька сегментів. Під час передачі сигналу частота змінюється безперервно відповідно до порядку, в якому сегменти послідовності змінюють частоту.

За допомогою CSS сигнал розподіляється на кількох частотах і постійно змінює свої частоти, що значно підвищує його стійкість до шуму.

Таблиця 2.1 – Порівняння малопотужних бездротових технологій

Критерій	**Chirp (CSS)**	**UWB**	**BLE**	**Wi-Fi**
Точність локалізації, м	1,0–2,0	+/– 0,4	< 5,0	< 10,0
Діапазон, м	Оптимальний: 10–500 Макс: 1000	Оптимальний: 0–50 Макс: 200	Оптимальний: 0–25 Макс: 100	Оптимальний: 0–50 Макс: 500
Затримка, с	< 0,001	< 0,001	3–5	3–5
Споживання енергії	Дуже низьке, для портативних пристроїв	Низьке, для портативних пристроїв	Дуже низьке, для портативних пристроїв	Середнє
Вартість	$	$$	$$	$$$
Частотний діапазон, ГГц	ISM: 2,4 (2,4–2,4835)	3,1–10,6	2,4	2,4; 5
Швидкість передачі даних, Мбіт/с	До 2	До 27	До 2	До 1000

Крім того, розподіл частот та безперервна зміна частоти підвищують безпеку сигналу, оскільки без знання послідовності зміни частот іншим важко розшифрувати сигнал. Технологія CSS застосовується в багатьох комунікаційних додатках, включаючи GPS-навігаційні системи, мережі LoRa та IoT-застосунки [21].

2.2 Звичайні протоколи маршрутизації Ad-hoc Wireless Network

На рис. 2.3 представлено огляд класичних протоколів маршрутизації, поділених на три основні групи: проактивні, реактивні та гібридні. У межах цієї

роботи ми використовуємо реактивну групу для розробки та експериментування з мережею LoRa Mesh [21].

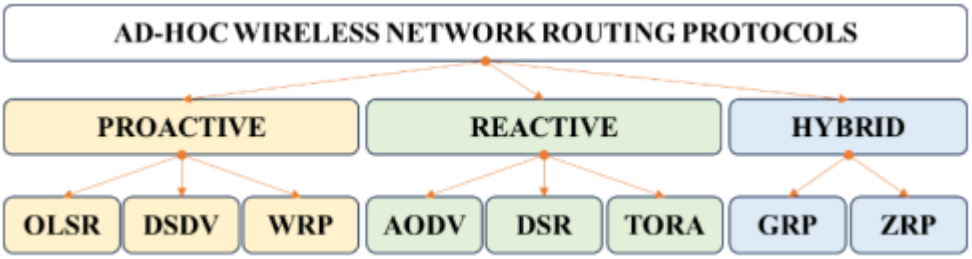


Рисунок 2.3 – Класифікація класичних протоколів маршрутизації

Динамічний маршрут з джерела (англ. Dynamic Source Routing, DSR) – це простий та ефективний протокол маршрутизації, спеціально розроблений для використання в багатохопкових бездротових ad-hoc мережах. DSR дозволяє мережі самостійно налаштовуватися та організовуватися без необхідності в існуючій інфраструктурі чи управлінні. Це реактивний протокол, де всі компоненти повністю працюють на вимогу. Протокол базується на концепції маршрутизації з джерела, в якій відправник пакета визначає повну послідовність вузлів, через які пакети пересилаються.

Алгоритм маршрутизації з тимчасовим порядком (англ. Temporally Ordered Routing Algorithm, TORA) є ініційованим джерелом реактивним протоколом маршрутизації. TORA є високоефективним, адаптивним, безциклічним і масштабованим протоколом, що ґрунтується на алгоритмі зміни зв’язків. Головна мета TORA – обмежити передачу повідомлень у динамічних мобільних обчислювальних середовищах. Це означає, що протокол розроблений для зменшення витрат на зв’язок, адаптуючись до локальних змін топології в мережі ad-hoc [9]. Ще одна ключова особливість протоколу TORA – локалізація контрольних пакетів в невеликій області (наборі вузлів) поблизу змін топології через розриви маршрутів. Таким чином, кожен вузол запитуваної мережі містить свою локальну топологію та інформацію про маршрути сусідніх вузлів. TORA підтримує кілька маршрутів для передачі пакетів між джерелом і вузлом призначення в мобільній ad-hoc мережі, що забезпечує можливості

мультипотокової маршрутизації. Дію TORA можна порівняти з водою, що тече вниз до вузла-отводу через мережу труб, які моделюють маршрути в реальному світі. З'єднання труб представляють вузли, самі труби представляють маршрутні зв’язки між вузлами, а вода в трубах представляє пакети, що течуть між вузлами через маршрутні зв’язки до призначення.

Протокол маршрутизації за запитом відстані векторів ad-hoc (англ. Ad-hoc On-Demand Distance Vector, AODV) призначений для бездротових мереж і мобільних телефонів ad-hoc. Цей протокол встановлює маршрути до місць на вимогу та підтримує як unicast, так і multicast. Він розроблений для ефективної роботи в мережах з великою кількістю вузлів, які нерівномірно розподілені [24].

Спеціальний вектор відстані на запит (AODV) – це чистий алгоритм маршрутизації за запитом. Вузли, які не беруть участі в певному каналі, не зберігають інформацію про маршрутизацію і не займаються обміном таблицями маршрутизації. Таким чином, кількість широкомовних пакетів, необхідних для створення маршруту через AODV, зведена до мінімуму на відміну від широкомовної передачі, яка використовується для підтримки повного маршруту, як у DSDV (Destination-Sequenced Distance-Vector Routing). Завдяки цьому мінімізується трафік маршрутизаційних пакетів у мережі.

Коли вихідний вузол має потребу надіслати повідомлення до вузла призначення, але не має маршруту до нього, він ініціює процес пошуку маршруту. Для цього вихідний вузол надсилає пакет запиту на маршрут (англ. Route Request, RREQ) усім своїм сусідам. Вузли, які отримують цей пакет, пересилають запит своїм сусіднім вузлам і так далі, доки вузол призначення або проміжний вузол із достатньою інформацією про маршрут не отримає повідомлення. На рис. 2.4 показано, як широкомовні пакети RREQ поширюються через мережу від вихідного вузла A до вузла призначення E [22].

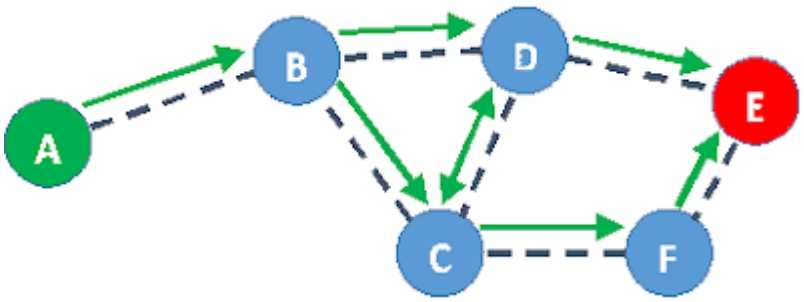


Рисунок 2.4 – Розповсюдження пакету запиту маршруту RREQ

Як і в маршрутизаційному алгоритмі DSDV, для запобігання циклам у маршрутах і забезпечення актуальності маршрутної інформації використовується порядковий номер призначення. Кожен вузол має свій унікальний порядковий номер і ідентифікатор трансляції, який збільшується щоразу при ініціюванні запиту маршруту. Ідентифікатор трансляції разом із IP-адресою вузла унікально визначає кожен пакет RREQ. До пакету RREQ включено таку інформацію про ідентифікацію вихідного вузла:

– номер кнопки;
– трансляційний ідентифікатор;
– останній порядковий номер вузла призначення.

Проміжні вузли відповідають на пакети RREQ тільки в тому випадку, якщо мають маршрут до пункту призначення з порядковим номером, що дорівнює або перевищує порядковий номер у пакеті RREQ. Щоб підвищити ефективність маршрутизації, проміжні вузли зберігають адреси своїх сусідів при першому отриманні широкомовного пакета RREQ, тим самим створюючи інформацію про зворотний шлях до вихідного вузла (зворотний шлях). Усі подальші пакети RREQ і отримані порядкові номери ігноруються. На рис. 2.5 показано зворотний шлях до вихідного вузла, оновлений у таблицях маршрутизації.

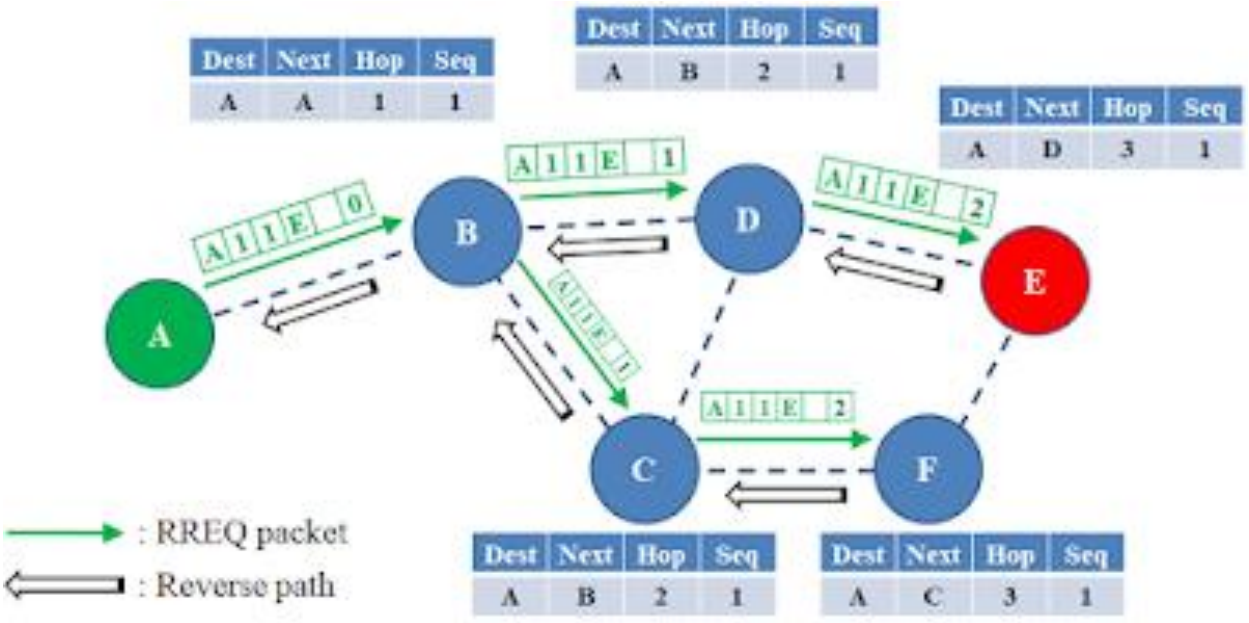


Рисунок 2.5 – Зворотний шлях до джерела

Як тільки пакет RREQ досягає пункту призначення або проміжного вузла, що має достатньо інформації про маршрут до пункту призначення, цей вузол надсилає відповідь маршруту (англ. Route Reply, RREP) назад до вузла, який отримав першу копію пакета RREQ. Рис. 2.6 ілюструє маршрут проходження пакета відповіді RREP.

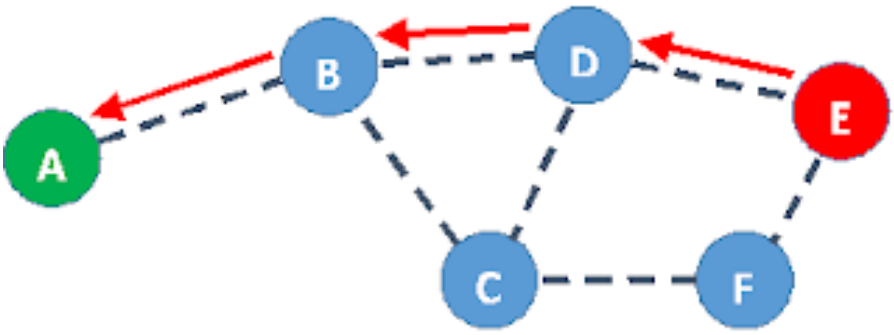


Рисунок 2.6 – Шлях пакета RREP

Повертаючись за встановленим зворотним шляхом, пакет RREP дозволяє вузлам на маршруті встановлювати інформацію про маршрут до вузла призначення (прямий шлях) у міру отримання пакета RREP. Ця маршрутна інформація до пункту призначення формує повноцінний шлях від вузла-джерела до вузла призначення. Пакети RREP продовжують рух зворотним шляхом, доки не

досягнуть вихідного вузла. Завдяки цьому AODV забезпечує підтримку двосторонньої передачі (симетричні канали). На рис. 2.7 зображено прямий шлях від вихідного вузла, представлений в таблицях маршрутизації.

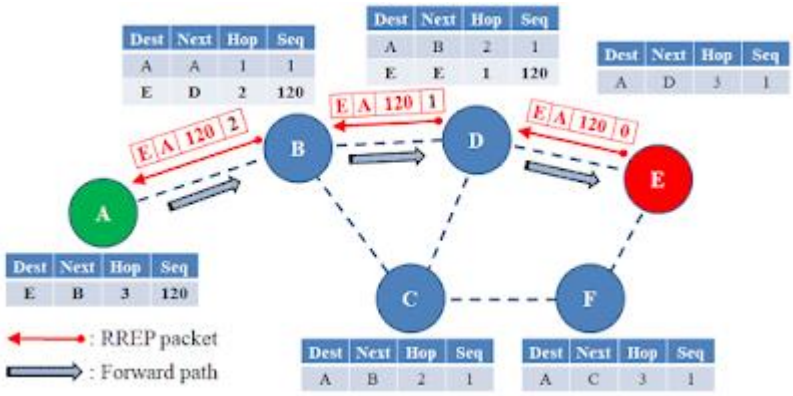


Рисунок 2.7 – Шлях від джерела до місця призначення (прямий шлях)

Кожен запис маршруту в таблиці маршрутизації має асоційований таймер. Якщо маршрут не використовується протягом заданого періоду, запис видаляється.

При зміні топології мережі (наприклад, переміщенні вузла) ініціюється процес динамічного маршрутизування для пошуку нового маршруту до призначення. Рух вузлів вздовж маршруту призводить до генерації повідомлень про збій лінку, які розповсюджуються по мережі до джерела. Отримавши таке повідомлення, джерело може перезапустити процес пошуку маршруту.

Протокол HELLO використовується для підтримки сусідських відносин та виявлення змін топології. Вузли періодично розсилають HELLO-пакети, щоб повідомити сусідів про свою доступність. За відсутністю відповіді на HELLO-пакети, вузол може припустити, що сусід недоступний і ініціювати процедуру пошуку альтернативного маршруту.

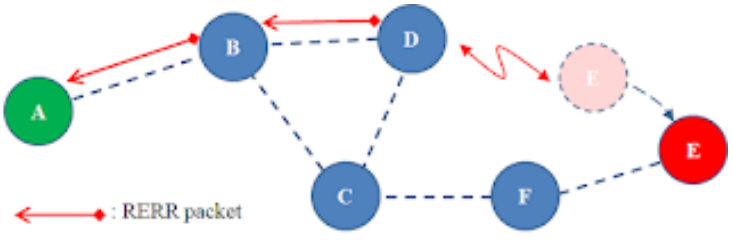


Рисунок 2.8 – Пакет помилок маршруту

Коли з’єднання розірвано і пакет не може досягти пункту призначення, вузол передає пакет помилки маршруту, скорочено RERR, назад до вузла джерела. Якщо вихідному вузлу все ще потрібен маршрут до пункту призначення, він перезапустить процес виявлення маршруту до пункту призначення. На рис. 2.8 показано, що з’єднання розривається, а пакет RERR надсилається назад до вихідного вузла.

2.3 Структура пакету

LoRa належить до технологій зв’язку для IoT, які використовуються для створення широкомасштабних мереж з низьким енергоспоживанням – LPWAN. LoRa орієнтована на застосунки, де віддалені датчики передають дані з низькими швидкостями передачі. Розмір пакета LoRa обмежений 255 байтами.

На рис. 2.9 показана структура пакета LoRa і пакета LoRaMesher. Видно, що пакет LoRaMesher інкапсульований у полі корисного навантаження пакета LoRa. Сам пакет LoRaMesher містить заголовок та корисне навантаження. Згідно з розробкою в [25], заголовок включає 4 байти для адреси вузла призначення і джерела, 1 байт для зазначення типу повідомлення і 1 байт для вказівки розміру корисного навантаження.

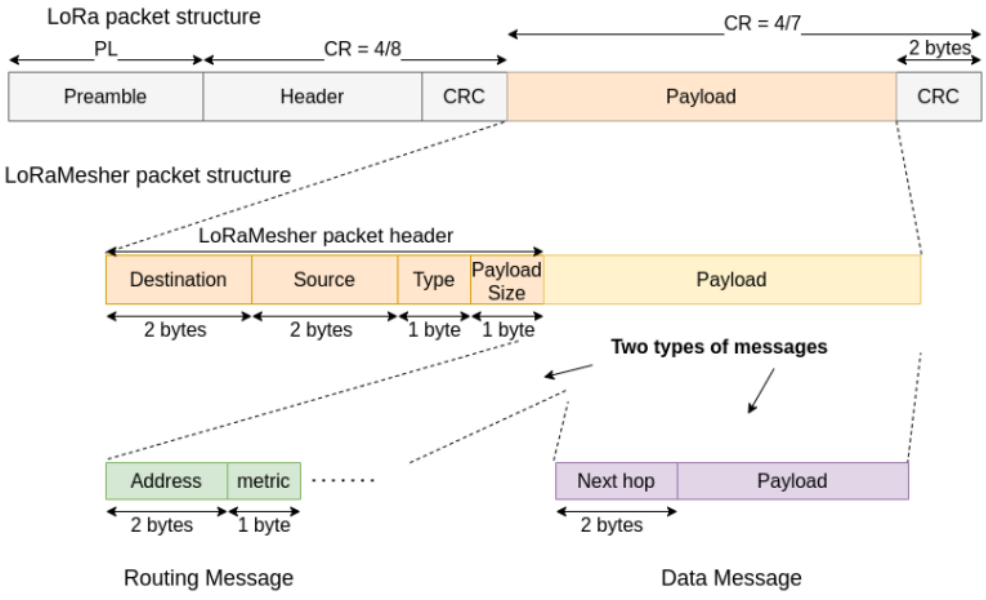


Рисунок 2.9 – Стандартний пакет LoRa

На рис. 2.10 показано алгоритм обробки отриманого маршрутизуючого пакета. Як тільки отримане повідомлення визначається як маршрутизуюче, вузол перевіряє, чи відправник уже записаний у таблиці маршрутизації. Далі обробляються записи таблиці маршрутизації, отримані від сусіднього вузла, для оновлення локальної таблиці маршрутизації. Після обробки пакет LoRaMesher з маршрутизуючим повідомленням видаляється.

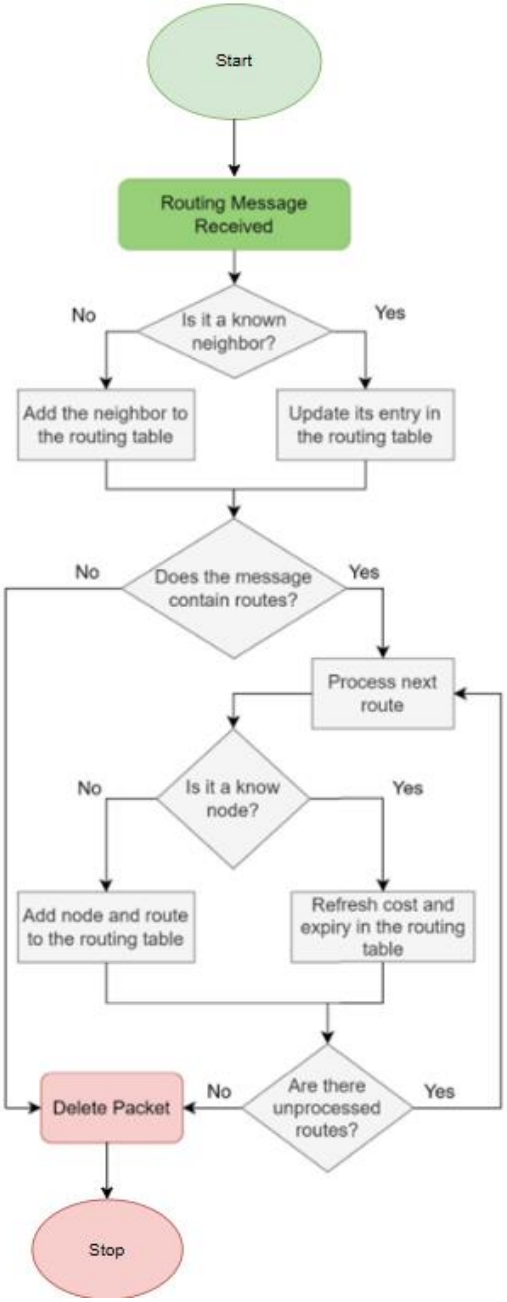


Рисунок 2.10 – Діаграма маршрутизації отриманого повідомлення

Повідомлення з даними містять інформацію з рівня додатків і мають конкретний пункт призначення в LoRa mesh-мережі. Коли вузол із бібліотекою LoRaMesher отримує таке повідомлення, можливі такі результати:

1) якщо цей вузол є пунктом призначення, бібліотека передає дані на рівень додатків;

2) якщо вузол не є пунктом призначення, але вказаний як наступний крок у маршруті, він повинен передати повідомлення далі, оновлюючи поле «наступного кроку» з адресою, знайденою в його локальній таблиці маршрутизації;

3) якщо вузол не є ні пунктом призначення, ні наступним кроком, він видаляє повідомлення з даними.

У другому випадку, якщо вузол має передати повідомлення на адресу, яка відсутня в його таблиці маршрутизації, повідомлення також буде видалено.

LoRaMesher розроблений для виконання шести завдань, що обробляють пакети LoRa. Кожне завдання реалізується за допомогою окремої процедури, яка працює з чергами, описаними раніше, для обробки пакетів. Пакети додаються та видаляються з черг відповідно до функцій завдання, і таким чином завдання обмінюються запитами та взаємодіють одне з одним.

Основна функція **Завдання прийому** – отримати LoRa-пакет, виділити його корисне навантаження, перетворити його в LoRaMesher-пакет, створити елемент у черзі пакетів і додати цей пакет до черги Q_RP. Після цього завдання надсилає сповіщення Завданню обробки про необхідність обробки нового пакета. Це завдання працює з інтервальною системою ISR, яка, за допомогою бібліотеки RadioLib, дозволяє відразу реагувати на отримання пакета.

Завдання прийому має найвищий пріоритет, що забезпечує максимальну кількість отриманих пакетів. Однак тривалість цього завдання критична, оскільки під час його виконання інші пакети не можуть бути отримані, тому оптимізація його роботи напряму впливає на максимальну пропускну здатність прийому пакетів. Основні кроки:

1) отримати LoRa-пакет;

2) виділити корисне навантаження LoRa-пакета, перетворивши його в LoRaMesher-пакет;

3) створити елемент у черзі пакетів і додати LoRaMesher-пакет;

4) додати елемент черги в Q_RP;

5) надіслати сповіщення Завданню обробки.

Завдання відправки витягує елемент черги пакетів із Q_SP, відправляє його та видаляє. Якщо пакет містить дані, перевіряється, чи призначення є в таблиці маршрутизації, і якщо так, оновлюється поле наступного кроку. Це завдання виконується щоразу, коли до Q_SP додається новий пакет. Щоб дотримуватися обмежень на робочий цикл, кожного разу, коли пакет відправляється, бібліотека обчислює час перебування пакета в ефірі, додає обов'язкову затримку перед відправкою наступного пакета, а також запускає процедуру виявлення активності каналу (CAD) для прослуховування LoRa-преамбул.

Якщо виявлена LoRa-преамбула, завдання чекає випадкову затримку і знову запускає CAD. Пакет відправляється лише за відсутності преамбули, що є наближенням алгоритму CSMA/CA. Завдання відправки має другий за важливістю пріоритет. Перед кожним відправленням пакетів вимикається ISR Завдання прийому, і після відправлення ISR знову активується.

Основні кроки:

1) отримати сповіщення про додавання пакета до Q_SP;

2) витягнути елемент черги пакетів із Q_SP;

3) якщо пакет містить дані, оновити поле наступного кроку, якщо адреса є в таблиці маршрутизації;

4) дочекатися випадкової затримки.

Завдання маршрутизації виконується періодично і створює пакет з маршрутизуючим повідомленням. Це повідомлення використовується для обміну таблицею маршрутизації вузла з сусідніми вузлами та для формування таблиці маршрутизації в кожному вузлі. Повідомлення маршрутизації містить таблицю маршрутизації відправника, яка складається з адрес вузлів і метрики, що

представляє собою кількість ходів, необхідних для досягнення вузла. Пріоритет цього завдання нижчий, ніж у Завдання відправки, але вищий, ніж у Завдання обробки. Періодичність виконання можна змінювати, налаштовуючи змінну *HELLO_PACKETS_DELAY*. У поточній реалізації таблиця маршрутизації підтримує лише шлях з найменшою кількістю хопів до досягнення призначення.

Виконання **Завдання обробки** ініціюється Завданням прийому. Щоразу, коли отримується пакет, Завдання прийому надсилає сповіщення Завданню обробки. Після отримання цього сповіщення Завдання обробки бере перший елемент черги пакетів з Q_RP і визначає, чи є пакет LoRaMesher маршрутизуючим або даними. Якщо це маршрутизуюче повідомлення, воно обробляється згідно з описаним вище алгоритмом. У разі, якщо це повідомлення даних, воно обробляється відповідно до рис. 2.11.

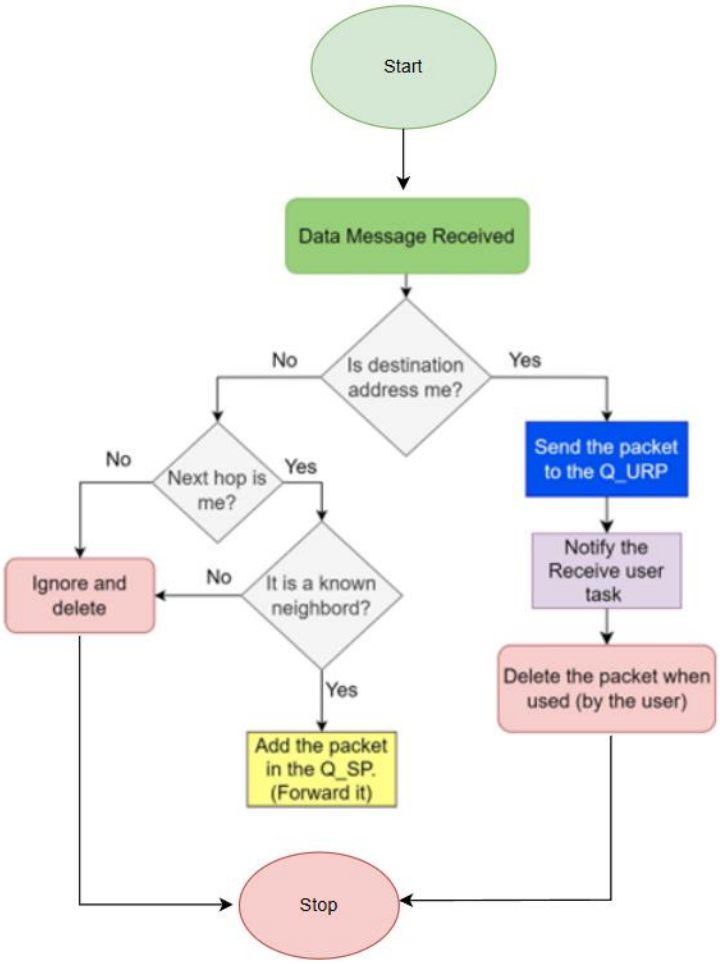


Рисунок 2.11 – Схема отримання повідомлення даних

Завдання відправки користувача реалізоване на рівні застосунку і відповідає за код, який здійснює передачу даних за допомогою бібліотеки LoRaMesher. Кожного разу, коли застосунок бажає надіслати повідомлення даних до іншого вузла, він повинен викликати функцію *createPacketAndSend*, де вказується адреса призначення, адреса пам'яті для корисного навантаження та кількість байтів, які потрібно надіслати. Якщо код програми використовує структури або класи в C++, які містять дані корисного навантаження, рекомендується працювати з вирівнюванням в один байт, щоб уникнути наявності порожніх байтів у ньому.

```
int dataTablePosition = 0;

for (;;) {
    //If the routing table is empty, wait 20000
        milliseconds and check again.
    if (radio.routingTableSize() == 0) {
        vTaskDelay(20000 / portTICK_PERIOD_MS);
        continue;
    }

    //If dataTablePosition is equal or greater
        than routingTableSize, start again from 0.
    if (radio.routingTableSize() <=
        dataTablePosition)
        dataTablePosition = 0;

    //Get the address from the routing table
    uint16_t addr =
        radio.routingTable[dataTablePosition].
            networkNode.address;

    //Create packet and send it.
    radio.createPacketAndSend(
        addr, userPayload, 1);

    dataTablePosition++;

    //Wait 120000 milliseconds to send the next
        packet
    vTaskDelay(120000 / portTICK_PERIOD_MS);
}
```

Рисунок 2.12 – Завдання відправлення користувача

Мета **Завдання отримання користувача** – отримати елементи з черги Q_URP і обробити пакети LoRaMesher відповідно до запитів з рівня застосунку. Завдання отримання користувача отримує сповіщення від бібліотеки щоразу, коли Завдання обробки отримує і визначає повідомлення даних для вузла. Це завдання реалізоване в коді програми та повинно містити *ulTaskNotifyTake(pdPASS, portMAX_DELAY)*,

функцію операційної системи FreeRTOS, яка дозволяє завданню залишатися в режимі сну до отримання сповіщення.

```
for (;;) {
    // Wait for the notification of Process Task
        and enter blocking
    ulTaskNotifyTake(pdPASS, portMAX_DELAY);

    // Iterate through all the packets inside the
        Q_URP
    while (radio.getReceivedQueueSize() > 0) {
        //Get the first element inside the
            Received User Packets FiFo
        AppPacket<dataPacket>* packet = radio.
            getNextAppPacket<dataPacket>();

        //Do something with the packet.
        printDataPacket(packet);

        //Delete the packet when used. It is very
            important to call this function to
            release the memory of the packet.
        radio.deletePacket(packet);
    }
}
```

Рисунок 2.13 – Завдання приймання користувача

Код програми має подбати про видалення повідомлень з черги, які більше не потрібні. Це видалення повідомлень важливе для правильного управління пам'яттю пристрою. Бібліотека LoRaMesher надає функцію *deletePacket(packet)* для видалення пакета.

У прикладі коду на рис. 2.13 показано, як інтегрувати Завдання отримання користувача в рівень програми. Якщо вузли, доступні в мережі LoRa, не відомі заздалегідь і їх потрібно виявити, то бібліотека LoRaMesher пропонує можливість читати таблицю маршрутизації вузла на рівні програми. Таким чином, програма може вибрати з допустимих адрес вузлів, до яких будуть надіслані повідомлення даних. Фрагмент коду на рис. 2.12 ілюструє цю інтеграцію.

Висновки до розділу 2

У цьому розділі розглянуто ключові аспекти роботи технологій LoRa та LoRaMesher, їх основні характеристики та можливості застосування в мережах Інтернету речей (IoT). LoRa, завдяки використанню методу розширення спектру

CSS, забезпечує ефективну передачу даних на великі відстані з низьким енергоспоживанням, стійкістю до перешкод та здатністю працювати у складних умовах.

LoRaMesher доповнює функціонал LoRa, дозволяючи створювати самоналаштовувані mesh-мережі, які підвищують надійність, масштабованість та ефективність зв'язку. Завдяки використанню протоколів маршрутизації, таких як DSR, TORA та AODV, забезпечується оптимальна передача даних між вузлами, а також адаптивність до змін топології мережі.

Шість основних завдань LoRaMesher автоматизують обробку пакетів, маршрутизацію та взаємодію між вузлами мережі. Реалізація завдань на рівні користувача спрощує інтеграцію бібліотеки в прикладні програми, дозволяючи ефективно керувати передачею і прийомом даних, забезпечуючи правильне управління пам’яттю пристроїв.

Таким чином, технології LoRa та LoRaMesher є перспективними рішеннями для побудови ефективних і масштабованих IoT-мереж, які можуть бути адаптовані для використання в розумних містах, сільському господарстві, системах моніторингу енергії та інших сферах, що потребують надійного зв’язку на великих відстанях.

3 ТЕХНІЧНЕ АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

У даному розділі розглядається питання технічно-апаратного оснащення системи дальнього радіозв’язку на основі одноплатних мікрокомп’ютерів. Було проведено порівняння та пошук даних щодо використання мікроконтролерів як основи для побудови мережі. Досліджується питання вибору та підключення LoRa-модулю.

3.1 Вибір апаратної частини

Завдання полягає у зборі даних на великій території та ще не охопленої інтернетом, що обчислюються кілометрами без використання Wi-Fi та мережі Ethernet.

У вирішенні цього завдання допоможуть радіомодулі, що передають дані з використанням технології зв'язку на великі відстані (Long Range, LoRa). Ця технологія запатентована компанією Semtech і реалізована в мікросхемах приймачів (трансіверів), таких як SX1268, SX1276, SX1278.

Залежно від вихідної потужності передавача, типу антени, робочої частоти, наявності прямої видимості чи перешкод для проходження радіохвиль як будинків, лісу, перешкод із боку інших джерел радіовипромінювання та інших чинників дальність може становити від сотень метрів до десятків кілометрів.

У законодавстві України існують обмеження щодо використовуваних частот, рівнів потужності та часу передачі. Для використання протоколу LoRa необхідно знайти перелік та умови застосування безліцензійних діапазонів частот (ISM), визначених законодавством України.

В Україні правила використання радіочастотного спектра регулюються Національною комісією, що здійснює державне регулювання у сфері зв'язку та інформатизації (НКРЗІ). До безліцензійних діапазонів частот, які можуть бути використані для протоколу LoRa, належать 433 МГц, 868 МГц, 2400 МГц та 5725 МГц [23].

Для створення мереж Інтернету речей в Україні використовується діапазон частот 868–869 МГц з максимальною потужністю до 25 мВт. У цьому діапазоні виділяються піддіапазони:

– 868,0–868,6 МГц із додатковими обмеженнями: час активної передачі сигналу не повинен перевищувати 1 % від загального часу;

– 868,7–869,2 МГц – обмежень немає.

Оскільки законодавство в цій сфері може змінюватися, важливо регулярно відстежувати актуальні норми, опубліковані на офіційних ресурсах НКРЗІ.

Метод модуляції LoRa використовує техніку розширення спектру (англ. Spread Spectrum Modulation, SSM) та варіацію лінійної частотної модуляції (англ. Chirp Spread Spectrum, CSS). При цьому дані кодуються широкосмуговими імпульсами із частотою, що змінюється на деякому часовому інтервалі. Також застосовується пряма корекція помилок (англ. Forward Error Correction, FEC). Велика дальність зв'язку при невисокій потужності передавача досягається тому, що протокол LoRa приймає сигнали нижче рівня шуму

У табл. 1 перераховані деякі модулі виробництва EBYTE з інтерфейсом UART, розраховані різні частотні діапазони і потужність.

Таблиця 3.1 – Модулі EBYTE із технологією LoRa

Модуль	**Частота, МГц**	**Потужність, дБм/мВт**	**Відстань, км**	**Чип**
E30-170T20D	148,000–173,500	10–20/10–100	2	S14463
E32-170T30D	160,000–173,500	21–30/125–1000	8	SX1278
E22-400T22D	410,125–493,125	22/158	10	SX1268
E22-400T30D	410,125–493,125	30/1000	12	SX1268
E32-433T20D	410,000–441,000	10–20/10–100	3	SX1278
E32-433T27D	410,000–441,000	18–27/63–500	5	SX1278

Модуль	Частота, МГц	Потужність, дБм/мВт	Відстань, км	Чип
E32-433T30D	410,000–441,000	21–30/125–1000	8	SX1278
E32-868T20D	862,000–893,000	10–20/10–100	3	SX1276
E32-868T30D	862,000–893,000	21–30/125–1000	12	SX1276
E32-915T20D	862,000–893,000	10–20/10–100	3	SX1276
E32-915T30D	862,000–893,000	21–30/125–1000	10	SX1276

Модуль E32-868T30D є оптимальним вибором для побудови мережі завдяки його технічним характеристикам, що забезпечують надійність і широкий радіус дії. Він працює в частотному діапазоні 862–893 МГц, що відповідає безліцензійним стандартам в Україні, дозволяючи уникнути регуляторних обмежень. Завдяки потужності передачі в межах 21–30 дБм (125–1000 мВт), модуль забезпечує стабільний зв'язок на відстанях до 12 км, що робить його ідеальним для застосувань, де потрібне широке покриття. Крім того, використання чипа SX1276 гарантує сумісність із популярними LoRa-протоколами, забезпечуючи гнучкість у розробці інтернету речей та інших бездротових рішень. Таким чином, E32-868T30D є універсальним і потужним рішенням для проєктів, що вимагають максимальної ефективності та надійності.

Потужність модулів може регулюватися в межах, зазначених у таблиці 1. Існують також способи збільшення дальності без збільшення потужності передавача. Це використання антен з великим посиленням, спрямованих антен та ретрансляторів. Варто відзначити, що компанія EBYTE оснащує свої модулі LoRa термокомпенсованим кварцовим генератором (TCXO), який забезпечує підвищену якість зв'язку в умовах зовнішнього середовища.

На модулі EBYTE є сім контактів: M0, M1, RXD, TXD, AUX, VCC та GND, а також роз'єм антени (рис. 3.1).

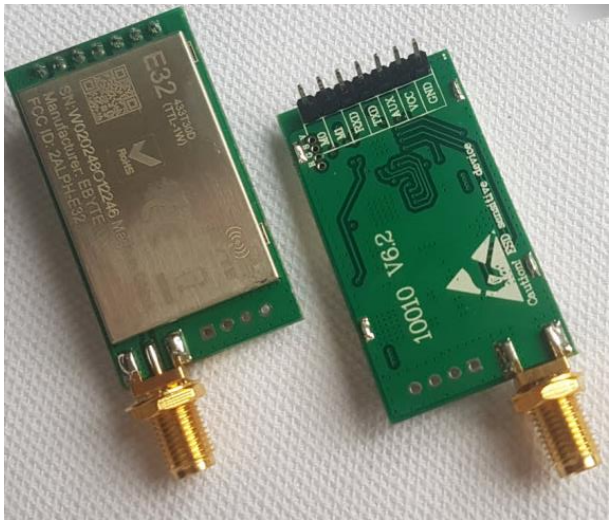


Рисунок 3.1 – Модуль LoRa E32-868T30D [15]

M0 (Mode 0): цей контакт використовується для вибору режиму роботи модуля разом із контактом M1. Залежно від того, як ці два контакти підключені (до 0 або 1 логічного рівня), модуль переходить у відповідний режим роботи:

– якщо M0 підключений до логічного 0, модуль може працювати в нормальному режимі;

– якщо підключений до логічного 1, разом із M1 визначає режим конфігурації або передачі. M0 разом із M1 утворюють комбінацію для чотирьох основних режимів роботи.

M1 (Mode 1): додатковий контакт для вибору режиму роботи. У поєднанні з M0 задає конкретний режим функціонування. Наприклад:

– 00: нормальний режим (Normal Operation Mode) – передача і прийом даних у стандартному режимі;

– 01: режим пробудження (Wake-up Mode) – використовується для більш швидкої передачі після виходу зі сплячого режиму;

– 10: режим конфігурації (Configuration Mode) – дозволяє змінювати параметри, такі як частота, швидкість передачі, потужність тощо;

– 11: сплячий режим (Sleep Mode) – мінімізує споживання енергії, підходить для пристроїв із живленням від батарей.

RXD (Receive Data): це вхідний контакт для приймання даних від мікроконтролера. RXD призначений для підключення до TX пінів контролера (наприклад, Arduino). Модуль використовує цей контакт для прийому інформації, яку потрібно передати по LoRa-протоколу до іншого пристрою.

TXD (Transmit Data): це вихідний контакт для передачі даних від LoRa модуля до мікроконтролера. TXD підключається до RX піну мікроконтролера, наприклад, Arduino, щоб мікроконтролер міг отримувати дані, що передаються через LoRa-зв’язок.

AUX (Auxiliary): допоміжний контакт, що сигналізує про стан модуля. Наприклад:

– сигналізує про завершення передачі даних;

– інформує про готовність модуля до роботи або перехід у новий режим;

– використовується для синхронізації або індикації стану. Наприклад, коли AUX знаходиться в низькому стані (логічний 0), це може означати, що передача ще триває.

VCC: це контакт для живлення модуля. LoRa модулі зазвичай працюють від напруги 3,3 В або 5 В, залежно від їхньої специфікації. Забезпечення стабільного живлення є критичним для коректної роботи модуля.

GND (Ground): контакт землі (нульовий потенціал), який є спільною точкою для електричних з’єднань. Правильне підключення до GND необхідне для забезпечення електричної стабільності модуля і для коректної роботи схем з мікроконтролером.

Для плат Arduino на 5 В (Nano, Uno, Mega) рекомендується використовувати резистори 1–5 К на контактах AUX і TXD, щоб запобігти вигоранню. Було протестовано плати Nano та Uno і не виявлено жодних проблем із вигоранням. Однак виробник пропонує використовувати підтягування, щоб уникнути небезпечних ситуацій. Крім того, варто зауважити, що LiPo безпосередньо живить модулі LoRa, тому що вони потребують 100–1000 мВт залежно від модуля, і це занадто багато, щоб вимагати від плат Arduino. Більше того, якби підключався

модуль потужністю 1 Вт до підвищувального перетворювача, він би теж ймовірно згорів.

Ще одна річ, на яку слід звернути увагу: модуль потужністю 1 Вт може працювати краще з вищою напругою – у табл. 3.1 даних для E32-868T30D зазначено, що 5 В може досягти найкращої вихідної потужності, але я цього не перевіряв. Усе, що тут зазначено, базується на кожному модулі LoRa, що живиться від 3,7 В і має необмежений струм.

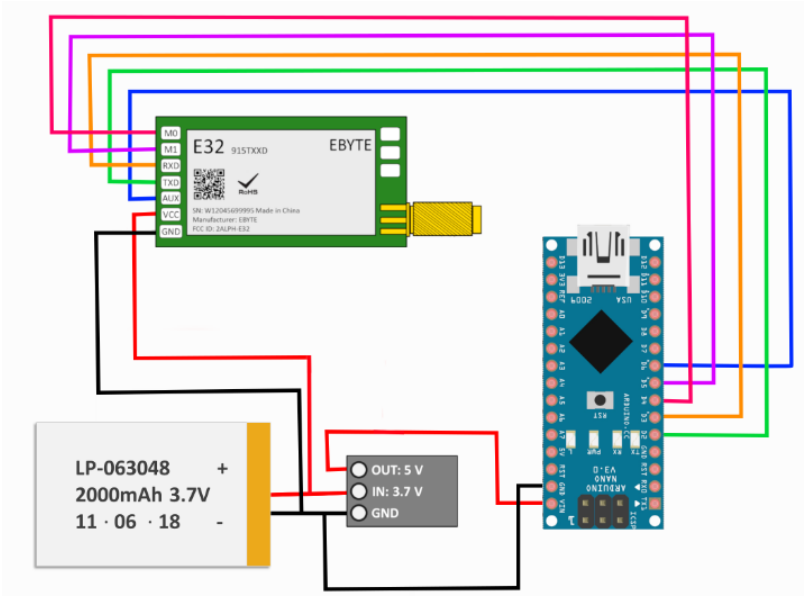


Рисунок 3.2 – Схема підключення

На рис. 3.2 показано схему підключення модуля E32-915T30D (LoRa-модуль), плати Arduino Nano, та акумулятора типу LiPo (3,7 В; 2000 мАг) через модуль живлення.

Акумулятор підключений до модуля живлення. Живлення з акумулятора надходить на вхідний роз'єм модуля (позначений як IN+ і IN-). Вихідний роз'єм модуля живлення (OUT 5V і GND) забезпечує стабільне 5 В для живлення інших компонентів.

Вихідна напруга 5 В із модуля живлення підключена до виводу `VIN` плати Arduino Nano, забезпечуючи її роботу, також до LoRa-модуля на контакт `VCC` (живлення). Загальний провід (GND) також з'єднаний між акумулятором, LoRa-модулем та Arduino Nano.

LoRa-модуль E32-915T30D підключений до Arduino Nano. Контакти M0 та M1 LoRa-модуля під'єднані до відповідних цифрових пінів Arduino Nano D2 і D3. Контакт AUX модуля підключений до іншого цифрового піна на Arduino Nano. Контакти TX та RX LoRa-модуля підключені відповідно до пінів RX та TX на Arduino Nano для UART-комунікації. LoRa-модуль отримує 5 В живлення через контакт VCC, а контакт GND з'єднаний із загальною землею (GND).

Схема забезпечує зв’язок LoRa-модуля з Arduino Nano для передачі даних, а також стабільне живлення від акумулятора через модуль перетворення напруги.

Антену для передавача буде використано стандартну штирьову для частоти передачі даних 868 МГц. Оптимальна довжина антени розраховується за формулою:

$$L = \frac{c}{f * \sqrt{\varepsilon}}, \quad (3.1)$$

де L – довжина антени (метри);

c – швидкість світла у вакуумі;

f – частота роботи антени (герци, Гц). Для модуля E32-868T30D частота становить 868 МГц;

ε – відносна діелектрична проникність середовища навколо антени (безрозмірна величина).

Розрахунки:

Частота: $f = 868 * 10^6$.

Діелектрична проникність: $\varepsilon = 1$ (вважаємо, що антена знаходиться у повітрі).

Отже, розрахуємо довжину хвилі (λ):

$$\lambda = \frac{c}{f} = \frac{3 * 10^8}{868 * 10^6} = 0{,}345 \text{ м}. \quad (3.2)$$

Це повна довжина хвилі. Для штирьової антени зазвичай використовується 1/4 довжини хвилі:

$$L = \frac{\lambda}{4} = \frac{0{,}345}{4} = 0{,}086 \text{ м} = 8{,}6 \text{ см}. \quad (3.3)$$

Отже, оптимальна довжина антени для частоти 868 МГц становить приблизно 8,6 см у повітрі.

Рисунок 3.3 – Приклад антени [16]

Таблиця 3.2 – Характеристики антени

Діапазон частот	868–915 МГц
Центральна частота	895 МГц
Посилення	5 dBi
КСВ	≤ 1,5
Опір	50 Ом
Матеріал корпусу	ABS
Матеріал роз'єму	Мідь
Робоча температура	Від мінус 40 °C до + 85 °C
Відносна вологість	до 95 %

Ця антена ідеально підходить для використання в мережах LoRaWAN завдяки своєму робочому діапазону частот 868–915 МГц, що відповідає стандартам бездротового зв'язку. Посилення у 5 dBi забезпечує стабільний сигнал і збільшену дальність передачі, а низький коефіцієнт стоячої хвилі (КСВ ≤ 1,5) мінімізує втрати сигналу. Висока стійкість до екстремальних температур (від мінус 40 °C до + 85 °C) та вологості до 95 % дозволяє використовувати антену в різноманітних кліматичних умовах, що робить її надійним вибором для зовнішніх та промислових застосувань.

3.2 Налаштування модулю LoRa

Перед тим як зібрати всю систему разом, необхідно один раз виконати попереднє налаштування модуля LoRa. Ця процедура є важливою, оскільки без коректного налаштування модуль може не працювати на бажаних частотах, потужностях чи режимах зв'язку.

З цією задачею допоможе плата конвертор яка конвертує TTL на USB (рис. 3.4)

Рисунок 3.4 – Конвертор з TTL на USB [17]

Для конфігурації буде використовуватись спеціальна програма для конфігурації модулів від компанії EBYTE (рис. 3.5).

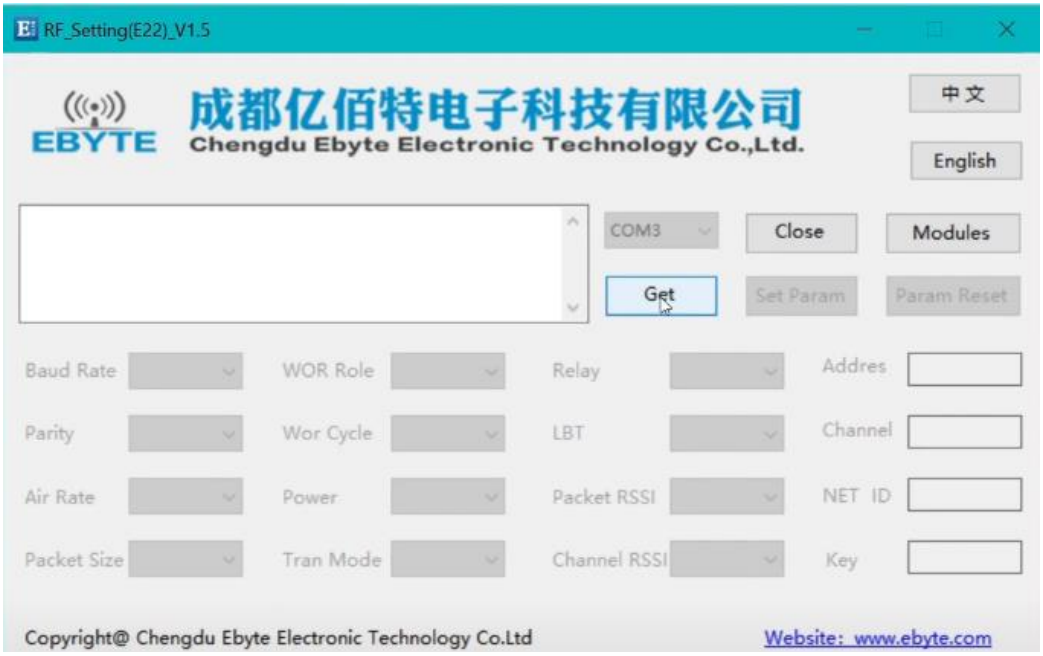


Рисунок 3.5 – Стартове вікно програми

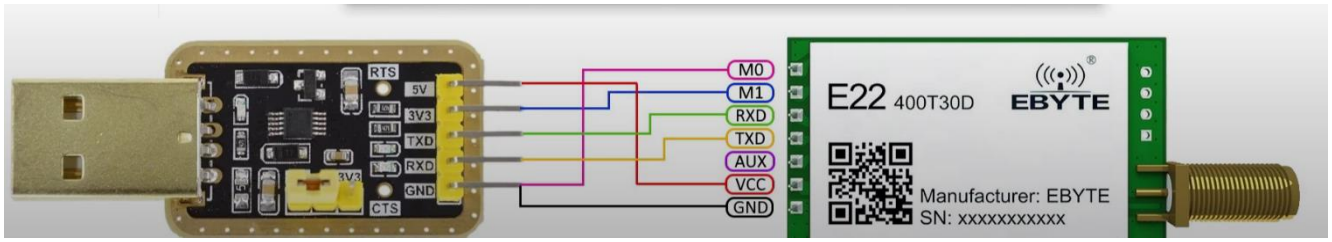


Рисунок 3.6 – Підключення модулю LoRa для конфігурації

Зв'язки між адаптером і LoRa-модулем реалізовані наступним чином: контакт 5 В на адаптері підключено до VCC модуля для забезпечення живлення, а контакт GND-адаптера з'єднано з GND-модуля для спільного заземлення. Для передачі даних контакт TXD-адаптера підключається до RXD-модуля, а RXD-адаптера з'єднується з TXD-модуля для забезпечення зворотної передачі інформації. Контакти M0 і M1 модуля з'єднуються з відповідними контактами адаптера, що дозволяє налаштовувати режими роботи модуля. Додатково, контакт AUX-модуля підключений до адаптера для зчитування стану модуля.

У представленому інтерфейсі програми (рис. 3.5) для налаштування модуля EBYTE доступні такі параметри. Baud Rate визначає швидкість передачі даних між модулем і мікроконтролером, виставляємо значення на 9,6 кбіт/с для того, щоб модуль спілкувався з Arduino як можна швидше.

Parity біт чітності виставляємо такий, як на Arduino, а саме 8N1. Air Rate регулює швидкість передачі даних по радіоканалу, що впливає на дальність і надійність зв’язку, тому для оптимальної дальності в 12 км для модуля вибираємо швидкість 2,4 кбіт/с. Packet Size задає максимальний розмір пакету даних для передачі. Виставляємо значення на 240 байт (максимальне значення) для того, щоб якщо буде передано 300 байт інформації, 240 було відправлено першим пакетом, а 60 байт – другим пакетом.

WOR Role визначає режим енергозбереження: модуль може працювати як передавач або приймач у сплячому режимі. WOR Cycle задає інтервал пробудження для прийому сигналу в режимі енергозбереження.

Power налаштовує потужність передачі сигналу, що впливає на дальність і енергоспоживання, ставимо на максимальне значення. Tran Mode дозволяє вибрати нормальний режим передачі або спеціальний (для тестування чи іншої специфіки). Relay вмикає або вимикає функцію ретрансляції для збільшення радіусу дії. LBT (Listen Before Talk) активує перевірку вільного каналу перед передачею, щоб уникнути конфліктів. Packet RSSI та Channel RSSI дозволяють увімкнути показники сили сигналу на рівні пакету або каналу відповідно, але відловлювати цей самий байт дуже важко, потрібно знати заздалегіть розмір отриманих даних, тому ця функція використовуватись не буде. Key задає ключ для шифрування даних, забезпечуючи їх безпеку під час передачі.

Address використовується для ідентифікації конкретного пристрою в мережі значення можна встановити в діапазоні від 0 до 65535. NET ID визначає ідентифікатор мережі для розділення кількох груп пристроїв. Channel встановлює частотний канал зв'язку, він працює наступним чином – якщо буде обрано 23 канал, то частотний канал дорівнюватиме 868,125 + 23 = 891,125.

3.3 Arduino Nano

Даний мікроконтролер є функціональним аналогом Arduino UNO. На рис. 3.7. наведено схематичне зображення плати.

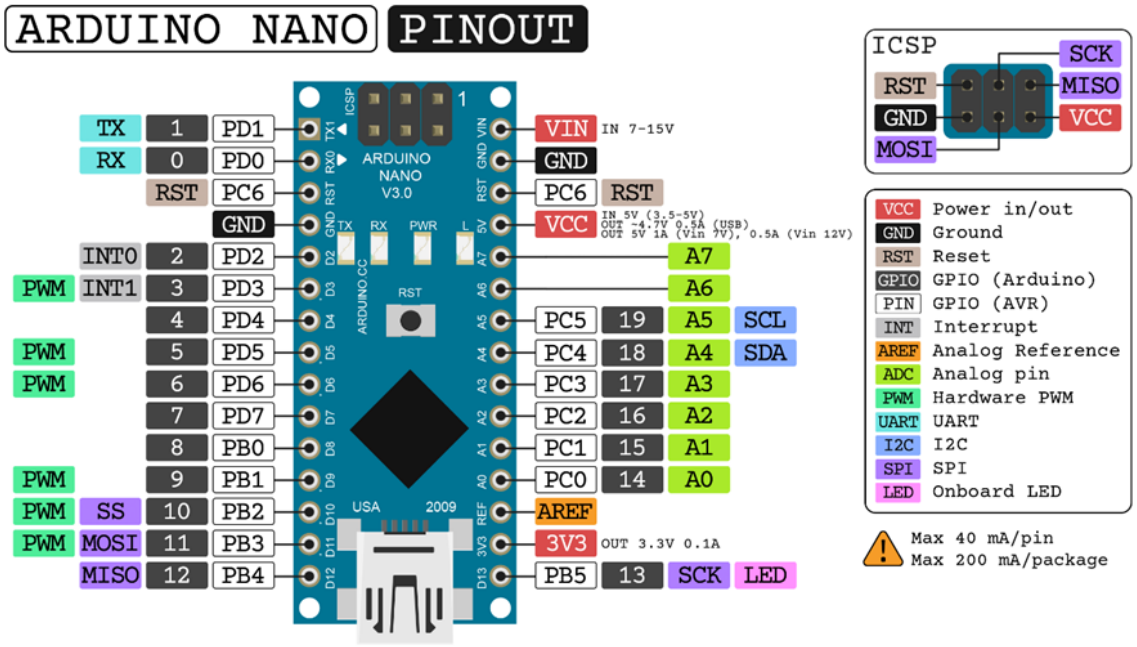


Рисунок 3.7 – Схема контактів Arduino Nano

Залежно від ревізії Arduino Nano може бути реалізований на базі ATmega328 (третя ревізія) або ATmega168 (друга ревізія). Розглядались технічні характеристики Arduino Nano третьої ревізії, на базі ATmega328:

– напруга живлення: рекомендована 7–12 В, гранична 6–20 В;
– споживання: 19 мА;
– FLASH-пам’ять: 32 кбайт;
– SRAM-пам’ять: 2 кбайт;
– EEPROM: 1 кбайт;
– тактова частота: 16 МГц;
– габарити: 45 мм × 18 мм;
– 14 цифрових входів/виходів та 8 аналогових.

Arduino Nano третьої ревізії на базі ATmega328 ідеально підходить для проєктів з обмеженим простором та низьким енергоспоживанням. Її компактні габарити (45 мм × 18 мм) дозволяють інтегрувати її у невеликі пристрої, а підтримка живлення від 7 до 12 В забезпечує гнучкість у виборі джерела живлення. Низьке споживання енергії (19 мА) робить її ефективною для автономних пристроїв.

3.4 Збірка системи

Після початкового налаштування LoRa-модуля через спеціальну програму наступним кроком є підготовка Arduino до роботи з модулем. Для цього необхідно написати і завантажити на Arduino скетч, який дозволить організувати передачу даних між двома системами через LoRa-з’єднання. Такий код забезпечить обмін інформацією, враховуючи всі особливості роботи з модулем, наприклад, налаштування з’єднання, передачу команд і обробку отриманих даних.

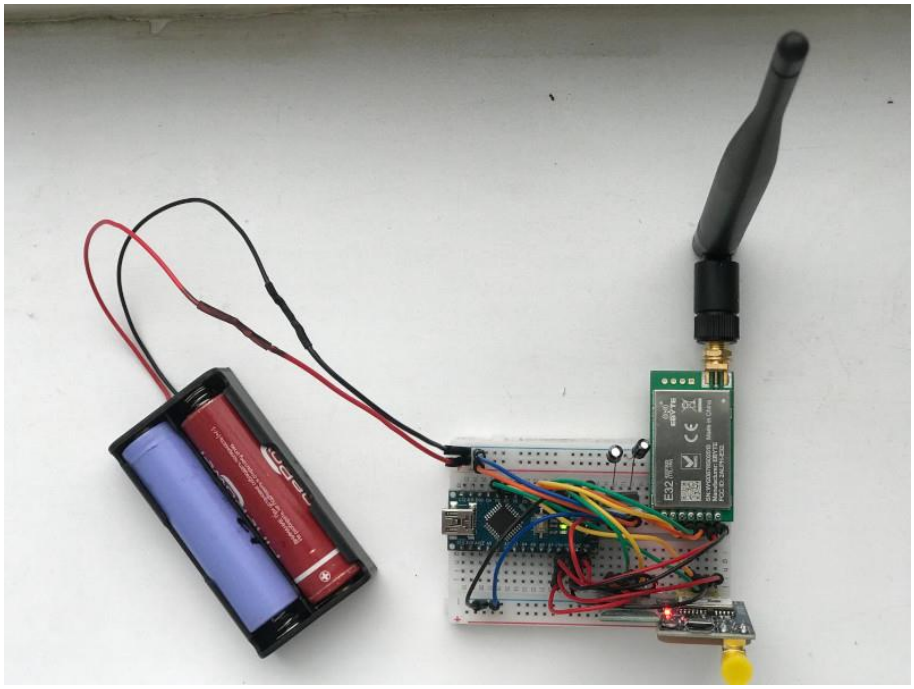


Рисунок 3.8 – Зібрана система

Щоб налаштувати плату Arduino, спочатку потрібно підключити її до комп’ютера за допомогою кабелю USB Mini. Це забезпечить живлення та дозволить передавати дані між комп’ютером і платою. Після підключення операційна система має автоматично розпізнати пристрій і встановити потрібні драйвери.

Далі відкрий програму Arduino IDE. Перед завантаженням коду важливо переконатися, що вибрані правильна модель плати та відповідний COM-порт. Це можна зробити в меню «Інструменти»: там обираються параметри «Плата» та «Порт». Якщо все налаштовано правильно, плата зможе приймати скетч.

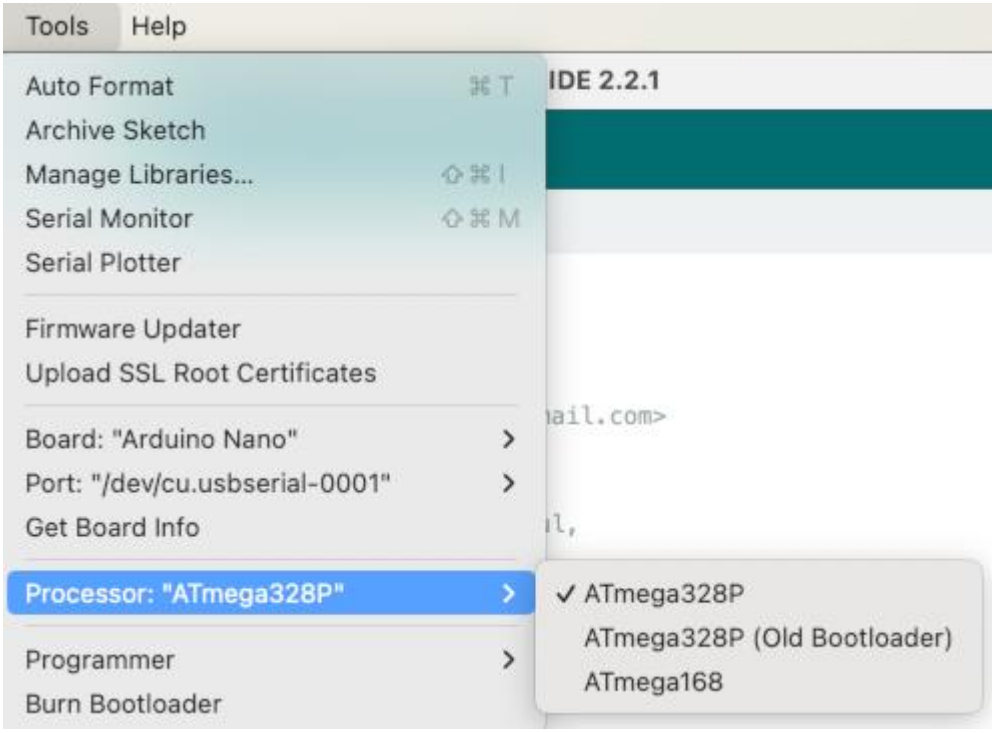


Рисунок 3.9 – Вибір та конфігурація Arduino

{"id":1090218,"key":"2FHB3SRRUYEKG195","t":30.4375,"i":3686}
{"id":1090218,"key":"2FHB3SRRUYEKG195","t":30.5,"i":3687}
{"id":1090218,"key":"2FHB3SRRUYEKG195","t":30.4375,"i":3688}
{"id":1090218,"key":"2FHB3SRRUYEKG195","t":30.4375,"i":3689}
{"id":1090218,"key":"2FHB3SRRUYEKG195","t":30.4375,"i":3690}
{"id":1090218,"key":"2FHB3SRRUYEKG195","t":30.4375,"i":3691}

Рисунок 3.10 – Перевірка відправки та отримання даних із датчика

Для перевірки того, що дані дійсно передаються, можна підключити приймальний пристрій або використовувати спеціальне програмне забезпечення, яке прийматиме та виводитиме отримані дані (рис. 3.10). Якщо все працює правильно, то на комп’ютері у консолі Arduino IDE можна буде побачити датасет який передає датчик погодних умов, які поступово зростають на приймальному боці.

Висновки до розділу 3

У цьому розділі проведено детальний аналіз та вибір технічного апаратного забезпечення для системи далекого радіозв’язку на основі технології LoRa. Розглянуто переваги використання LoRa-модулів, їхні технічні характеристики та особливості роботи в різних умовах. Вибір модуля E32-868T30D обґрунтовано його здатністю забезпечувати стабільний зв'язок на великих відстанях, що є критичним для розробки IoT-рішень. Звернено увагу на законодавчі обмеження щодо частот та потужності передачі в Україні, що важливо враховувати при створенні бездротових мереж.

Також детально розглянуто налаштування модулів за допомогою програмного забезпечення та необхідність коректної конфігурації параметрів, таких як швидкість передачі даних, потужність і адресація. Описано схему підключення LoRa-модуля до мікроконтролера Arduino Nano, а також необхідні кроки для його налаштування та перевірки працездатності. Підготовка коду для передачі даних між пристроями через LoRa-з’єднання забезпечує надійний обмін інформацією та можливість розширення мережі.

Загалом, розроблена система враховує як технічні вимоги, так і законодавчі обмеження, що робить її придатною для практичного використання в умовах великої відстані та низького енергоспоживання.

4 ТЕСТУВАННЯ СИСТЕМИ

У цьому розділі відбувається процес перевірки розробленої системи в реальних умовах експлуатації. У рамках тестування один з датчиків LoRaWAN було встановлено на житловому комплексі «Рів’єра» в місті Миколаїв, що дозволило оцінити ефективність зв’язку та передачі даних через LoRa в умовах міського середовища з різними перешкодами та змінами рельєфу. Тестування допомогло зібрати дані для подальшого вдосконалення системи і підтвердити її працездатність у реальних умовах.

4.1 Тестування у прямій видимості

Для тестування розробленої системи один з датчиків LoRaWAN було встановлено на житловому комплексі «Рів’єра» в місті Миколаїв, на намиві (рис. 4.1). Це дозволило здійснити практичні випробування на реальній місцевості та перевірити ефективність передачі даних через LoRa-з’єднання в умовах міської забудови, що характеризуються наявністю різних перешкод і змінними умовами.

Рисунок 4.1 – Дистанція тесту № 1

Рисунок 4.2 – Приклад місцевості

{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "121 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "122 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "121 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "122 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "123 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "121 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "123 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "121 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "121 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "2,84 km", "ping": "121 ms" }

Рисунок 4.3 – Отриманий пакет з тесту № 1

Аналізуючи результати першого тесту (рис. 4.3), можна зробити кілька висновків. По-перше, усі вимірювання показують стабільну затримку сигналу на рівні 121 мс при фіксованій відстані 2,84 км між точками передачі даних. По-друге, координати точки відправлення залишаються незмінними, що свідчить про проведення тесту в однакових умовах без переміщення обладнання. Також значення ідентифікатора та ключа залишаються сталими, що підтверджує цілісність ідентифікації кожного пакета. Таким чином, можна зробити висновок про стабільну роботу системи на заданій відстані з мінімальною затримкою.

На рис. 4.4 представлено графік, який ілюструє зміну напруги батареї LiPo (літій-полімерної батареї) в процесі роботи двох різних систем. Це дозволяє наочно побачити, як змінюється енергоспоживання кожної з систем протягом часу.

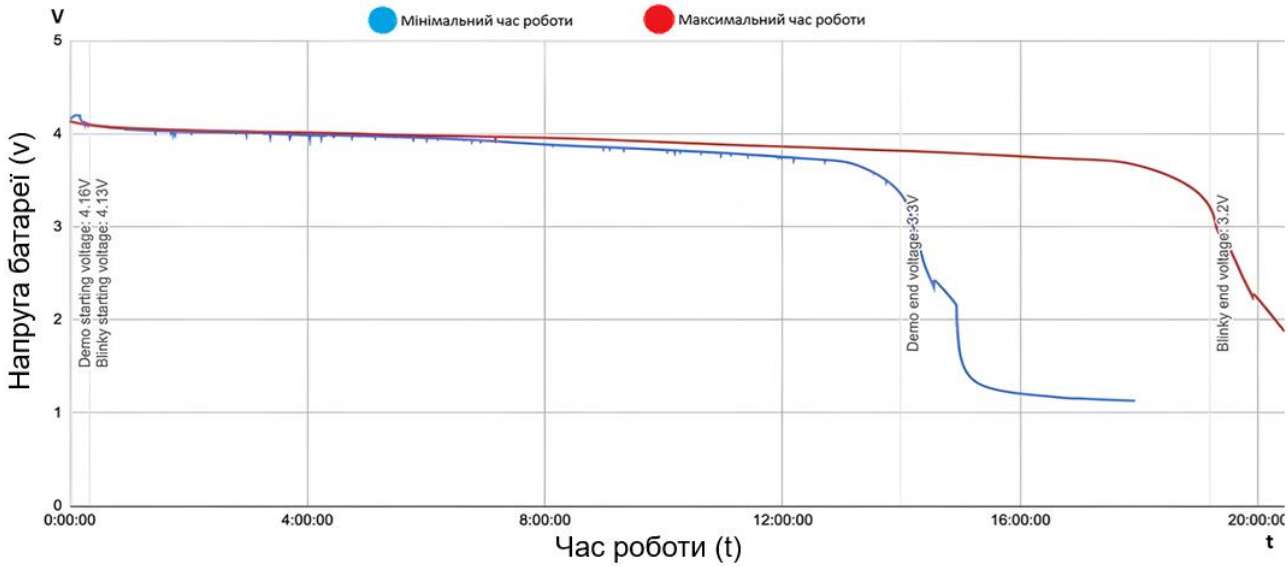


Рисунок 4.4 – Графік зміну напруги батареї (v)

Графік демонструє, як напруга батареї поступово знижується від початкового значення, яке становить близько 4,16 В для системи «Demo» і 4,13 В для системи «Blinky», до кінцевих значень – 3,3 В для «Demo» та 3,2 В для «Blinky». Таке зниження рівня напруги є нормальним явищем для батарей LiPo, оскільки вони витрачають енергію під час роботи і заряд залишкової батареї поступово зменшується.

Графік також показує часову залежність зміни напруги, що дозволяє оцінити тривалість роботи системи, перш ніж рівень напруги досягне критичної позначки. Ці дані є важливими для розуміння того, як довго кожен з пристроїв може працювати без підзарядки. З графіка видно, що в обох випадках час роботи системи майже однаковий, не зважаючи на те що один модуль (передатчик) відправляє пакет завжди, а інший (прийомник) приймає пакети без зупину.

Отримані дані можуть бути використані для подальшого вдосконалення енергоспоживання в пристроях, що працюють від батарей LiPo, а також для оцінки того, як система поводиться протягом тривалого часу використання, що особливо важливо для пристроїв, які використовуються в автономних або віддалених умовах, де необхідно мінімізувати частоту підзарядок.

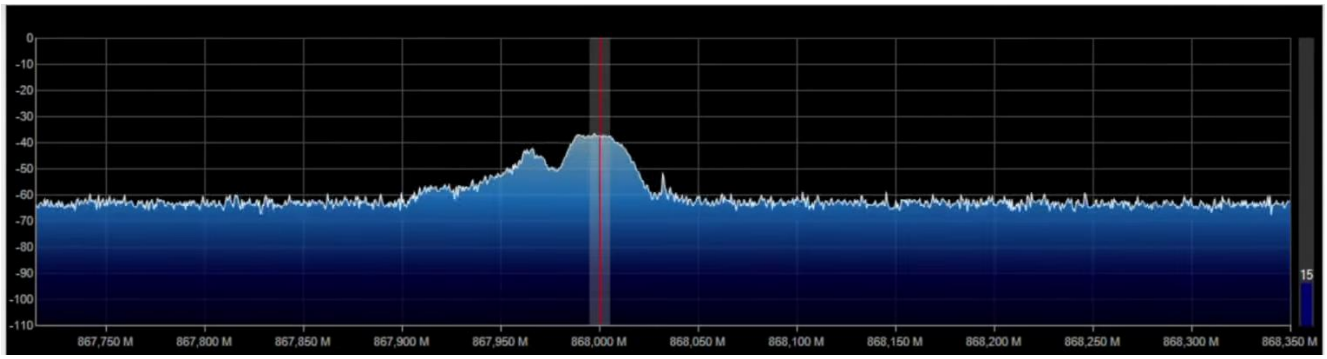


Рисунок 4.5 – Передача пакетів за допомогою LoRa на спектрі (MHz)

Спектр отриманий за допомогою SDR приймача (рис. 4.5): передається 8 байт корисних даних та 8 байт надлишкових даних, оскільки coding rate встановлено на 100 %. Також передаються додаткові «Шапки» від самого радіомодуля та Arduino.

4.2 Тестування у змішаному ландшафті

Наступний тест було проведено в умовах змішаної місцевості, що включала різні типи рельєфу, зокрема рівнини, пагорби та зони з рослинністю, які впливали на умови зв’язку та ефективність роботи системи (рис. 4.6–4.7). Загальна дистанція тестування склала 7,14 км, що дозволило оцінити можливості пристрою в реальних умовах експлуатації, де були присутні як відкриті ділянки, так і місця з перешкодами.

Проведення тесту в таких умовах є важливим для перевірки стабільності передачі даних та продуктивності пристрою під час роботи на різних ділянках маршруту. Це дозволяє зрозуміти, як система справляється з різними викликами, такими як зміна рельєфу, погіршення сигналу в зоні з перешкодами та вплив зовнішніх факторів, включаючи погодні умови та густоту рослинності.

Рисунок 4.6 – Дистанція тесту № 2

Рисунок 4.7 – Місцевість тесту № 2

{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "423 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "422 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "421 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "423 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "422 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "424 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "425 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "421 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "423 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "423 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "421 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "422 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "421 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "422 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "422 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "7,15 km", "ping": "423 ms" }

Рисунок 4.8 – Отриманий пакет з тесту № 2

У результаті цього тесту було встановлено, що при умовній прямій видимості затримка передачі пакета становила 423 мс (рис. 4.8). Це значення значно перевищує результати попереднього тесту, де затримка становила лише 121 мс. Варто зазначити, що попередній тест проводився за більш сприятливих умов: сигнал передавався при прямій видимості на рівні шостого поверху, і в зоні передачі не було жодних перешкод.

Під час нового тестування виникли різні фактори, які вплинули на збільшення затримки. Зокрема, наявність перешкод у вигляді будівель, дерев та інших об'єктів, які екранували сигнал і викликали його відбиття або розсіювання. Крім того, погодні умови та рельєф місцевості також могли зіграти свою роль у зниженні якості зв'язку. Слід також враховувати можливі завади від інших джерел сигналу, що могли спричинити додаткові затримки або втрату пакетів під час передачі.

4.3 Тестування в найгірших умовах на великій відстані

Останнє тестування системи проводилося у надзвичайно складних умовах, які значно впливали на якість передачі інформації. Локацією для цього експерименту було обрано село Володимирівка, розташоване на значній відстані від точки передачі сигналу (рис. 4.9). Основною перешкодою була відсутність прямої видимості – точка передачі знаходилася за лінією горизонту, що створювало додаткові труднощі для забезпечення стабільного зв’язку.

Крім цього, на шляху сигналу були густі дерева, які поглинали та розсіювали радіохвилі, знижуючи ефективність передачі. Сигнал проходив через інше село Шуріне, у якому розташовані дві телекомунікаційні вежі, що також створювали потенційні перешкоди через інтерференцію сигналів. Незважаючи на всі ці фактори, система мала продемонструвати свою здатність підтримувати зв’язок навіть у таких несприятливих умовах, що є важливим критерієм для оцінки її надійності та ефективності.

Рисунок 4.9 – Дистанція тесту № 3

Рисунок 4.10 – Місцевість тесту № 3

За результатами тестування було виявлено, що із збільшенням дистанції затримка передачі даних зростає майже в кілька разів (рис. 4.10). Це свідчить про те, що на великих відстанях система стикається з додатковими викликами, пов'язаними з поширенням сигналу та перешкодами на його шляху. Попри це,

система продемонструвала стабільну роботу, хоча й працювала на межі своїх можливостей, використовуючи максимальну потужність передавача для підтримання зв’язку.

{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8604 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8605 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8603 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8604 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8606 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8602 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8601 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8600 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8605 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8603 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8602 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8605 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8604 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8604 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8603 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8601 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8600 ms" }
{"id":1090218,"key":"2FHB3SRRUYEKG195","FROM": "POINT 1": "46.93197830300337, 31.90429520007093", "distance": "14,76 km", "ping": "8604 ms" }

Рисунок 4.10 – Місцевість тесту № 3

У порівнянні з попередніми тестами, де відстань становила лише 2,84 км та 7,14 км, а затримки — 121 мс та 423 мс відповідно, можна зробити висновок, що система працює значно стабільніше на менших дистанціях. У цьому випадку збільшення відстані майже в 5 разів призвело до зростання затримки майже в 70 разів, що вказує на лінійну залежність між відстанню та погіршенням показників зв’язку.

4.4 Аналіз результатів

Для побудови наступної діаграми були проведені додаткові тести (рис. 4.11).

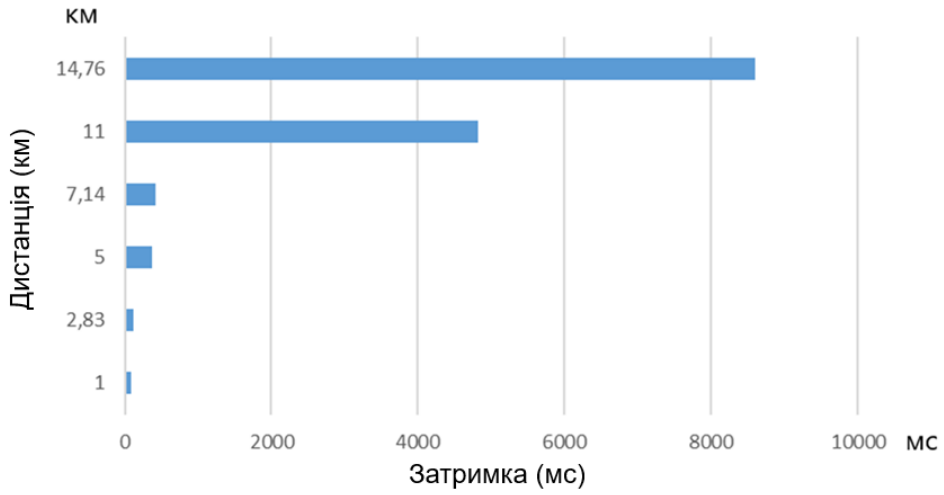


Рисунок 4.11 – Аналіз результатів

На рис. 4.11 наведено графік, що демонструє залежність затримки передачі даних (у мілісекундах) від відстані (у кілометрах) у мережі LoRaWAN. Цей графік ілюструє, як зі збільшенням відстані між точками зв’язку значно зростає час передачі даних.

На коротких відстанях, близько 1 км, затримка майже відсутня, що свідчить про стабільний і швидкий зв'язок. При відстані 2,83 км та 5 км затримка вже помітна, але все ще знаходиться на прийнятному рівні для більшості сценаріїв використання. Ці значення вказують на те, що мережа працює ефективно на середніх відстанях без значних втрат у швидкості передачі даних.

Однак із досягненням відстані 7,14 км затримка стає більш відчутною. Вона свідчить про те, що якість зв’язку починає погіршуватися. На відстані 11 км ситуація суттєво змінюється – затримка значно збільшується, досягаючи декількох тисяч мілісекунд. Нарешті, при максимальній відстані в 14,76 км затримка сягає свого піку, що робить передачу даних дуже повільною.

Це показує, що хоча LoRaWAN добре справляється з передачею даних на невеликих та середніх відстанях, на великих відстанях виникають значні труднощі із затримками. Це може бути важливим фактором при плануванні мережі, де необхідно забезпечити баланс між дальністю зв’язку та якістю передачі даних.

Рисунок 4.12 – Відсоток отриманих і пропущених пакетів

На діаграмі наочно показано співвідношення між отриманими та втраченими пакетами даних залежно від відстані між вузлами передавача (рис. 4.12). Аналізуючи графік, можна побачити, що на коротких і середніх дистанціях передача даних відбувається без втрат – усі пакети успішно досягають приймача. Це свідчить про стабільність і надійність зв’язку на відстанях до 7 км, де мережа працює ефективно та без збоїв.

Однак, коли відстань збільшується до 11 км, починають з’являтися незначні втрати пакетів – близько 10 %. Це можна пояснити погіршенням сигналу на більш віддалених ділянках мережі, де вплив зовнішніх факторів стає суттєвим. При подальшому збільшенні відстані до 14 км втрата пакетів зростає до 17 %. Такий рівень втрат свідчить про поступове зниження якості зв’язку, хоча він залишається прийнятним для ряду сценаріїв використання LoRaWAN. Втрату пакетів можна зменшити за рахунок збільшення ємність акумулятора та потужність модуля.

Висновки до розділу 4

У цьому розділі було проведено ретельне тестування системи передачі даних LoRaWAN у різних умовах. Перші випробування показали стабільний зв’язок на відстанях до 2,84 км із мінімальною затримкою, що свідчить про надійність і ефективність системи в оптимальних умовах. Під час тестування системи на відстані 7,14 км у змішаній місцевості, де сигнал зустрічав природні перешкоди, затримка зросла до 423 мс. Це очікувано, враховуючи вплив дерев, пагорбів та інших факторів. Останнє випробування проведено у найскладніших умовах – зв’язок здійснювався на відстані 14 км без прямої видимості. Попри значне збільшення затримки до кількох секунд, система змогла підтримувати передачу даних. Цей результат підтверджує залежність якості зв’язку від відстані та наявності фізичних перешкод.

Тести показали, що система працює стабільно й відповідає технічним вимогам, але для покращення її роботи на великих відстанях доцільно збільшити потужність передавача та використовувати більш ємні акумулятори. Загалом, система підтвердила свою ефективність і готовність до використання в реальних умовах.

ВИСНОВКИ

В результаті виконання кваліфікаційної магістерської роботи розроблено систему далекого радіозв’язку для контролю датчиків на основі мікрокомп’ютерів з використанням технології LoRa. Для досягнення поставленої мети виконано наступні завдання:

1) проведено аналіз існуючих технологій та протоколів бездротового зв'язку;

2) досліджено можливості одноплатних мікрокомп'ютерів у контексті побудови мережі для моніторингу датчиків;

3) розроблено апаратну та програмну архітектуру системи контролю датчиків з використанням LoRa;

4) реалізовано прототип системи та проведено його тестування для оцінки стабільності передачі даних та енергоспоживання.

Проведені етапи аналізу, проєктування, реалізації та тестування дозволили створити ефективну систему, здатну забезпечити стабільну передачу даних на великі відстані з мінімальним енергоспоживанням. Завдяки використанню LoRa, вдалося досягти високої надійності зв’язку навіть у складних умовах, що підтверджено результатами тестувань у різних середовищах.

Система демонструє високу адаптивність до змін мережевої топології та здатність підтримувати передачу даних із мінімальними втратами, що робить її перспективною для застосування в різних сферах, включаючи моніторинг довкілля, сільське господарство та розумні міста. Важливою перевагою є її масштабованість і можливість інтеграції з іншими IoT-рішеннями, що забезпечує гнучкість для подальшого розвитку та оптимізації.

Таким чином, виконана робота сприяє розвитку сучасних інформаційних технологій та пропонує ефективний інструмент для впровадження IoT-рішень із низьким енергоспоживанням і стабільною роботою на великих відстанях.

Системи на основі LoRa мають значний потенціал розвитку, враховуючи зростаючий попит на енергоефективні та далекобійні рішення для IoT. Одним із головних напрямків удосконалення є підвищення надійності передачі даних на великих відстанях шляхом оптимізації алгоритмів маршрутизації та впровадження ретрансляторів, що дозволить забезпечити стабільний зв’язок навіть у складних умовах із значними перешкодами.

Також перспективним є інтегрування системи з іншими бездротовими технологіями, такими як NB-IoT або 5G, для створення гібридних мереж, які зможуть одночасно підтримувати як далекобійний зв’язок, так і високу швидкість передачі даних. Розширення функціональних можливостей системи, зокрема впровадження додаткових датчиків та аналітичних модулів, дозволить використовувати її у сферах моніторингу навколишнього середовища, сільського господарства, транспорту та розумних міст.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. LoRa та LoRaWAN: революція в під’єднанні до Інтернету речей - Jooby. URL: https://jooby.eu/uk/blog/lora-ta-lorawan-revolyuciya-v-pidyednanni-do-intetu/ (дата звернення: 12.12.2024).

2. Технологія LoRaWAN: опис, огляд трьох класів кінцевих пристроїв, швидкість передачі даних, безпека. URL: https://deps.ua/ua/knowegable-base/reference-information/66634.html (дата звернення: 15.12.2024).

3. LoRa та LoRaWAN: перспективи використання у системах моніторингу на основі одноплатних мікрокомп'ютерів. URL: https://iotdesign.com.ua/articles/lora-lorawan-systemy-monitoringu (дата звернення: 12.10.2024).

4. Системи моніторингу та управління з використанням протоколу LoRa. URL: https://telematics.ua/lora-monitoring-systems (дата звернення: 14.10.2024).

5. Aref M., Banihashem S., Lorenz P. IoT-Based Systems Using LoRa for Long-Distance Communication: A Comprehensive Review. *Sensors*. 2022. Vol. 22, No. 11:3874. DOI: 10.3390/s22113874.

6. LoRaWAN і системи контролю: приклади використання у сільському господарстві. URL: https://www.senscomm.com.ua/blog/lora-agriculture (дата звернення: 15.10.2024).

7. Огляд технології LoraWan. Мережа LoraWan що це? URL: https://www.atiko.com.ua/articles-ua/obzor-tekhnologii-lorawan-ua/ (дата звернення: 12.12.2024).

8. Lauridsen M., Sørensen M., Khan T. LoRaWAN для розумних міст та сільського господарства: аналіз ефективності та виклики. IoT Journal. (2023). DOI: 10.1109/JIOT.2023.00235.

9. Мікрокомп'ютери з підтримкою LoRa для віддаленого моніторингу. URL: https://embeddedcomputing.com/microsystems/lora-microcomputers (дата звернення: 13.10.2023).

10. Costa J. P., Pires J. M., Rosário A. LoRa Networks for Environmental Sensing: Challenges and Opportunities. *International Journal of Distributed Sensor Networks*. (2021). Vol. 17, No. 9. DOI: 10.1177/15501477211056395.

11. Актуальні розробки систем на базі LoRa для моніторингу середовища. URL: https://openhardware.com.ua/lora-systems (дата звернення: 12.10.2024).

12. Dragicevic M., Krizanic S. Low-Power Long-Range Communication with LoRa: Performance in Urban and Rural Areas. International Journal of Wireless Information Networks. (2023). DOI: 10.1007/s10776-023-00513-0.

13. Архітектура систем LoRaWAN: від одноплатних комп'ютерів до промислових рішень. URL: https://lora-solutions.com.ua/architecture (дата звернення: 14.10.2024).

14. Варанкін Д. В. Обухова К. О. Розподілена система детектування рухомих об’єктів та обміну інформацією на базі LoRaWAN. *Ольвійський форум-2024: стратегії країн причорноморського регіону в геополітичному просторі*. Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2024. 144-146 с.

15. LoRa модуль бездротового зв'язку (868MHz-915MHz). URL: https://prom.ua/ua/p1833421967-e220-900t22d-lora.html?srsltid=AfmBOoroeJVPy81qF33WODOQ4kuPmy9lGFOceyLYZjg4nKCEllGWvWWkUak

16. LoRa Antenna Indoor Omni Rubber Aerial. URL: https://www.aliexpress.com/item/32811151718.html?src=google

17. USB перетворювач TTL. URL:https://www.stall.com.ua/ru/product/18822/?srsltid=AfmBOoppjWrf8DYuVXDWUX35mWg3BXTBCMIgkFxvqopoUn-jHbudkhw9Exo

18. Основи енергоефективності у бездротових сенсорних мережах / О. Гордієнко, К. Павленко. Електроніка та автоматика. (2021). № 2. С. 19-27.

19. Libelium. LoRaWAN Wireless IoT Communication. URL: https://www.libelium.com/ (дата звернення: 19.10.2024).

20. Dragicevic M., Kershaw S. LoRa Mesh Networking for Extended IoT Coverage. IoT Applications Journal. (2023). Vol. 15, No. 3. DOI: 10.1089/iot.2023.00513.

21. Альтернативні підходи до організації бездротового зв’язку в IoT / П. Горбаченко, Л. Іващенко. Сучасні технології. (2023). № 7. С. 28-41.

22. LoRa Network Architecture and Performance Analysis / M. Davis, J. Clark. Wireless Systems Review. (2022). Vol. 28, No. 8. P. 91-104. DOI: 10.1002/wsr.20220981.

23. Налаштування мереж LoRaWAN для сільськогосподарських застосувань / М. Кравченко, Ю. Іванов. Сучасні інформаційні системи. (2020). № 4. С. 42-54.

24. Dragicevic M., Krizanic S. Low-Power Long-Range Communication with LoRa: Performance in Urban and Rural Areas. International Journal of Wireless Information Networks. (2023). DOI: 10.1007/s10776-023-00513-0.

25. Centelles R. P. Назустріч сітчастим мережам LoRa для IoT: Ph.D. дис. каф. комп. архіт. Політехнічний університет Каталонії, Барселона, Іспанія, листопад 2021 р.

ДОДАТОК А

Скетч для керування контролерами системи

```
Serial.println("runaway");#include <Stream.h>

#if ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

/*
create the transciever object
*/

EBYTE::EBYTE(Stream *s, uint8_t PIN_M0, uint8_t PIN_M1, uint8_t PIN_AUX)

{

	_s = s;
	_M0 = PIN_M0;
	_M1 = PIN_M1;
	_AUX = PIN_AUX;


}

/*
Initialize the unit--basicall this reads the modules parameters and stores the parameters
for potential future module programming
*/

bool EBYTE::init(uint8_t Attempts) {

	bool ok = true;

	pinMode(_AUX, INPUT);
	pinMode(_M0, OUTPUT);
	pinMode(_M1, OUTPUT);

	_Attempts = Attempts;

	delay(PIN_RECOVER);

	if (_Attempts < 1){
		_Attempts = 1;
	}
	if (_Attempts > 10){
		_Attempts = 10;
	}

	SetMode(EBYTE_MODE_NORMAL);

	// first get the module data (must be called first for some odd reason
	ok = ReadModelData();

	if (!ok) {
```

```
		return false;
	}
	// now get parameters to put unit defaults into the class variables
	ok = ReadParameters();


	if (!ok) {
		return false;
	}

	return true;
}


/*
Method to indicate availability
*/

bool EBYTE::available() {

	return _s->available();

}

/*
Method to indicate availability
*/

void EBYTE::flush() {

	_s->flush();

}

/*
Method to write a single byte...not sure how useful this really is. If you need to send
more that one byte, put the data into a data structure and send it in a big chunk
*/

void EBYTE::SendByte( uint8_t TheByte) {

	_s->write(TheByte);

}


/*
Method to get a single byte...not sure how useful this really is. If you need to get
more that one byte, put the data into a data structure and send/receive it in a big chunk
*/

uint8_t EBYTE::GetByte() {

	return _s->read();

}


/*
Method to send a chunk of data provided data is in a struct--my personal favorite as you
```

```
need not parse or worry about sprintf() inability to handle floats
TTP: put your structure definition into a .h file and include in both the sender and reciever
sketches
NOTE: of your sender and receiver MCU's are different (Teensy and Arduino) caution on the
data
types each handle ints floats differently
*/

bool EBYTE::SendStruct(const void *TheStructure, uint16_t size_) {


              _buf = _s->write((uint8_t *) TheStructure, size_);

              CompleteTask(1000);

              return (_buf == size_);

}


/*
Method to get a chunk of data provided data is in a struct--my personal favorite as you
need not parse or worry about sprintf() inability to handle floats
TTP: put your structure definition into a .h file and include in both the sender and reciever
sketches
NOTE: of your sender and receiver MCU's are different (Teensy and Arduino) caution on the
data
types each handle ints floats differently
*/


bool EBYTE::GetStruct(const void *TheStructure, uint16_t size_) {

       _buf = _s->readBytes((uint8_t *) TheStructure, size_);

       CompleteTask(1000);

       return (_buf == size_);
}


/*
Utility method to wait until module is doen tranmitting
a timeout is provided to avoid an infinite loop
*/

void EBYTE::CompleteTask(unsigned long timeout) {

       unsigned long t = millis();

       // make darn sure millis() is not about to reach max data type limit and start over
       if (((unsigned long) (t + timeout)) == 0){
              t = 0;
       }

       // if AUX pin was supplied and look for HIGH state
       // note you can omit using AUX if no pins are available, but you will have to use
delay() to let module finish

       // per data sheet control after aux goes high is 2ms so delay for at least that long
```

```
	// some MCU are slow so give 50 ms

	if (_AUX != -1) {

		while (digitalRead(_AUX) == LOW) {
			//Serial.println("waiting for aux");
			delay(2);
			if ((millis() - t) > timeout){
				//Serial.println("aux timeout");
				break;
			}
		}
	}
	else {
		// if you can't use aux pin, use 4K7 pullup with Arduino
		// you may need to adjust this value if transmissions fail
		delay(1000);

	}

	// delay(PIN_RECOVER);
}

/*
method to set the mode (program, normal, etc.)
*/

void EBYTE::SetMode(uint8_t mode) {

	// data sheet claims module needs some extra time after mode setting (2ms)
	// most of my projects uses 10 ms, but 40ms is safer

	delay(PIN_RECOVER);

	if (mode == EBYTE_MODE_NORMAL) {
		digitalWrite(_M0, LOW);
		digitalWrite(_M1, LOW);

	}
	else if (mode == MODE_WAKEUP) {
		digitalWrite(_M0, HIGH);
		digitalWrite(_M1, LOW);

	}
	else if (mode == MODE_POWERDOWN) {
		digitalWrite(_M0, LOW);
		digitalWrite(_M1, HIGH);

	}
	else if (mode == MODE_PROGRAM) {
		digitalWrite(_M0, HIGH);
		digitalWrite(_M1, HIGH);

	}

	// data sheet says 2ms later control is returned, let's give just a bit more time
	// these modules can take time to activate pins
	delay(PIN_RECOVER);

	// clear out any junk
```

```
	// added rev 5
	// i've had some issues where after programming, the returned model is 0, and all settings appear to be corrupt
	// i imagine the issue is due to the internal buffer full of junk, hence clearing
	// Reset() *MAY* work but this seems better.
	ClearBuffer();

	// wait until aux pin goes back low
	CompleteTask(4000);

}


// i've asked EBYTE what's supposed to happen--got an unclear answer
// but my testing indicates it clears buffer
// I use this when needing to restart the EBYTE after programming while data is still streaming in
// to let the unit start reading from a cleared internal buffer

// it does NOT return the ebyte back to factory defaults
// if your unit gets corrupt or you need to restore values, you will have to do brute force
// example for and E44-915
// look at the data sheet for default values
//  Trans.SetAddressH(0);
//  Trans.SetAddressL(0);
//  Trans.SetSpeed(0b00011100);
//  Trans.SetChannel(1);
//  Trans.SetOptions(0b01000100);
//  Trans.SaveParameters(PERMANENT);


void EBYTE::Reset() {


	SetMode(MODE_PROGRAM);

	_s->write(0xC4);
	_s->write(0xC4);
	_s->write(0xC4);

	CompleteTask(4000);

	SetMode(EBYTE_MODE_NORMAL);

}


void EBYTE::SetSpeed(uint8_t val) {
	_Speed = val;
}
void EBYTE::SetOptions(uint8_t val) {
	_Options = val;
}
uint8_t EBYTE::GetSpeed() {
	return _Speed ;
}
uint8_t EBYTE::GetOptions() {
	return _Options;
}
```

```
/*
method to set the high bit of the address
*/


void EBYTE::SetAddressH(uint8_t val) {
	_AddressHigh = val;
}

uint8_t EBYTE::GetAddressH() {
	return _AddressHigh;
}

/*
method to set the lo bit of the address
*/

void EBYTE::SetAddressL(uint8_t val) {
	_AddressLow = val;
}


uint8_t EBYTE::GetAddressL() {
	return _AddressLow;
}


/*
method to set the channel
*/

void EBYTE::SetChannel(uint8_t val) {
	_Channel = val;
}
uint8_t EBYTE::GetChannel() {
	return _Channel;
}


/*
method to set the air data rate
*/

void EBYTE::SetAirDataRate(uint8_t val) {

	_AirDataRate = val;
	BuildSpeedByte();

}

uint8_t EBYTE::GetAirDataRate() {
	return _AirDataRate;
}


/*
method to set the parity bit
*/
```

```
void EBYTE::SetParityBit(uint8_t val) {
	_ParityBit = val;
	BuildSpeedByte();
}
uint8_t EBYTE::GetParityBit( ) {
	return _ParityBit;
}

/*
method to set the options bits
*/

void EBYTE::SetTransmissionMode(uint8_t val) {
	_OptionTrans = val;
	BuildOptionByte();
}
uint8_t EBYTE::GetTransmissionMode( ) {
	return _OptionTrans;
}

void EBYTE::SetPullupMode(uint8_t val) {
	_OptionPullup = val;
	BuildOptionByte();
}
uint8_t EBYTE::GetPullupMode( ) {
	return _OptionPullup;
}

void EBYTE::SetWORTIming(uint8_t val) {
	_OptionWakeup = val;
	BuildOptionByte();
}
uint8_t EBYTE::GetWORTIming() {
	return _OptionWakeup;
}

void EBYTE::SetFECMode(uint8_t val) {
	_OptionFEC = val;
	BuildOptionByte();
}
uint8_t EBYTE::GetFECMode( ) {
	return _OptionFEC;
}

void EBYTE::SetTransmitPower(uint8_t val) {

	_OptionPower = val;
	BuildOptionByte();

}

uint8_t EBYTE::GetTransmitPower() {
	return _OptionPower;
}



/*
method to compute the address based on high and low bits
*/
```

```
void EBYTE::SetAddress(uint16_t Val) {
	_AddressHigh = ((Val & 0xFFFF) >> 8);
	_AddressLow = (Val & 0xFF);
}


/*
method to get the address which is a collection of hi and lo bytes
*/


uint16_t EBYTE::GetAddress() {
	return (_AddressHigh << 8) | (_AddressLow );
}

/*
set the UART baud rate
*/

void EBYTE::SetUARTBaudRate(uint8_t val) {
	_UARTDataRate = val;
	BuildSpeedByte();
}

uint8_t EBYTE::GetUARTBaudRate() {
	return _UARTDataRate;
}

/*
method to build the byte for programming (notice it's a collection of a few variables)
*/
void EBYTE::BuildSpeedByte() {
	_Speed = 0;
	_Speed = ((_ParityBit & 0xFF) << 6) | ((_UARTDataRate & 0xFF) << 3) | (_AirDataRate
& 0xFF);

}


/*
method to build the option byte for programming (notice it's a collection of a few variables)
*/

void EBYTE::BuildOptionByte() {
	_Options = 0;
	_Options = ((_OptionTrans & 0xFF) << 7) | ((_OptionPullup & 0xFF) << 6) |
((_OptionWakeup & 0xFF) << 3) | ((_OptionFEC & 0xFF) << 2) | (_OptionPower&0b11);
}


bool EBYTE::GetAux() {
	return digitalRead(_AUX);
}


/*
method to save parameters to the module
*/
```

```
void EBYTE::SaveParameters(uint8_t val) {

	SetMode(MODE_PROGRAM);
	/*
	ClearBuffer();


	Serial.print("val: ");
	Serial.println(val);

	Serial.print("_AddressHigh: ");
	Serial.println(_AddressHigh);

	Serial.print("_AddressLow: ");
	Serial.println(_AddressLow);

	Serial.print("_Speed: ");
	Serial.println(_Speed);

	Serial.print("_Channel: ");
	Serial.println(_Channel);

	Serial.print("_Options: ");
	Serial.println(_Options);
	*/


	_s->write(val);
	_s->write(_AddressHigh);
	_s->write(_AddressLow);
	_s->write(_Speed);
	_s->write(_Channel);
	_s->write(_Options);

	delay(PIN_RECOVER);

	CompleteTask(4000);

	SetMode(EBYTE_MODE_NORMAL);


}

/*
method to print parameters, this can be called anytime after init(), because init gets
parameters
and any method updates the variables
*/

void EBYTE::PrintParameters() {

	_ParityBit = (_Speed & 0XC0) >> 6;
	_UARTDataRate = (_Speed & 0X38) >> 3;
	_AirDataRate = _Speed & 0X07;

	_OptionTrans = (_Options & 0X80) >> 7;
	_OptionPullup = (_Options & 0X40) >> 6;
	_OptionWakeup = (_Options & 0X38) >> 3;
	_OptionFEC = (_Options & 0X07) >> 2;
	_OptionPower = (_Options & 0X03);
```

```
      Serial.println("----------------------------------------");
      Serial.print(F("Model no.: "));  Serial.println(_Model, HEX);
      Serial.print(F("Version  : "));  Serial.println(_Version, HEX);
      Serial.print(F("Features : "));  Serial.println(_Features, HEX);
      Serial.println(F(" "));
      Serial.print(F("Mode    (HEX/DEC/BIN):    "));        Serial.print(_Save,    HEX);
Serial.print(F("/"));           Serial.print(_Save,      DEC);      Serial.print(F("/"));
Serial.println(_Save, BIN);
      Serial.print(F("AddH   (HEX/DEC/BIN):   "));      Serial.print(_AddressHigh,   HEX);
Serial.print(F("/"));      Serial.print(_AddressHigh,      DEC);      Serial.print(F("/"));
Serial.println(_AddressHigh, BIN);
      Serial.print(F("AddL   (HEX/DEC/BIN):   "));      Serial.print(_AddressLow,   HEX);
Serial.print(F("/"));      Serial.print(_AddressLow,      DEC);      Serial.print(F("/"));
Serial.println(_AddressLow, BIN);
      Serial.print(F("Sped    (HEX/DEC/BIN):    "));        Serial.print(_Speed,    HEX);
Serial.print(F("/"));        Serial.print(_Speed,        DEC);        Serial.print(F("/"));
Serial.println(_Speed, BIN);
      Serial.print(F("Chan    (HEX/DEC/BIN):    "));        Serial.print(_Channel,    HEX);
Serial.print(F("/"));       Serial.print(_Channel,       DEC);       Serial.print(F("/"));
Serial.println(_Channel, BIN);
      Serial.print(F("Optn    (HEX/DEC/BIN):    "));        Serial.print(_Options,    HEX);
Serial.print(F("/"));       Serial.print(_Options,       DEC);       Serial.print(F("/"));
Serial.println(_Options, BIN);
      Serial.print(F("Addr   (HEX/DEC/BIN):   "));      Serial.print(GetAddress(),   HEX);
Serial.print(F("/"));      Serial.print(GetAddress(),      DEC);      Serial.print(F("/"));
Serial.println(GetAddress(), BIN);
      Serial.println(F(" "));
      Serial.print(F("SpeedParityBit  (HEX/DEC/BIN)      : "));   Serial.print(_ParityBit,
HEX);   Serial.print(F("/"));      Serial.print(_ParityBit,   DEC);   Serial.print(F("/"));
Serial.println(_ParityBit, BIN);
      Serial.print(F("SpeedUARTDataRate (HEX/DEC/BIN) : "));  Serial.print(_UARTDataRate,
HEX);  Serial.print(F("/"));    Serial.print(_UARTDataRate,  DEC);  Serial.print(F("/"));
Serial.println(_UARTDataRate, BIN);
      Serial.print(F("SpeedAirDataRate (HEX/DEC/BIN)   : "));  Serial.print(_AirDataRate,
HEX);  Serial.print(F("/"));    Serial.print(_AirDataRate,  DEC);  Serial.print(F("/"));
Serial.println(_AirDataRate, BIN);
      Serial.print(F("OptionTrans (HEX/DEC/BIN)        : "));  Serial.print(_OptionTrans,
HEX);  Serial.print(F("/"));    Serial.print(_OptionTrans,  DEC);  Serial.print(F("/"));
Serial.println(_OptionTrans, BIN);
      Serial.print(F("OptionPullup (HEX/DEC/BIN)      : "));  Serial.print(_OptionPullup,
HEX);  Serial.print(F("/"));    Serial.print(_OptionPullup,  DEC);  Serial.print(F("/"));
Serial.println(_OptionPullup, BIN);
      Serial.print(F("OptionWakeup (HEX/DEC/BIN)      : "));  Serial.print(_OptionWakeup,
HEX);  Serial.print(F("/"));    Serial.print(_OptionWakeup,  DEC);  Serial.print(F("/"));
Serial.println(_OptionWakeup, BIN);
      Serial.print(F("OptionFEC (HEX/DEC/BIN)           : "));  Serial.print(_OptionFEC,
HEX);   Serial.print(F("/"));     Serial.print(_OptionFEC,   DEC);   Serial.print(F("/"));
Serial.println(_OptionFEC, BIN);
      Serial.print(F("OptionPower (HEX/DEC/BIN)        : "));  Serial.print(_OptionPower,
HEX);  Serial.print(F("/"));    Serial.print(_OptionPower,  DEC);  Serial.print(F("/"));
Serial.println(_OptionPower, BIN);

      Serial.println("----------------------------------------");

}

/*
method to read parameters,
*/
```

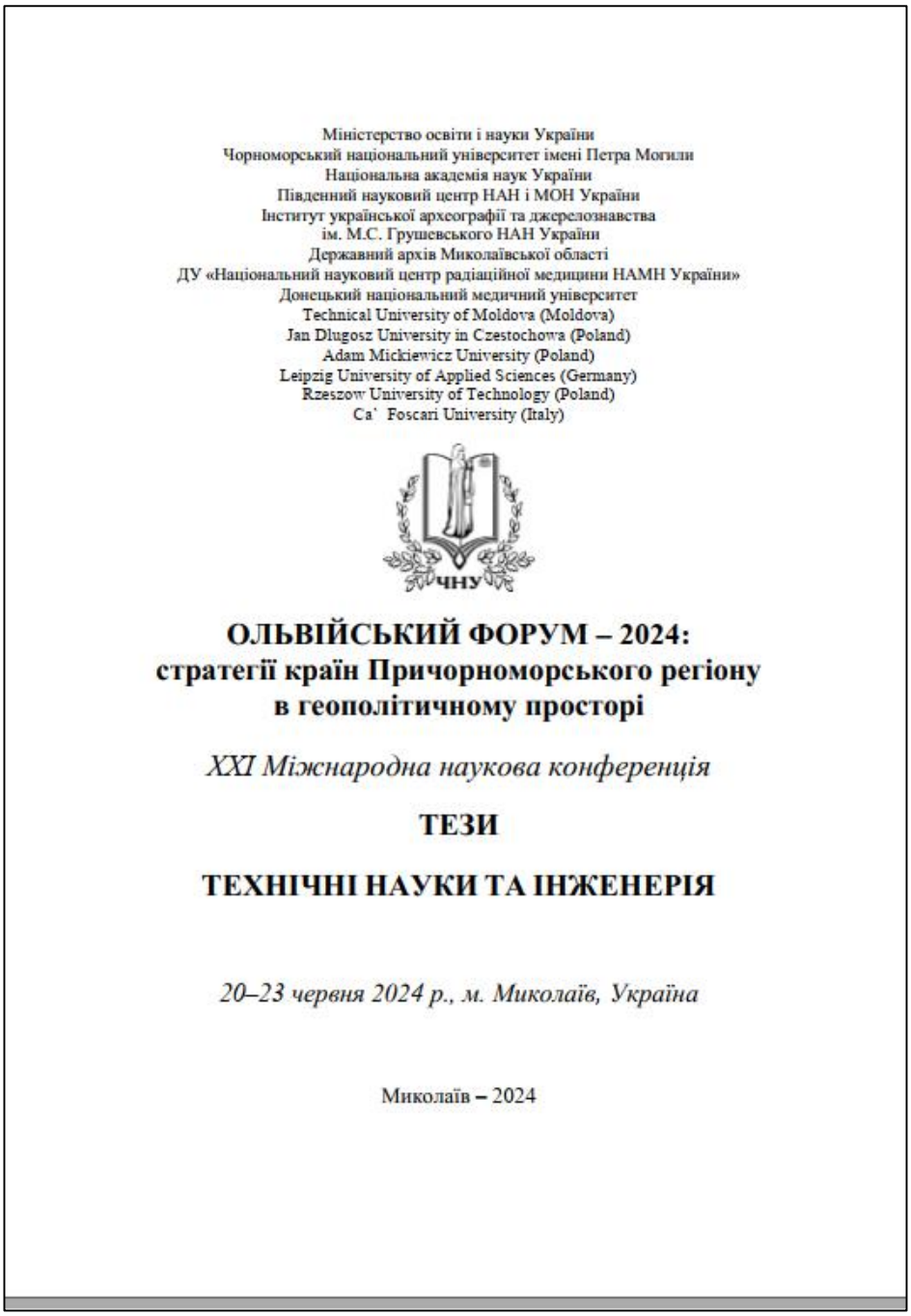
```
bool EBYTE::ReadParameters() {

	_Params[0] = 0;
	_Params[1] = 0;
	_Params[2] = 0;
	_Params[3] = 0;
	_Params[4] = 0;
	_Params[5] = 0;

	SetMode(MODE_PROGRAM);

	_s->write(0xC1);
	_s->write(0xC1);
	_s->write(0xC1);
	_s->readBytes((uint8_t*)&_Params, (uint8_t) sizeof(_Params));

	_Save = _Params[0];
	_AddressHigh = _Params[1];
	_AddressLow = _Params[2];
	_Speed = _Params[3];
	_Channel = _Params[4];
	_Options = _Params[5];

	_Address =  (_AddressHigh << 8) | (_AddressLow);
	_ParityBit = (_Speed & 0XC0) >> 6;
	_UARTDataRate = (_Speed & 0X38) >> 3;
	_AirDataRate = _Speed & 0X07;

	_OptionTrans = (_Options & 0X80) >> 7;
	_OptionPullup = (_Options & 0X40) >> 6;
	_OptionWakeup = (_Options & 0X38) >> 3;
	_OptionFEC = (_Options & 0X07) >> 2;
	_OptionPower = (_Options & 0X03);

	SetMode(EBYTE_MODE_NORMAL);

	if (0xC0 != _Params[0]){
		return false;
	}

	return true;

}


bool EBYTE::ReadModelData() {

	_Params[0] = 0;
	_Params[1] = 0;
	_Params[2] = 0;
	_Params[3] = 0;
	_Params[4] = 0;
	_Params[5] = 0;

	bool found = false;
	int i = 0;

	SetMode(MODE_PROGRAM);
	_s->write(0xC3);
```

```
_s->write(0xC3);
_s->write(0xC3);
_s->readBytes((uint8_t*)& _Params, (uint8_t) sizeof(_Params));
_Save = _Params[0];
_Model = _Params[1];
_Version = _Params[2];
_Features = _Params[3];
SetMode(EBYTE_MODE_NORMAL);

//Serial.print("_Params[0] ");Serial.println(_Save);
//Serial.print("_Params[1] ");Serial.println(_Model);
//Serial.print("_Params[2] ");Serial.println(_Version);
//Serial.print("_Params[3] ");Serial.println(_Features);


if (0xC3 != _Save) {


	// i'm not terribly sure this is the best way to retry
	// may need to set the mode back to normal first....
	for (i = 0; i < _Attempts; i++){
		SetMode(MODE_PROGRAM);
		_Params[0] = 0;
		_Params[1] = 0;
		_Params[2] = 0;
		_Params[3] = 0;
		_Params[4] = 0;
		_Params[5] = 0;

		_s->write(0xC3);
		_s->write(0xC3);
		_s->write(0xC3);

		_s->readBytes((uint8_t*)& _Params, (uint8_t) sizeof(_Params));
		_Save = _Params[0];
		_Model = _Params[1];
		_Version = _Params[2];
		_Features = _Params[3];
		SetMode(EBYTE_MODE_NORMAL);
		//Serial.print("_Attempts ");Serial.println(_Attempts);
		//Serial.print("_Params[0] ");Serial.println(_Params[0]);
		//Serial.print("_Params[1] ");Serial.println(_Params[1]);
		//Serial.print("_Params[2] ");Serial.println(_Params[2]);
		//Serial.print("_Params[3] ");Serial.println(_Params[3]);
		//Serial.print("_Params[4] ");Serial.println(_Params[4]);
		//Serial.print("_Params[5] ");Serial.println(_Params[5]);


		if (0xC3 == _Params[0]){
			found = true;
			break;
		}

		delay(100);

	}
}
else {
	found = true;
}
```

```
	SetMode(EBYTE_MODE_NORMAL);

	return found;

}

/*
method to get module model and E50-TTL-100 will return 50
*/

uint8_t EBYTE::GetModel() {

	return _Model;

}

/*
method to get module version (undocumented as to the value)
*/

uint8_t EBYTE::GetVersion() {

	return _Version;

}

/*
method to get module version (undocumented as to the value)
*/


uint8_t EBYTE::GetFeatures() {

	return _Features;

}

/*
method to clear the serial buffer

without clearing the buffer, i find getting the parameters very unreliable after programming.
i suspect stuff in the buffer affects rogramming
hence, let's clean it out
this is called as part of the setmode

*/
void EBYTE::ClearBuffer(){

	unsigned long amt = millis();

	while(_s->available()) {
		_s->read();
		if ((millis() - amt) > 5000) {

			break;
		}
	}

}
```

ДОДАТОК Б

Апробація роботи



Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
Національна академія наук України
Південний науковий центр НАН і МОН України
Інститут української археографії та джерелознавства
ім. М.С. Грушевського НАН України
Державний архів Миколаївської області
ДУ «Національний науковий центр радіаційної медицини НАМН України»
Донецький національний медичний університет
Technical University of Moldova (Moldova)
Jan Dlugosz University in Czestochowa (Poland)
Adam Mickiewicz University (Poland)
Leipzig University of Applied Sciences (Germany)
Rzeszow University of Technology (Poland)
Ca` Foscari University (Italy)

ЧНУ

**ОЛЬВІЙСЬКИЙ ФОРУМ – 2024:
стратегії країн Причорноморського регіону
в геополітичному просторі**

XXI Міжнародна наукова конференція

ТЕЗИ

ТЕХНІЧНІ НАУКИ ТА ІНЖЕНЕРІЯ

20–23 червня 2024 р., м. Миколаїв, Україна

Миколаїв – 2024

Отже, на даний момент не існує єдиного методу, який задовольняє всі виділені характеристики, проте комбінація методів або компонент з окремих методів може дозволити збільшити наявність позитивних критично важливих характеристик в кінцевому варіанті рекомендованого методу і/чи засобу, а популярність використання технології Kubernetes доводить актуальність проведення глибоких досліджень в цьому напрямку.

Список використаних джерел

1. Predicting resource consumption of Kubernetes container systems using resource models / G. Turin et al. *Journal of Systems and Software*. 2023. P. 111750. DOI: 10.1016/j.jss.2023.111750.
2. Wang X., Zhao K., Qin B. Optimization of task-scheduling strategy in edge Kubernetes clusters based on deep reinforcement learning. *Mathematics*. 2023. Vol. 11, no. 20. P. 4269. DOI: 10.3390/math11204269.
3. ModSoft-HP: Fuzzy Microservices Placement in Kubernetes / E. G. M. Petrakis et al. *Electronics*. 2023. Vol. 13, no. 1. P. 65. DOI: 10.3390/electronics13010065.
4. Chouliaras S., Sotiriadis S. Auto-scaling containerized cloud applications: A workload-driven approach. *Simulation Modelling Practice and Theory*. 2022. P. 102654. DOI: 10.1016/j.simpat.2022.102654.

УДК 004.735

Варанкін Д. В.,
магістрант кафедри комп’ютерної інженерії,
Обухова К. О.,
ст. викладач кафедри комп’ютерної інженерії,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

СИСТЕМА РАДІОЗВ'ЯЗКУ ДАЛЬНЬОГО РАДІУСУ НА ОСНОВІ LORAWAN ДЛЯ КОНТРОЛЮ ДАТЧИКІВ

На сьогоднішній день лише 53% земного шару мають доступ до мережі інтернет, а отже залишається відкритим питання збору даних на великій території, котра ще не охоплена інтернетом. Тому виникає завдання передачі даних на відстані, що обчислюються кілометрами, без використання Wi-Fi та мережі Ethernet.

Для вирішення даної проблеми можна використовувати радіозв’язок за технологією LoRa (Long Range). Залежно від вихідної потужності

ПІДСЕКЦІЯ: Комп'ютерна інженерія

передавача, типу антени, робочої частоти, наявності прямої видимості чи перешкод для проходження радіохвиль (будинки, ліс, перешкоди із боку інших джерел радіовипромінювання та ін.) дальність може становити від сотень метрів до десятків кілометрів. Схема роботи мережі наведена на рис 1.

Рис.1. Схема роботи мережі LoRaWAN

Як і будь-яка інша технологія, LoRa має свої плюси та мінуси, що визначають можливість використання для конкретного завдання.

До сильних сторін технології LoRa відносяться:

1) висока дальність передачі, порівняно з іншими бездротовими технологіями;
2) висока проникаюча здатність у міській забудові;
3) швидкість та простота розгортання мережі. Топологія «зірка» дозволяє швидко покривати великий радіус однією базовою станцією (шлюзом), не використовуючи додаткового обладнання;
4) тривалий термін служби акумулятора. Залежно від класу пристрою і заданих параметрів виходу в мережу LoRaWAN, термін служби елемента живлення може досягати 10 років;
5) простота масштабування;
6) невисока вартість базових станцій та кінцевих пристроїв порівняно зі слабострумними системами.

Як у будь-якої іншої системи, у LoRaWAN є й недоліки:

1) затримка передачі даних від кінцевих пристроїв до програми: від кількох секунд до кількох десятків секунд;

2) невисока пропускна здатність: від кількох сотень біт/с за кілька десятків кбіт/с.

На жаль, швидкість передачі даних LoRa невелика, близько 2,4–19,2 кбіт/с. Однак, цього достатньо, наприклад, для телеметрійних систем і віддаленого контролю, систем розумного будинку або інших подібних систем.

Список використаних джерел

1. LoRaWAN® transforms businesses by connecting wireless IoT sensors simply and affordably. URL: https://lora-alliance.org (Last accessed: 30.04.2024).
2. Технологія LoRaWAN. URL: https://tromsat.ua/news/company/lorawan-technology/ (Last accessed: 30.04.2024).
3. Яка дальність передачі даних у мережі LoRaWAN в міських умовах URL: https://www.atiko.com.ua/articles-ua/yaka-dalnist-peredachi-danyh-u-merezhi-lorawan-v-miskyh-umovah/ (дата звернення: 30.04.2024).

УДК 004.67

Гончаров Д. С.,
аспірант, викладач кафедри комп’ютерної інженерії,
ЧНУ імені Петра Могили, м. Миколаїв, Україна

ЕМПІРИЧНА ОЦІНКА ЯКОСТІ ВИМІРЮВАНЬ ДАТЧИКА ПУЛЬСУ ШЛЯХОМ СТАТИСТИЧНОЇ ОБРОБКИ ДАНИХ

Захворювання є невід’ємною частиною життя людини. З кожним роком зростає кількість людей, котрі прагнуть слідкувати за станом здоров’я не тільки під час тренувань або хвороби, а і в повсякденному житті. Одним із найпоширеніших показників, які відслідковують люди, є серцевий ритм.

За допомогою вимірювань серцевого ритму можна своєчасно виявити такі хвороби, як ішемічна хвороба серця, гіпертонія, вроджені вади серця, клапанні захворювання серця, кардіоміопатія тощо.

Для вимірювань серцевого ритму використовують такі датчики, як MAX30102, Pulse Sensor, AD8232. Клас точності AD8232 залежить від діапазону вхідної напруги, кількості усереднень та температури:

- діапазон вхідної напруги: ±2,6 В; ±5,12 В; ±10,24 В;