Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка та дослідження методу масштабування безпроводової сенсорної мережі оптико-електронних станцій із застосуванням технології LoRa
(тема)

Виконав:
студент 2 курсу, групи ІКІм 22-1
Шлома О.К.
(прізвище, ініціали)
Спеціальність: 172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми: освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма: Інфокомунікаційна інженерія
(повна назва освітньої програми)
Керівник: професор кафедри ІКІ ім. В.В. Поповського
Шостко І.С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ Лемешко О.В.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії імені В.В. Поповського
(повна назва)
Рівень вищої освіти другий (магістерський)
Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)
Тип програми освітньо-професійна
(повна назва)
Освітня програма Інфокомунікаційна інженерія
(повна назва)

ЗАТВЕРДЖУЮ
Зав. кафедри _____
(підпис)
«____» _____2024р

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Шлома Олександру Костянтиновичу .
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка та дослідження методу масштабування безпроводової сенсорної мережі оптико-електронних станцій із застосуванням технології LoRa.
затверджена наказом по університету від «19» жовтня 2023 р. №1212 Ст.
2. Термін подання студентом роботи до екзаменаційної комісії 15.12.2023 р.
3. Вихідні дані до роботи: стандарти та технології безпроводових мереж для порівняння, LoRa WAN на основі IEEE 802.15.4g, топографічні моделі OpenStreetMap, технічна характеристика модулів безпроводового зв'язку з чипом Semtech SX1262, алгоритми управління безпроводовою сенсорною мережею LoRa.
4. Перелік питань, що потрібно опрацювати в роботі:
1) Провести аналіз відомих рішень побудови безпроводових сенсорних мереж, та запропонувати топологію сенсорної мережі LoRa з урахуванням особливостей роботи оптико-електронних станцій.
2) Провести аналіз технічних можливостей та особливостей модулів зв'язку з чипом Semtech SX1262, та врахувати це при побудові мережі LoRa.
3) Провести аналіз можливостей створення окремих кластерів у мережі LoRa з можливістю їх підключення чи відключення залежно від необхідності покриття потрібної території (масштабування мережі).
4) Розробка програмного забезпечення для організації безпроводової сенсорної

мережі оптико-електронних станції та розрахунку маршрутів із застосування технології LoRa

5. Перелік графічного матеріалу із зазначенням креслень, плакатів, комп'ютерних ілюстрацій: Демонстраційний матеріал у вигляді ppt-презентації: результати аналізу сучасних безпроводових сенсорних мереж; метод масштабування безпроводової сенсорної мережі оптико-електронних станцій із застосуванням технології LoRa; алгоритми управління безпроводовою сенсорною мережею оптико-електронних станцій із урахуванням особливостей технології LoRa; програмне забезпечення у вигляді веб-застосунку.

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		(підпис)	(дата)
Основна частина	професор Шостко Ігор Світославович		31.12.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	19.10.2023	Виконано
2	Збір матеріалів для дослідження	30.10.2023	Виконано
3	Розробка 1 розділу	15.11.2023	Виконано
4	Розробка 2 розділу	30.11.2023	Виконано
5	Розробка 3 розділу	21.12.2023	Виконано
6	Розробка 4 розділу	28.12.2023	Виконано
7	Оформлення кваліфікаційної роботи	31.12.2023	Виконано

Дата видачі завдання 19 жовтня 2023 року

Студент _____ Шлома О.К.
(підпис) (прізвище, ініціали)

Керівник роботи _____ професор Шостко І.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 77 сторінок, 59 рисунків, 4 таблиці, 15 джерел за переліком посилань, 1 додаток.

БЕЗПРОВОДОВА СЕНСОРНА МЕРЕЖА, LORA, ЗАВАДОСТІЙКІСТЬ, МАСШТАБУВАННЯ, ПРОГРАМУВАННЯ, ТЕЛЕКОМУНІКАЦІЇ, ОПТИКО-ЕЛЕКТРОНА СТАНЦІЯ, КОДУВАННЯ, АЛГОРИТМ КЛАСТЕРІЗАЦІЇ.

Об'єкт дослідження - процес розроблення безпроводової сенсорної мережі оптико-електронних станцій.

Предмет дослідження - метод масштабування безпроводової сенсорної мережі оптико-електронних станцій зі застосуванням технології LoRa.

Метою кваліфікаційної роботи є розроблення методу, алгоритмів та програмного забезпечення для масштабування безпроводової сенсорної мережі оптико-електронних станцій із застосуванням технології LoRa.

Методи дослідження – системний аналіз, порівняння, узагальнення та класифікація.

В кваліфікаційній роботі проведено аналіз сучасних технології організації безпроводових сенсорних мереж, виконано порівняння їх за ключовими параметрами. Для виконання завдань кваліфікаційної роботи були обрані модулі зв’язку Semtech SX1262 з підтримкою технології LoRa. Було розроблено алгоритми та програмне забезпечення для масштабування безпроводової сенсорної мережі оптико-електронних станцій. Розроблене програмне забезпечення дозволяє взаємодіяти з інтерактивною картою при проектуванні мережі, а також розраховувати найкоротші маршрути між вузлами мережі і додавати вузли ретранслятори при потребі.

ABSTRACT

The report contains 77 pages, 59 figures, 4 tables, 15 sources, 1 application.

WIRELESS SENSOR NETWORK, LORA, INTERFERENCE RESISTANCE, SCALING, PROGRAMMING, TELECOMMUNICATIONS, OPTIC-ELECTRON STATION, ENCODING, CLUSTERING ALGORITHM.

The object of research is the scaling process of the wireless sensor network of optical-electronic stations.

The subject of research is the method of scaling the wireless sensor network of optical-electronic stations using LoRa technology.

The method of qualification work is the development of method, algorhytms and software for organizing the scaling of a wireless sensor network of optical-electronic stations using LoRa technology.

Research methods - systematic analysis, comparison, generalization and classification.

In the qualification work, an analysis of modern technologies for the organization of wireless sensor networks was carried out, and their comparison was made according to key parameters. Semtech SX1262 communication modules with support for LoRa technology were chosen to perform professional tasks. Algorithms and software were developed for scaling the wireless sensor network of optical-electronic stations. The developed software allows you to interact with the interactive map when designing the network, as well as calculate the shortest routes between network nodes and add relay nodes as needed.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БСМ - Безпроводова сенсорна мережа
ОЕС - Оптико електрона станція
CRC - Cyclic redundancy check
CR - Coding rate
FEC - Forward error correction
FFD - Full Function Device
IEEE - Institute of Electrical and Electronics Engineers
IoT - Internet of Things
LAN - Local Area Network
MAN - Metropolitan Area Network
OSM - Open Street Map
PAN - Personal Area Network
RFD - Reduced Function Device
RSSI - Received Signal Strength Indicator
UART - Universal Asynchronous Receiver
WAN - Wide Area Network

ВСТУП

Зі збільшенням кількості пристроїв неминуче зростає навантаження на мережеві пристрої. Для організації гармонійного та поступового росту мереж, необхідно планувати, шукати і розробляти методи та алгоритм масштабування. Ефективна взаємодія мережевих пристроїв у кластерній топології – це досягнення завдяки сукупності мережевих протоколів, вірно обраних методів та професійно побудованих алгоритмів в поєднанні з програмними застосунками і налаштуваннями мережевих пристроїв.

Актуальність роботи полягає в необхідності масштабувати безпроводову сенсорну мережу оптико-електронних станції. Для виконання задач масштабування необхідно розробити алгоритми, програмне забезпечення та методи, що дозволяли б організувати кластера мережі, а також формувати комірки кластерів.

Метою кваліфікаційної роботи є розробка методу, алгоритмів та програмного забезпечення для масштабування безпроводової сенсорної мережі оптико-електронних станцій із застосуванням технології LoRa.

В першому розділі кваліфікаційної роботи були сформовані вимоги до безпроводової сенсорної мережі ґрунтуючись на функціях оптико-електронних станції. Для обрання кращої технології було проведено порівняльний аналіз, за результатом якого було встановлено що технологію LoRa повністю відповідає поставленим вимогам. Також було досліджено топології організації безпроводових сенсорних мереж.

В другому розділі кваліфікаційної роботи було досліджено технологію LoRa та протокол маршрутизації LoRaWAN більш детально. Було проаналізовано переваги та недоліки модулів зв’язку SemTech SX1262 у вигляді USB-конектора від Waveshare, а також порівняно чипсети версії SX1262 та SX1278. Для мережевого трафіку було сформовано окремі пакети і подготовлено програмне забезпечення для сумісності передачі даних з протоколом LoRaWAN.

В третьому розділі кваліфікаційної роботи було проведено дослідження можливості створення окремих кластерів у мережі. Результатом дослідження є алгоритм формування кластеру та комірок кластеру з наведених мережевих

пристроїв. Було показано схему кінцевих пристроїв та командного вузла, а також можливу архітектуру кластеру у вигляді графу.

В четвертому розділі кваліфікаційної роботи виконана реалізація програмного забезпечення у вигляді веб-застосунку для планування безпроводової сенсорної мережі з оптико-електронних станції. Програмне забезпечення дозволяє взаємодіяти з інтерактивною картою та візуалізувати розташування ОЕС. Також були реалізовані додаткові програмні модулі, що візуалізують топологію у вигляді графу та у автоматичному порядку додають вузли ретранслятори при потребі з обчислення найкоротших маршрутів між вузлами мережі.

1 РОЗГЛЯД ТА АНАЛІЗ ВІДОМИХ МЕТОДІВ ПОБУДОВИ СУЧАСНИХ БЕЗПРОВОДОВИХ СЕНСОРНИХ МЕРЕЖ ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ LORA

У сучасному світі технологій, ера безпроводових комунікацій відкриває нові горизонти у різноманітних галузях життя. Однією з ключових інноваційних областей є безпроводові сенсорні мережі (БСМ), які здатні радикально змінити сприйняття та взаємодію з фізичним світом. Ці мережі складаються з безлічі датчиків, що збирають дані з навколишнього середовища та передають їх через безпроводові канали зв'язку.

Визначальною особливістю безпроводових сенсорних мереж є їх спроможність до самоорганізації, низьке енергоспоживання та висока масштабованість, що робить їх ідеальними для різноманітних застосувань: від екологічного моніторингу до розумних будинків і медицини. Важливість БСМ полягає у зборі та обробці великих обсягів даних, що забезпечує глибокий аналіз і прогнозування різних явищ, від кліматичних змін до патернів споживання енергії.

Розвиток технологій безпроводового зв'язку, таких як ZigBee, LoRa, Sigfox, відкриває нові можливості для БСМ. Ці технології сприяють підвищенню ефективності, дальності передачі даних та гнучкості мережі, а також відіграють ключову роль у забезпеченні її надійності та безпеки. Для побудови безпроводової сенсорної мережі та виконання завдання кваліфікаційної була обрана технології LoRa (Long-Range).

1.1 Аналіз загальних відомостей про безпроводові сенсорні мережі

За період останніх десятиліть спостерігається стрімкий прогрес у сфері технологій, що супроводжується зростанням вимог до систем та мереж зв'язку. Ключовими аспектами, які визначають сучасні телекомунікаційні мережі, є їх відмовостійкість, захищеність та енергоефективність. Ці критерії сформувалися в результаті тривалого періоду еволюції інтернет-інфраструктури та накопиченого досвіду користувачів із різноманітних сфер життя. Однією з передових технологій, що зазнала стрімкого розвитку у наш час, є безпроводові сенсорні мережі. Їх сучасна концепція була розроблена у 2003 році з впровадженням першої ревізії протоколу IEEE 802.15.4. Основною задачею БСМ є забезпечення

обміну інформацією між численними пристроями, задіяними в процесах збору та обробки даних. БСМ представляє собою широку мережу сенсорних вузлів, які розміщуються в просторі та спілкуються між собою, забезпечуючи високу енергоефективність за рахунок невеликої швидкості передачі даних. Незважаючи на деякі обмеження, такі мережі здатні охоплювати відстані від декількох метрів до десятків кілометрів. Для комунікацій між модулями використовуються технології, такі як LoRa, SigFox, ZigBee, кожна з яких має свої специфічні переваги та недоліки.

Кожен вузол в БСМ виконує функції збору даних, обробки інформації та безпроводової комунікації. Вузли можуть бути оснащені датчиками для моніторингу різних параметрів, таких як температура, вологість, тиск, рівні шуму та інші. Зібрані дані передаються на центральний сервер або оброблюються локально. На рис. 1.1 зображено приклад організації безпроводової сенсорної мережі з використанням командного вузла, та кінцевих пристроїв.

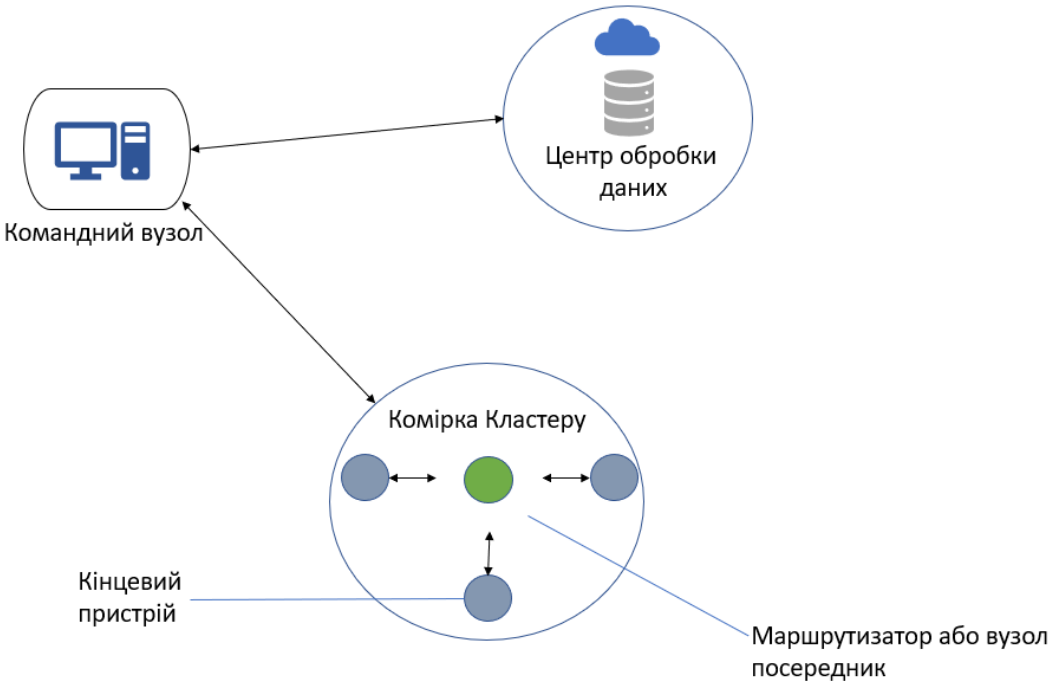


Рисунок 1.1 – Приклад організації безпроводової сенсорної мережі

Серед усіх особливостей безпроводових сенсорних мереж, можна виділити наступні:

Енергоефективність. Однією з ключових особливостей БСМ є енергоефективність, оскільки більшість сенсорних вузлів живляться від

автономних джерел живлення. Це вимагає оптимізації енергоспоживання для подовження терміну служби мережі;

- самоорганізація. Вузли в БСМ здатні самостійно організовувати мережу, вибираючи найоптимальніші маршрути із автоматичними протоколами маршрутизації для передачі даних;

- масштабованість. БСМ можуть масштабуватися від невеликих локальних мереж до великих систем, що охоплюють значні території;

- надійність. Висока ступінь доступності вузлів і маршрутів передачі даних забезпечує високу надійність БСМ.

У контексті передових досліджень у галузі сенсорних мереж, існує можливість розділити ці мережі на основі декількох ключових параметрів, що визначають їх функціональні характеристики та область застосування. Критерії класифікації сенсорних мереж можуть включати, але не обмежуються такими факторами, як масштаб мережі, конкретний об'єкт моніторингу, а також способи та технології передачі даних.

При розгляді масштабу організації мережі, можна виділити чотири основні типи: персональні мережі (PAN), локальні мережі (LAN), міські мережі (MAN) та глобальні мережі (WAN). Кожен із цих типів має унікальні характеристики та потенціал застосування у різноманітних сценаріях. Наприклад, PAN зазвичай використовуються для з'єднання пристроїв у безпосередній близькості до користувача, тоді як WAN можуть охоплювати значно більші географічні регіони, забезпечуючи розгортання масштабних мережевих рішень. Більш детально організацію мереж було досліджено у інших наукових публікаціях [1,2,3]. Графічне представлення класифікації мереж за їх розмірами можна побачити на рис. 1.2.

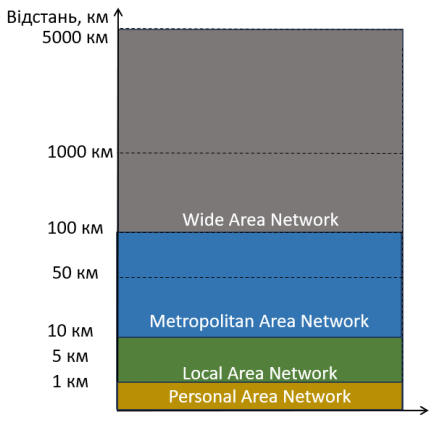


Рисунок 1.2 – Класифікація мереж за розміром

Враховуючи задачі кваліфікаційної роботи, можна зробити висновок, що ефективним рішенням буде побудова локальної мережі (LAN) та її подальше інтегрування в міську мережу (MAN).

Створення декількох окремих LAN є ключовим рішенням для оптимізації внутрішніх процесів. Це дозволить забезпечити надійне, швидке та безпечне з'єднання між ключовими пристроями в межах однієї географічної локації. Локальна мережа сприятиме ефективнішому обміну даними, спільному використанню ресурсів (наприклад, камер, сервоприводів та інших)[4,5].

Наступним кроком для масштабування мережі і покриття більшої площини є інтеграція локальної мережі в міську мережу (MAN), яка відкриє нові можливості. MAN забезпечить високошвидкісний зв'язок між різними локаціями в межах міста, що є важливим для масштабування вже існуючих пристроїв то поєднання у одну суспільну мережу з різних локації. Це також відіграє ключову роль в забезпеченні доступу до централізованих ресурсів, обміну великими обсягами даних та підтримці дистанційної роботи.

Враховуючи це, стає очевидно, що побудова та інтеграція LAN та MAN є стратегічно важливими кроками, що забезпечують технологічну основу для виконання поставлених задач, підвищуючи ефективність та продуктивність на мережевому рівні.

1.2 Топології побудови безпроводових сенсорних мереж

Вибір коректної топології є одним із ключових аспектів, який визначає ефективність, надійність та масштабованість мережі. Безпроводовоі сенсорні мережі стають все більш поширеними в різноманітних галузях застосування. Ці мережі складаються з безлічі сенсорних вузлів, які збирають та передають дані, часто в складних та динамічних умовах.

Важливість вибору топології безпроводової сенсорної мережі полягає в тому, що вона визначає, як саме вузли будуть комунікувати один з одним, яким чином вони будуть передавати дані до центрального вузла або сервера, а також як будуть розподілятися ресурси та енергія всередині мережі. Топологія має безпосередній вплив на параметри мережі, такі як тривалість роботи вузла, пропускна здатність, затримки у передачі даних, а також загальну стійкість мережі до збоїв та відмов.

Серед різних топології безпроводових сенсорних мереж можна виділити точка-точка, зіркову, деревоподібну, сітчасту та інші гібридні структури. Кожна з цих топології має свої переваги та недоліки, залежно від конкретного випадку застосування та вимог до мережі. Тому обрання оптимальної топології вимагає ретельного аналізу специфіки завдання, умов експлуатації та очікуваних результатів використання мережі.

У контексті сучасних досліджень у галузі безпроводовоих сенсорних мереж [6], можна виділити два основних типу мережевої архітектури: самоорганізовані (адаптивні) та централізовано керовані системи. У самоорганізованих БСМ, комунікаційні механізми базуються на принципах децентралізації, де кожен вузол мережі автономно визначає маршрути передачі даних. Це сприяє формуванню гнучких ad-hoc мереж, які не мають статичної або передбачуваної структури. В таких системах, вузли динамічно перенаправляють пакети даних до інших вузлів мережі, реагуючи на зміни у мережевому середовищі. З іншого боку, у централізовано керованих БСМ, маршрутизація та управління мережею здійснюються згідно з попередньо визначеними алгоритмами та структурами. У таких мережах можна розрізняти два основні типи комунікації між вузлами: одноступінчасту (Single-hop) та багатоступінчасту (Multi-hop) передачу даних. У випадку одноступінчастої передачі, дані передаються безпосередньо від вузла до центрального оброблювача або кінцевого пристрою. Натомість, у багатоступінчастій передачі, дані проходять через кілька проміжних сенсорних вузлів, що може підвищувати ефективність та надійність мережі. На рис. 1.3 наведено приклади побудови маршрутів із використанням переприйомів у керованих безпроводових сенсорних мережах, демонструючи різноманітність можливих конфігурацій та підходів до передачі даних.

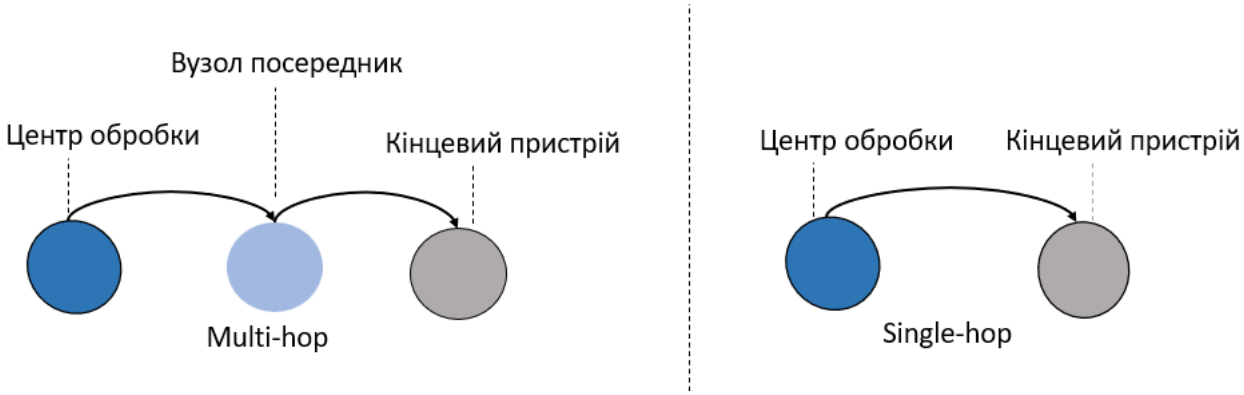


Рисунок 1.3 – Приклад багатоступічанчастої та одноступінчастої маршрутизації

Вибір конкретної топології та стратегії маршрутизації відіграє вирішальну роль у реалізації цілей мережі, враховуючи такі фактори, як обмеження на енергоспоживання, дальність зв'язку та загальні умови експлуатації.

У процесі розробки безпроводових сенсорних мереж зазвичай застосовуються два основних типи пристроїв: Full Function Device (FFD) та Reduced Function Device (RFD). Full Function Device (англ. повнофункціональний пристрій) - володіє здатністю виконувати різноманітні ролі у мережі, включаючи функції контролера, маршрутизатора, та кінцевого пристрою. Це надає FFD пристроям значну гнучкість та можливість встановлювати з'єднання та комунікувати з іншими пристроями, як FFD, так і RFD. В свою чергу Reduced Function Device (англ. пристрій з обмеженим функціоналом) - обмежений лише можливістю виконувати роль кінцевого пристрою в мережі. Слід зазначити, що RFD не мають можливості встановлювати зв'язок з іншими RFD пристроями.

Апаратне забезпечення FFD пристрою є більш складними та потужними, оскільки вони оснащені розширеними обчислювальними можливостями, великими обсягами пам'яті та ефективнішими процесорами. Це, відповідно, впливає на їх вартість, роблячи FFD дорожчими у порівнянні з RFD.

Під час проектування безпроводової сенсорної мережі критично важливо мати принаймні один FFD пристрій, що гарантує ефективне управління та комунікацію в межах мережі. Вибір між FFD та RFD пристроями повинен базуватися на специфічних вимогах та цілях конкретної мережі. Під час розробки безпроводових сенсорних мереж необхідно враховувати специфічні характеристики мережевих пристроїв та умови їх застосування. Мультифункціональність сучасних пристроїв, призначених для створення сенсорних мереж, відкриває широкі можливості для реалізації різноманітних варіантів топології мереж. Завдяки адаптивності налаштувань, існує можливість створення мереж різного типу, зокрема централізованих, децентралізованих та гібридних структур.

1. Топологія типу Peer-to-peer, яка також відома як "точка-точка", представляє собою одну з найбільш елементарних структур у сфері комп'ютерних мереж. Ця топологія характеризується прямим з'єднанням між двома повнофункціональними пристроями (FFD) або між одним обмежено функціональним пристроєм (RFD) та одним повнофункціональним пристроєм (FFD). На рис. 1.4 демонстру-

ється структурна схема топології "точка-точка", де видно взаємодію між двома вузлами мережі, яка полягає в безпосередньому обміні даними.

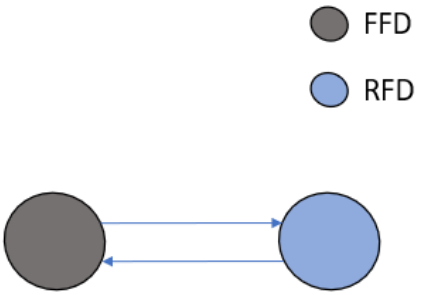


Рисунок 1.4 – Топологія типу «точка-точка»

2. Топологія "зірка" представляє собою модель централізованої мережі, в якій один з пристроїв виконує роль концентратора, що здійснює обробку запитів та пакетів даних від інших вузлів мережі, які не мають прямого з'єднання між собою. У такій структурі концентратор відіграє ключову роль координатора та маршрутизатора. Однією з основних переваг топології "зірка" є відсутність конфліктів між вузлами мережі, оскільки всі з'єднання проходять через центральний вузол. Проте значним недоліком такої структури є те, що при виході з ладу центрального вузла мережа перестає функціонувати. На рис. 1.5 відображено структурну організацію топології "зірка", де ілюструється централізоване з'єднання вузлів із концентратором.

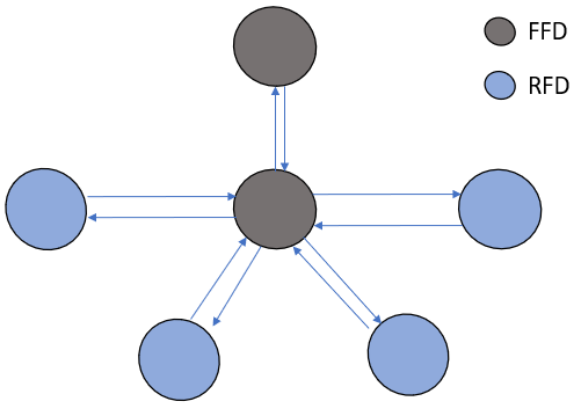


Рисунок 1.5 – Топологія типу «зірка»

3. Топологія "дерево" має структурну подібність до топології "зірка", зокрема в контексті їх спільних переваг та недоліків. Основний принцип структури топології "дерево" полягає в чіткій ієрархічній організації, де кожен

вузол верхнього рівня з'єднаний із вузлами нижчого рівня за допомогою зв'язку, що нагадує форму зірки. Ця ієрархічна структура передбачає, що потужність і завантаженість вузлів зменшуються з верхнього рівня до нижчих. На рис. 1.6 демонструється структура топології "дерево", де видно як центральні вузли зв'язуються із периферійними, утворюючи ієрархічну мережу.

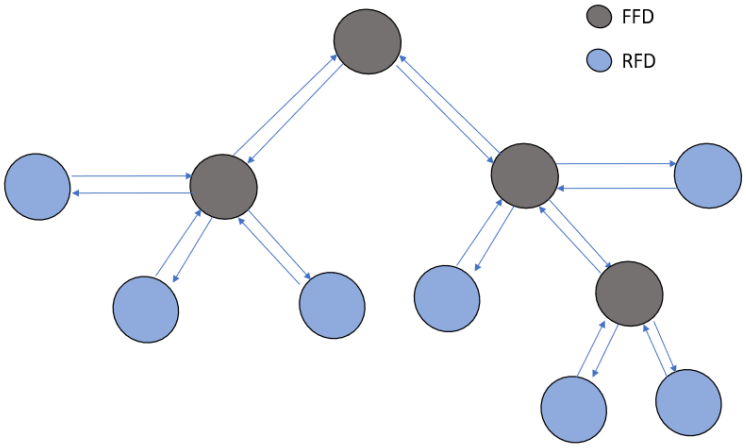


Рисунок 1.6 – Топологія типу «дерево»

4. «Сітчаста» топологія це розвинена топологія типу «дерево» та використовується для побудови децентралізованих мереж. Вона характеризується тим, що кожен вузол мережі виконує функції повноцінного маршрутизатора, активно приймає участь в процесі маршрутизації даних у межах всієї мережі. Такий підхід забезпечує високий рівень надійності та гнучкості, оскільки дані можуть передаватися через різні маршрути, мінімізуючи ризик збоїв у зв'язку. На рис. 1.7 представлена структура "сітчастої" топології, на якому можна побачити взаємозв'язки між різними вузлами, які формують складну мережеву систему.

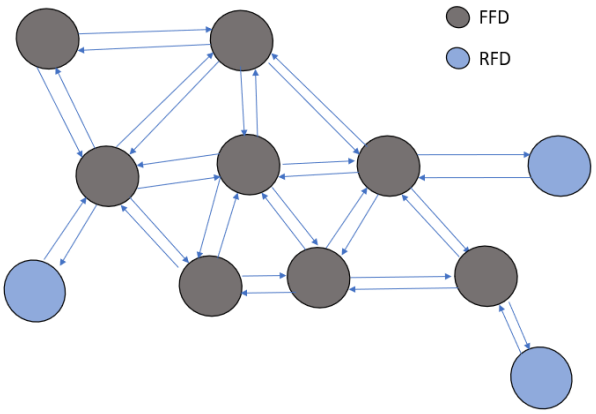


Рисунок 1.7 – «Сітчаста» топологія

Таким чином для побудови мережі з підтримкою технології LoRa, необхідно використовувати топологію типу дерево або сітка. Це дозволить запровадити високо масштабовану та ефективну мережу, забезпечити певний рівень відмовостійкості та захищеності.

1.3 Вимоги та технічні характеристики оптико-електронних станції для організації мережі

Оптико-електронні станції - це високотехнологічні системи, що інтегрують оптичні та електронні компоненти для перетворення, передачі та обробки інформації. Ці станції широко застосовуються в різних сферах, включаючи військові технології. Основним їх завданням є перетворення оптичних сигналів у електронні та навпаки, що дозволяє виконувати точну обробку даних з високою швидкістю та ефективністю.

Ця система розроблена для проведення кругового або секторного моніторингу повітряного простору, здатна автоматично вирішувати широкий спектр завдань, які включають виявлення, розпізнавання, ідентифікацію всіх спостережуваних високоманеврових повітряних об'єктів, а також забезпечує високоточне супроводження обраних повітряних об'єктів. До таких об'єктів відносяться літаки, вертольоти, безпілотні літальні апарати (БПЛА), ракети, дрони, квадрокоптери, артилерійські снаряди, міни тощо. Система також забезпечує відображення і запис точних координат цих об'єктів, а також їхніх візуальних зображень [5]. На рис. 1.8 зображено оптико-електрону станцію, яка використовується як кінцевий пристрій у БСМ.

Рисунок 1.8 – Дослідний зразок оптико-електронної станції

Структура цієї системи базується на архітектурі надійної, витривалої, просторово розподіленої, ешелонованої системи, яка є частиною інтегрованої сенсорної інфокомунікаційної мережі, що складається з малогабаритних оптико-електронних станцій (ОЕС). Ці станції побудовані за модульним принципом, що забезпечує їх високу надійність, ремонтопридатність і ефективність формування потрібних тактико-технічних характеристик, враховуючи останні досягнення у сфері опто-електроніки та інтелектуальної відеоаналітики [5].

Автоматична оптико-електронна система в процесі виявлення та супроводу ціль вирішує такі завдання:

- покращення якості зображення (зменшення видимих шумів на зображенні, покращення контрастність зображення, підвищення чіткості);
- виявлення всіх рухомих об’єктів у відеопотоці;
- пошук рухомого об’єкта за апріорною інформацією про його тип, форму,
- швидкість і характер руху;
- вимірювання геометричних параметрів об’єктів спостереження;
- автоматична ідентифікація типу об’єкта за порівнянням зі стандартом в
- форма, колір, текстура поверхні, швидкість руху;
- формування спеціальних точок супроводжуючого об’єкта;
- точна опора об’єкта на обрані точки його поверхні;
- прогнозування траєкторій супроводжуваних об’єктів;
- повторне захоплення об’єкта у разі відмови його супроводу.

Таким чином, базуючись на основних задачах ОЕС можна сформулювати наступні вимоги до мережі:

- далекобійність комунікації (від 100 метрів до 5 кілометрів);
- використання мережі у складних погодних та географічних умовах;
- захищеність каналу зв’язку від завад та втручання;
- універсальність модулів;
- енергоефективність.

Організована БСМ повинна забезпечувати керування топологією з великою кількістю ОЕС у різноманітних погодних умовах, географічних локаціях та мати певну автономність для забезпечення резервування. Сформована мережа може бути організована як з одного кластера, так і з декількох комірок, що будуть доповнювати топологію мережі. Розроблені алгоритми та програмне забезпечення

повинні надати змогу масштабувати та розбити на кластеру безпроводову сенсорну мережу з повною відповідністю вимогам оптико-електронних станції.

1.4 Порівняння технологій організації безпроводових сенсорних мереж

На поточний момент серед великої кількості технології організації безпроводових сенсорних мереж, можна виділити наступні: LoRa, ZigBee, SigFox. Кожна з цих мереж має власні недоліки і переваги.

LoRa (Long Range) - це запатентована технологія далекобійного та малопотужного зв’язку, що була розроблена компанією Semtech. Разом з технологією був розроблений протоколом LoRaWAN не комерційною організації LoRa Alliance. LoRa з LoRaWAN формують малопотужні безпроводові мережі LPWAN (Low Power Wide Area Network). LoRa є абревіатурою від Long Range (англ. велика відстань). LoRaWAN це протокол рівня управління доступом до середовища. Він використовується поверх фізичного рівня. Згідно з офіційною документацію компанії Semtech відстань передачі даних варується від 1 до 10 кілометрів на відкритий місцевості, і до 5 кілометрів у міській забудові. Комунікація між пристроями виконується у неліцензованому діапазоні частот (433/868/915 МГц), що дозволяє вільно використовувати технологію при побудові IoT мереж. Особливість технологій LoRa полягає у використані методів модуляції, що базуються на методах розширення спектру SMM і лінійній частотній модуляції CSS. Цей метод модуляції дозволяє працювати з сигналами рівня 20 дБ, що є нижче рівня шумів. Технологія LoRa займає фізичний рівень моделі OSI, а LoRaWAN - всі верхні рівні. Швидкість передачі даних варується від 0.17 кбіт/с до 50 кбіт/с в залежності від налаштувань модулів.

ZigBee - технологія безпроводовогоі зв'язку, яка підпорядковується стандарту IEEE 802.15.4. Головною метою її розробки було створення енергоефективних та економічних рішень. Мережі ZigBee характеризуються властивістю самоорганізації, де пристрої автоматично формують мережу та встановлюють зв'язок при їх активації. У мережі ZigBee існують три основні типи пристроїв: координатор (FFD пристрій), який визначає основну топологію мережі та є необхідним для її функціонування; маршрутизатор, який виступає як проміжний вузол і може запускати додаткові додатки; та кінцевий пристрій, призначений для спілкування з пристроями вищих рівнів та передачі даних, але не здатний до з'єднання інших пристроїв у мережу. Для спілкування між пристроями

використовується частота 2.4 ГГц. ZigBee-пристрої взаємодіють із верхніми рівнями моделі OSI, включаючи фізичний рівень та керування доступом до мережевого середовища, мають обмежену дальність дії, приблизно 100 метрів.

SigFox представляє собою технологію, яка використовує промисловий та науковий радіодіапазон ISM для забезпечення зв'язку. Ця система працює на частотах 868 МГц у Європі та 902 МГц у США. Однією з ключових особливостей SigFox є застосування широкосмугового сигналу, що забезпечує комутацію крізь щільні перешкоди. Для маршрутизації даних у системі використовується протокол LPWAN (Low Power Wide Area Network). Топологія мережі, що використовується в SigFox, базується на принципі "зірки". Щодо обмежень передачі даних, пристрої в мережі SigFox мають здатність відправляти до 140 повідомлень на добу, з максимальним обмеженням розміру повідомлення в 12 байт.

Кожна з цих технології будує різні за типом організації та масштабами мережі. На рис. 1.9 наведено розподіл безпроводових технології за далекобійністю та швидкістю передачі даних.

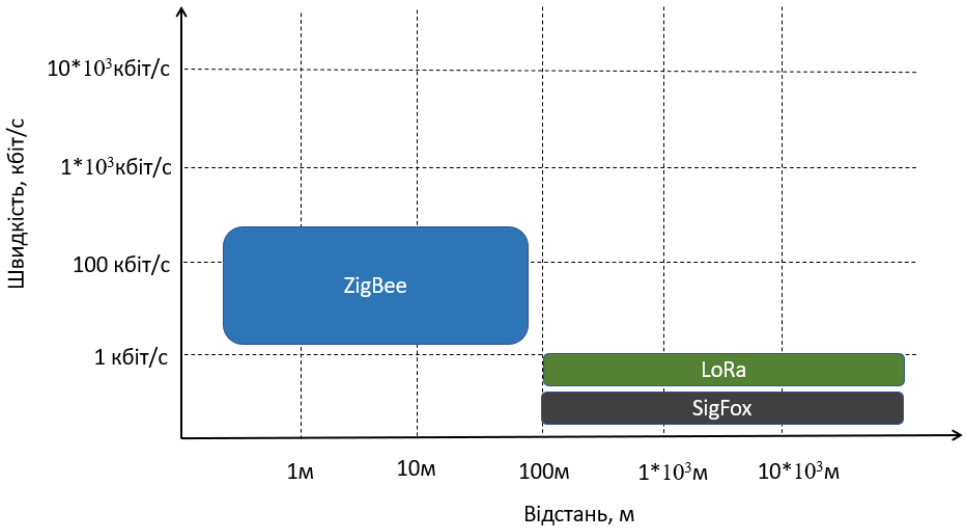


Рисунок 1.9 - Розподіл технології за швидкість та далекобійністю

У табл. 1.1 наведено порівняльний аналіз технології LoRa, ZigBee та SigFox. Їх було порівняно за ключовими характеристиками, які враховуються при організації мережі для супровуду функціюнування ОЕС. Кожна з цих технології будує різні за організацією та масштабами мережі, а також мають абсолютні різні методи модуляції, що потенційне впливає на далекобійність та захищеність

мереж. Для ОЕС важлими параметрами є погодні умови та робота в місті поблизу забудови та інших перешкод, у тому числі вплив інших випромінювачів.

Таблиця 1.1 – Порівняння безпроводових технологій

Назва:	LoRa	ZigBee	SigFox
Стандарт:	власний	IEEE 802.15.4	власний
Частота:	433 МГц 868 МГц 915 МГц	868 МГц 915 МГц 2,4 ГГц	868 МГц 902 МГц
Топологія:	Зірка, сітчаста, точка-точка	Зірка, точка-точка, сітка	Зірка
Тип модуляції:	LoRa	BPSK, QPSK	2GFSK
Швидкість передачі даних, кбіт/с	0.3 – 50 кбіт/с	250 кбіт/с	0.1 кбіт/с
Дальність, м	100м-15000м	10м-100м	100м-15000м
Вплив зовнішніх факторів	Слабий	Середній	Слабий
Можливість модифікації	Присутня	Присутня	Відсутня
Швидкість Б/С	Необмежена	До 10 км/год	До 10 км/год
Наявність екосистеми	Присутня	Присутня	Відсутня

За результатами порівняння і аналізу мереж з таблиці 1.1, можна зробити висновок що технологія LoRa відповідає усім поставленим вимогам. Але для повноцінного функціонування мережі з ОЕС необхідно використовувати протокол комутації аналогічний до LoRaWAN для забезпечення умов для кластеризації, а також підготувати алгоритми керування кластерами, девайсами у комірках кластера, розробити або використати вже існуючи алгоритми черг, реалізувати методи маршрутизації використовуючи наробки з інших мережевих компанії та технологій, як Cisco, Huawei та інші. Реалізація команд керування та опис технічних характеристик модулів зв’язку LoRa Semtech SX1262 наведено у другому розділі кваліфікаційної роботи. Процеси створення кластерів та керування ними наведено у третьому розділі роботи.

2 АНАЛІЗ ТЕХНІЧНИХ МОЖЛИВОСТЕЙ ТА ХАРАКТЕРИСТИК МОДУЛІВ ЗВ'ЯКУ З ЧИПОМ SEMTECH SX1262

2.1 Технічні характеристики та особливості модулів зв'язку з використанням технології LoRa

LoRa це скорочення від англ. Long Range, що перекладається як «Велика дальність». Вона представляє собою патентовану технологію модуляції, що була розроблена компанією Semtech. Ця технологія забезпечує передачу інформації на великі відстані з використанням мінімальної потужності, що робить її ідеально підходящою для реалізації інтернету речей (IoT) та для застосувань, де енергоефективність та дальність дії є критично важливими. Разом з протоколом LoRaWAN, що розроблений не комерційною організації LoRa Alliance формують малопотужні безпроводові мережі LPWAN (Low Power Wide Area Network). Згідно з офіційною документацію [25] відстань передачі даних варується від 1 до 15 кілометрів на відкритому просторі, і до 5 км у щільній міській забудові. Комунікація між пристроями виконується у не ліцензованому діапазоні частот (433/868/900/915 МГц), що дозволяє вільно використовувати технологію при побудові IoT мереж. Особливість технологій LoRa полягає у використані методів модуляції з розширеним спектром (chirp spreadspectrum).

Модуляція LoRa ґрунтується на спектрально-ефективному методі передачі даних, що базується на технології, відомій як "розширення спектру". Основною особливістю цього методу є використання широкого спектрального діапазону для передачі інформації, що призводить до високої стійкості до перешкод та збільшення дальності передачі сигналу.

Спектрально-часове розширення (Chirp Spread Spectrum, CSS) є методом модуляції на фізичному. CSS реалізується шляхом прийняття синусоїдального сигналу та його лінійної зміни у частотному просторі з плином часу, що веде до формування сигналу зі змінною частотою (чирпу). Надалі цей сигнал модулюється на вибрану несучу частоту. На рис. 2.1 наведено приклад модуляції LoRa, що показує приклад розділу сигналу на «чирпи».

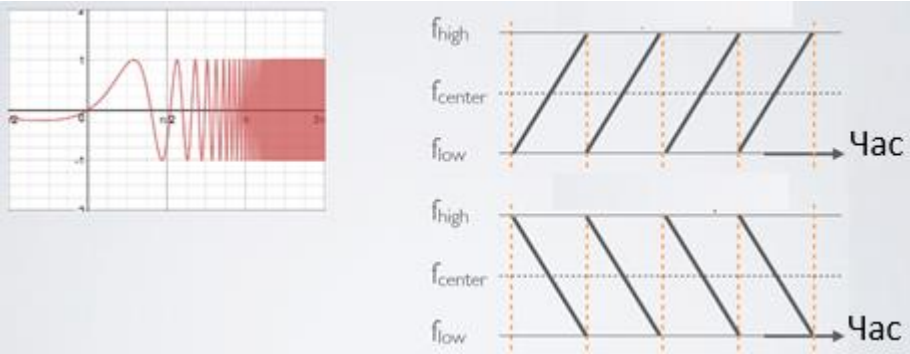


Рисунок 2.1 – Схема розподілу сигналу на «чирпи»[12]

Використання CSS з чирпуванням сигналу в широкій смузі пропускання дозволяє LoRa розподіляти енергію по широкому діапазону частот. Це значно підвищує стійкість системи до перешкод та шуму, а також покращує якість прийому сигналу на шлюзі. На рис. 2.2 наведено сигнал до та після модуляції.

Рисунок 2.2 – Сигнал до та після модуляції LoRa [12]

Доповнення до CSS є адаптивна швидкість передачі даних (Adaptive Data Rate, ADR) в LoRa - це метод, призначений для динамічної оптимізації швидкості передачі даних між кінцевими вузлами та шлюзами. Використовуючи ADR, пристрої можуть автоматично переключатися на вищі швидкості передачі даних у умовах низького рівня перешкод і на більш надійні, але повільніші режими при зростанні інтенсивності перешкод. ADR забезпечує оптимальну швидкість передачі даних, адаптуючись до змінних умов каналу, що максимізує час автономної роботи пристроїв та загальну пропускну спроможність мережі. В сукупності CSS і ADR разом надають технології LoRa значні переваги, зокрема великий радіус дії та високу стійкість до перешкод при мінімальному енергоспоживанні. Також вони сприяють підвищенню безпеки системи, оскільки для зовнішніх систем ці сигнали часто сприймаються як шум.

Серед інших переваг модуляції LoRa в порівнянні з іншими технологіями можна виділити високу чутливість прийому сигналу, що дозволяє передавати дані

на великі відстані при мінімальному споживанні енергії. Крім того, технологія LoRa здатна ефективно працювати в умовах сильних радіоперешкод та приймати сигнали нижче уровню шуму (-20дБм). [7].

LoRaWAN, як архітектурне та протокольне рішення, формулює стандартизований підхід до створення безпроводових сенсорних мереж, базуючись на LoRa - технології радіозв'язку з довгим радіусом дії. Використання цієї технологій разом з протоколом від вендора особливо актуально для розробки систем з тривалим терміном служби батареї та високим рівнем поширення сигналу, що робить її ідеальною для застосувань у військовій сфері. Архітектура LoRaWAN організована за принципом «зірка зірки», де в кожному кластері кінцеві вузли передають дані до центрального вузла, відтак центральний вузол транслює ці дані до серверів через стандартні мережеві протоколи.

Ключовими характеристиками LoRaWAN є здатність до адаптивної передачі даних, висока енергоефективність, застосування шифрування на рівні кінцевих пристроїв та в мережі загалом, а також підтримка великої кількості вузлів у мережі. LoRa визначає фізичний рівень в моделі OSI, в той час як LoRaWAN охоплює верхні рівні цієї моделі. Швидкість передачі даних у LoRaWAN може варіюватися в залежності від коефіцієнта розширення. Як протокол маршрутизації, в LoRaWAN використовується LPWAN (Low Power Wide Area Network).

У мережі LoRaWAN існують три класи кінцевих пристроїв: A, B і C, які розрізняються за режимами прийому та передачі даних, а також за енергоефективністю.

Клас A - це базовий клас, обов'язковий для підтримки всіма пристроями. Пристрої цього класу підключаються до мережі за запланованим розкладом та мають два додаткових часових вікна для прослуховування мережі після надсилання даних. Вони відрізняються тривалим терміном служби, однак для надсилання повідомлень на пристрій необхідно чекати наступного запланованого часу підключення до мережі.

Клас B - відрізняється від класу A наявністю додаткових запланованих періодів прослуховування мережі, які відбуваються за розкладом. Пристрої синхронізуються з базовою станцією через спеціальні маячки, дозволяючи точно визначити час активації пристрою для зв'язку.

Клас C - охоплює кінцеві пристрої, які постійно прослуховують мережу, за винятком періоду передачі даних. Вони застосовуються в ситуаціях, де немає

необхідності економити енергію, але потрібно опитувати пристрій у незапланований час.

Для побудови мережі LoRa для забезпечення зв’язку між пристроями ОЕС а також виконання завдання кваліфікаційної роботи будуть використовуватись FFD пристрої класу С.

У поточній конфігурації безпроводової мережі, призначеної для оптико-електронних станцій, застосовуються модулі LoRa EBYTE E32-433T20DT. Ці пристрої підключаються як модулі послідовного порту для прийому та передачі (UART). В наведеній моделі використовується радіочастотний чіп SX1278, розроблений та вироблений компанією Semtech. Керування мережевим простроєм E32-433T20DT виконується через підключення до зовнішнього мікроконтролера Arduino UNO через асинхронний/синхронний інтерфейс прийому/передачі (USART). Схема підключення наведена на рис. 2.3.

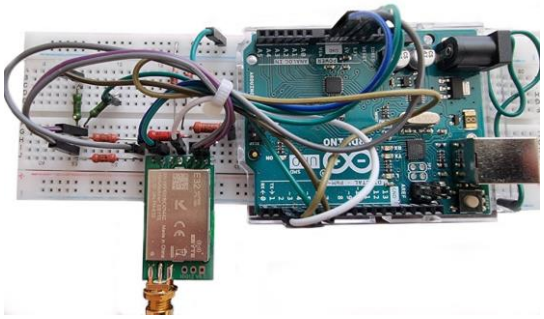


Рисунок 2.3 – Схема підключення мікроконтролера Arduino UNO до модулю E32-433T20DT

Однак слід зазначити що пряме підключення через порт UART додає обмеження стосовно можливості заміни модулів, масштабування мережі та використання сторонніх мікроконтролерів або персональних комп’ютерів без порту UART. Також використання мікроконтролеру разом з модулем зв’язку впливає на автономність мережі при живленні від батареї. Ефективною альтернативою можуть слугувати модулі USB-to-LoRa від компанії Waveshare, що за основу використовує модуль зв’язку SX1262. На рис. 2.4 наведено зображення модуля зв’язку разом з габаритами девайсу.

Рисунок 2.4 – Зображення модулю USB-to-LoRa від Waveshare

На відміну від попередній конфігурацій, використання USB інтерфейсу спрощує інтеграцію з комп'ютерним обладнанням та розширює можливості для інтеграції мережі. Порівняння чипу Semtech SX1262 з чипом SX1278 наведено у таблиці 2.1.

Таблиця 2.1 – Порівняльна таблиця модулів зв’язку Semtech

Параметр	SX1262	SX1278
Діапазон частот (МГц)	150 МГц – 960 МГц	433 МГц, 868 МГц, 915 МГц
Тип модуляції	LoRa, FSK, GFSK, MSK, GMSK	LoRa, FSK, GFSK, MSK, GMSK
Spreading Factor (SF)	7 - 12	6 - 12
Code Rate (CR)	4/5, 4/6, 4/7, 4/8	4/5, 4/6, 4/7, 4/8
Довжина пакету	256 Байт	58 Байт
Буфер	512 байт	512 Байт
Швидкість передачі даних, кбіт/с	0.018 кбіт/с – 19.2 кбіт/с	0.3 кбіт/с – 19.2 кбіт/с
Потужність	+22 дБм	+20 дБм
Чутливість прийому	-148 дБм	-139 дБм
Струм приймача (мА)	4.8 мА	10.3 мА
Струм передавача (мА)	23 – 139 мА	120 мА

Продовження таблиці 2.1

Напруга живлення, В	2.3 В – 5.5 В	2.3 В – 5.5 В
Температурний режим	-40 ~ 85 °C	-40 ~ 85 °C
Інтерфейси	USB / RS232	UART / RS232

В результаті аналізу технічних характеристик модулів зв’язку можна зробити висновок, що модуль Semtech SX1262 має кращу енергоефективність, ширший діапазон частот, який забезпечує більшу гнучкість та адаптивність. Також слід зазначити, що модулі з USB мають покращену чутливість прийому сигналу. Це забезпечить роботу в умовах високих перешкод. Використання модулів SX1262 дозволить масштабувати мережу, підготувати модулі ретранслятори, які в свою чергу дозволять збільшити зону покриття кластеру. Слід зазначити, що дві наведені версії модулів мають сумісність, що дозволить об’єднати модулі в єдину мережу.

2.2 Організація комутаційних процесів та формування пакетів даних

Модулі зв’язку мають два режими комунікації та чотири режими роботи, які конфігуруються за допомогою команд управління модулями, як вказано у офіційній документації до модулів LoRa [13].

Один з режимів комунікації характеризується як широкомовна розсилка. У цьому контексті, адреса пакету визначається як 0x00 0x00 або 0xFF 0xFF, а канал для розсилки, наприклад, встановлюється як 0x04. Це означає, що всі модулі, які налаштовані на прийом на четвертому каналі, будуть приймати пакети. Важливо відзначити, що для модуля-відправника не є обов'язковим перебувати на каналі, який використовується для розсилання пакетів. На рис. 2.5 зображено процес широкомовної розсилки даних.

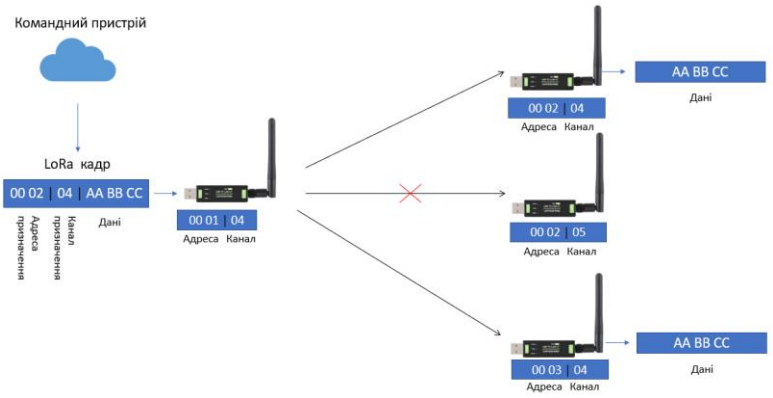


Рисунок 2.5 – Схема широкомовної розсилки

Інший режим роботи - фіксована передача даних, при якому декларується адреса пристрою призначення та канал прийому. Для управління передачею даних використовуються значення, подані у шістнадцятковому форматі. Наприклад, у пакеті може бути вказано адресу 0x0002 та канал 0x04, що означає, що пакет буде прийнятий лише модулем з адресою 0002, налаштованим на четвертий канал. Інші модулі, що не відповідають цим параметрам, відхилять такий пакет. Схема фіксованої розсилки зображена на рис. 2.6.

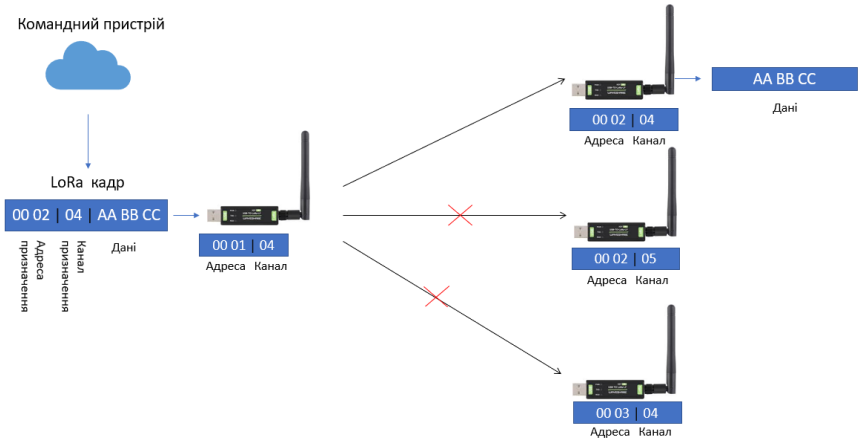


Рисунок 2.6 – Схема фіксованої розсилки

Широкомовній режим передачі даних використовується при ініціалізації мережі, що дозволяє пристроям поширити дані про свою адресу, доступність та відстань. Це дозволяє виконати ініціалізацію мережі та оновити таблицю маршрутизації комірок кластеру у автоматичному режимі.

Кадр LoRa включає в себе декілька ключових компонентів. Одним з таких елементів є Coding Rate (CR), що представляє собою механізм для виправлення помилок, забезпечуючи надійність корисного потоку даних. Іншим елементом є

Cyclic Redundancy Check (CRC), який є формою циклічної перевірки на наявність зайвої інформації в коді. Основні структурні елементи кадру LoRa охоплюють преамбулу, опціональний заголовок та корисне навантаження. Преамбула має вирішальне значення для синхронізації приймача з потоком даних. За умовчанням, довжина преамбули налаштована на послідовність із 16 символів, проте цей параметр можна змінити програмно. Важливо пам'ятати, що будь-які зміни в довжині преамбули мають бути скоординовані як на етапі передачі, так і на етапі прийому для забезпечення синхронізації між відправником та приймачем.

Технологія LoRa виділяє два типу кадрів, фіксованої та плаваючої довжини. Режим фіксованої довжини встановлюється за замовчуванням і тоді заголовок містить інформацію об довжині корисного навантаження в байтах, інформацію об CR та наявності додаткового 16-ти бітового CRC для корисного навантаження. Такий заголовок має максимальний розмір коригування ошибок (CR) 4/8 і має власний CRC. Заголовки з невірним CRC відкидаються миттєво. Структуру пакета фіксованої довжини наведено на рис. 2.7.

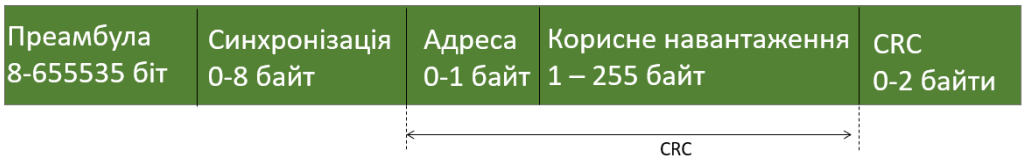


Рисунок 2.7 – Структура пакету фіксованої довжини

Фіксованої заголовок використовується у випадках, коли інформація об корисном навантаженні, CR та CRC відома зазделегіть. В такому випадку можливе скорочування часу передачі даних із застосуванням неявного заголовка. Для цього потрібно налаштувати у ручному режимі CRC на стороні прийому та передачі. Якщо пакет має невизначений або змінний розмір, тоді інформація про довжину пакета може бути передана в межах цього ж самого пакету. Схему пакету з плаваючиєю довжиною наведено на рис. 2.8. В такому випадку корисне навантаження не має фіксованого розміру.

Рисунок 2.8 – Структура пакету плаваючий довжини

2.3 Формування пакетів для передачі мережевої інформації

Для вирішення завдань квалфікаційної роботи були реалізовані спеціальні команді пакети для налаштувань мережі у автоматичному режимі. Особливість реалізації полягає у використанні не стандартної преамбули зі значеннням 100 біт. Щоб використовувати командні пакети – програмне забезпечення усіх елементів БСМ ОЕС повиннно бути оновленно до останньої версії. Структура кадру майже не змінюється окрім фіксованної преамбули та плаваючого розміру пакету.Таким чином, мінімальний розмір мережевого пакету буде 15 байт, а максимальний - 256 байт. На рис. 2.9 наведено приклад мережевого пакету.

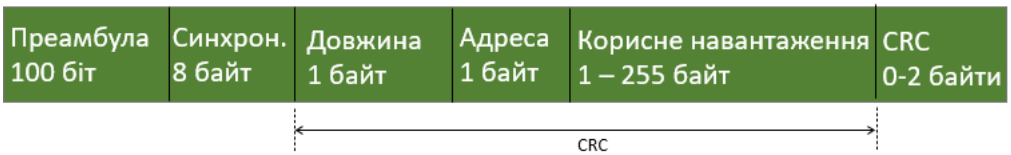


Рисунок 2.9 – Структура мережевого пакету

Також можна звернути увагу на протокол комутації LoRaWAN. Він знаходить широке застосування у різноманітних сферах IoT, включаючи смарт-сіті, смарт-ферми, віддалене моніторингове обладнання та багато інших сценаріїв, де потрібна далека передача даних з низьким споживанням енергії. Зазвичай усі повідомлення в мережі LoRaWAN зашифровані і мають наступний вигляд: 80C02301260021000266EEA76CCE0C1BBC7A36F69F. Отримані пакети необхідно дешифрувати за допомогою програмного забезпечення. Приклад реалізації програмного забезпечення для виконання поставленної задачі дешифрування кадру LoRa мовою Python наведено на рис. 2.10.

```
1   from lora.crypto import loramac_decrypt
2
3   app_session_key = 'F15283A7F32868AB674FE137265A4535'
4
5   payload = input("Enter LoRa message")
6   sequence_counter = int(payload[14:16] + payload[12:14], 16)
7   dev_addr = payload[8:10] + payload[6:8] + payload[4:6] + payload[2:4]
8   decrypted_payload = loramac_decrypt(
9                                       payload[18:34],
10                                      sequence_counter,
11                                      app_session_key,
12                                      dev_addr)
```

Рисунок 2.10 – Код для дешифрування повідомлень LoRaWAN

Для взаємодії з модулями LoRa використовується одноїмена бібліотека. В даному конкретнему випадку використовується частина бібліотеки для роботи з Крипто модулями lora.crypto. Функція loramac_decrypt використовується для дешифрування зашифрованих повідомлень LoRa. Далі на рядку 3 задається змінна app_session_key, яка містить ключ сесії у вигляді рядка. Цей ключ використовується для дешифрування повідомлень і є унікальним для кожної сессії. В рядку 5 користувач вводить до терміналу зашифроване повідомлення LoRa та зберігає їх. Наступники кроками з рядків 6-7 відбувається розбиття кадру на частини щоб відокремити преамбулу, адресу, канал та корисне навантаження. З 8 по 12 рядок виконується функція дешифрування з перестановленими параметриами. Цей код використовується для підготовки повідомлень і для формування власних мережевих пакетів, що дозволяє отримати сумісність власного протоколу з LoRaWAN.

Однако слід зазначити що LoRaWAN працює тільки з топологією типу «зірка». В певних реалізаціях БСМ це може бути суттєвим недоліком, тому використання LoRaWAN в якості основного протоколу комутації не рекомендується.

Таким чином було проаналізовано переваги та недоліки модулів зв’язку SemTech SX1262 у вигляді USB-конектора від Waveshare, а також порівняно чипсети версії SX1262 та SX1278. На основі проведеного порівняльного аналізу було рекомендовано замінити модулі зв’язку EBYTE E32 SX1278 на Waveshare USB-TO-LoRa SX1262. Було розглянуто особливості модуляції LoRa та режими роботи мережевих модулів. Для виконання поставлених задач квалифікційної роботи рекомендовано використовувати режим широкомовної розсилки при ініціалізації мережі. Пристрої повинні надавати інформацію про себе до наближених мережевих модулів для заповнення маршрутних таблиць серед інших модулів зв’язку. Після встановлення з’єднання пристрої перейдуть у режим фіксованой розсилки, що дозволить виконувати задачі комутації даних. Для виконання задач виокремлення мережевих пакетів було створено окремий клас комутаційних пакетів. Протокол LoRaWAN рекомендується використовувати в якості протоколу високого рівня для організації безпроводової сенсорної мережі оптико-електронних станций.

3 АНАЛІЗ МОЖЛИВОСТЕЙ СТВОРЕННЯ ОКРЕМИХ КЛАСТЕРІВ У МЕРЕЖІ LORA ТА МАСШТАБУВАННЯ БЕЗПРОВОДОВОЇ МЕРЕЖІ

3.1 Кластеризація у безпроводових сенсорних мережах

У рамках досліджень та розробок в області мережевих технологій, кластеризація безпроводових сенсорних мереж відокремлюється як одна з ключових стратегій, спрямованих на підвищення ефективності, масштабованості та надійності сучасних систем. Цей процес передбачає групування мережевих вузлів у кластери, які функціонують під керівництвом визначених керуючих вузлів. Основною метою такої організації є оптимізація мережевого управління та розподілу навантаження та ресурсів.

Кластеризація в мережевих системах являє собою процес організації вузлів у підгрупи, відомі як кластери, що спрямований на оптимізацію управління ресурсами та підвищення загальної ефективності мережі. Використання кластерів сприяє централізації управління в мережі, завдяки чому зменшується складність управління, особливо великими системи, а також забезпечує масштабованість мережі, дозволяючи легко інтегрувати нові вузли. Крім того, такий підхід ефективно знижує загальне мережеве навантаження, оскільки групування вузлів у кластери оптимізує мережевий трафік. У разі відмови одного з вузлів, кластерна структура може забезпечити резервування та автоматичне перенаправлення запитів, підвищуючи надійність мережі. Такий підхід дозволяє розбити об'ємні та складні мережі на більш керовані підсистеми, що сприяє покращенню продуктивності мережі, зниженню навантаження на неї, збільшенню її надійності, а також підвищенню відмовостійкості.

В мережі вузли можуть бути розподілені на кластери залежно від різноманітних факторів: географічне положення, тип обладнання, функціональні характеристики або обсяг навантаження. Застосування кластеризації є особливо важливим у сфері безпроводових сенсорних мереж, де вона сприяє ефективному збору даних та забезпечує енергоефективність. У таких мережах лідер кластера відіграє роль координатора, збираючи дані з інших вузлів у кластері та передаючи їх у агрегованому вигляді до центрального вузла або базової станції.

Завдяки кластеризації, мережеві системи набувають здатності до масштабування та пристосування до різних умов експлуатації. З огляду на переваги, кластеризація підвищує ефективність мережі через централізоване управління ресурсами, забезпечує гнучкість та адаптивність мережі до змінних умов. Також, вона сприяє балансуванню навантаження, рівномірно розподіляючи мережеві ресурси та обов'язки між вузлами.

Враховуючи ключові характеристики модулів зв'язку та специфічні вимоги до мережі ОЕС, було вирішено застосувати топологію гібридного типу. Основні переваги даної топології полягають у гнучкості її імплементації та незалежності окремих вузлів один від одного. Це дозволяє з легкістю замінювати несправні вузли без потреби конфігурації топології всієї мережі. При цьому, розроблені команди та алгоритми управління спрощують даний процес.

Однак, проблемою може стати централізація трафіку в одній точці, характерна для топології «зірка» або «дерево». Наприклад, вихід з ладу центрального вузла може призвести до збоїв в маршрутизації всієї мережі.

При масштабуванні мережі слід враховувати пропускну спроможність кожного пристрою. В даному випадку, вузьким місцем є командний вузол або інший девайс що знаходиться у центрі топології. На рис. 3.1 зображено приклад топології БСМ для ОЕС, де сірим кольором зображено пристрої збору, обробки та передачі даних, а командний вузол виокремлено червоним кольором.

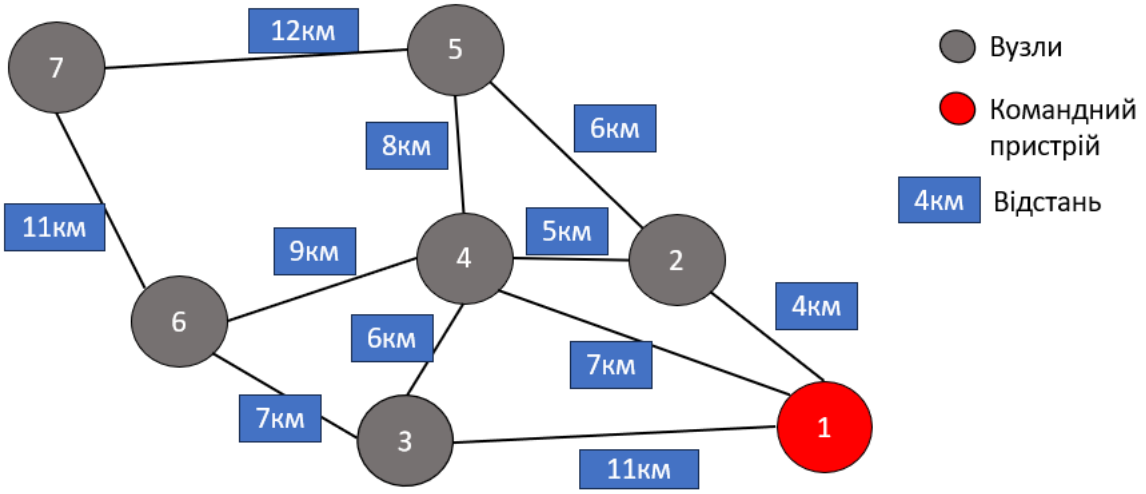


Рисунок 3.1 – Приклад топології мережі

Як можна побачити з рис. 3.1, вузол під номером 4 – становиться центром топології, і з точки зору базових принципів маршрутизації – центральним

маршрутизатором. Для повноцінного функціонування мережі, під час проектування необхідно робити розрахунки навантаження на кожен вузол. Особливістю безпроводової мережі ОЕС є те, що командний пристрій – це персональний комп'ютер або інший потужний девайс. Він займається збором даних об мережі, її стані, а також займається обробкою відеопотоку, аналізом даних що надходять з усіх пристроїв. В цей же час, вузли ОЕС – це оптико-електронна станція з підключеними до неї мікроконтролером або іншим девайсом та мережевий модуль. В розділі 3.2 кваліфікаційної роботи наведено рисунки та схеми побудови пристроїв в мережі ОЕС.

Оскільки усі модулі належать до класу С, та виконують функцію FFD – у мережі для усіх модулів буде використовуватись фіксований режим передачі. При обчисленні ліміту масштабування необхідно враховувати максимальний розмір даних, який може передавати LoRa. У поточній реалізації обмеження максимальної довжини повідомлення було збільшено з 58 до 256 байтів.

Модулі LoRa використовують механізм черг FIFO. Механізм черги FIFO (First-In-First-Out) є однією з фундаментальних концепцій у області інфокомунікацій. Цей механізм функціонує за принципом "першим прийшов - першим обслуговується", що означає, що елементи даних обробляються в порядку їх надходження. У структурі FIFO елементи зберігаються у черзі, де перший елемент, що надійшов у чергу, стає першим елементом, що виходить з неї. Це забезпечує простоту та передбачуваність, але може не бути оптимальним з точки зору ефективності для деяких сценаріїв, оскільки не враховується пріоритетність пакетів, що надходять до черги. Але використання цього механізму дозволяє забезпечити порядок і справедливість у обробці мережевого трафіку, але може привести до переповнення буферу черг. Алгоритм обробки черги в технології LoRa наведено на рис. 3.2.

Рисунок 3.2 – Алгоритм обробки черги

У технології LoRa, як тільки дані надходять з контролера і пакет досягає свого максимального розміру, відразу ж розпочинається передача даних. Користувач може продовжувати вводити дані, які будуть передаватися послідовно. Якщо розмір введених даних менший за 256 байтів і користувач не додає додаткові дані, модуль очікуватиме 1 секунду перед початком передачі.

В поточній задачі управління мережею та установками ОЕС, існує вимога до регулярного оновлення даних та передачі командних пакетів. Кожна ОЕС вимагає створення чотирьох командних пакетів для підтримки режиму своєї функціональності: два пакети призначені для управління сервоприводами, один - для керування камерою, та один - для управління тепловізором, а також необхідно отримувати службову інформацію та передавати транзитні пакети.

При визначенні максимально можливої кількості пристроїв у мережі ОЕС, ключовим фактором є пропускна здатність командного вузла. Ця пропускна здатність може бути під впливом зовнішніх чинників, а також потужності та дальності передачі сигналів. Враховуючи особливості модуляції LoRa, можна зробити висновок, що коротший маршрут – вигідніший маршрут, оскільки при

збільшені дистанції буде збільшено параметр Spreading Factor і в результаті швидкість передачі буде зменшена. У таблиці 3.1 наведено базовий приклад залежності параметрів швидкості передачі ланих від Spreading Factor та Coding Rate. Слід зауважити що на швидкість передачі даних будуть впливали зовнішні фактори і інші налаштування пристроїв.

Таблиця 3.1 – Залежність швидкості від коефіцієнту поширення

Spreading Factor (SF)	Bandwidth (BW) [kHz] (пропускна здатність)	Coding Rate (CR)	Bit Rate [bps] (швидкість)
SF7	125	4/5	5470
SF8	125	4/5	3125
SF9	125	4/5	1758
SF10	125	4/5	977
SF11	125	4/5	537
SF12	125	4/5	293

Швидкість передачі даних (біт/с) можна розрахувати за наступною формулою:

$$Rb = (SF * 4 * (BW/2SF)) / (4 + CR), \quad (3.1)$$

де Rb – швидкість передачі даних, біт/с;

SF – фактор розповсюдження;

CR – коефіцієнт кодування;

BW – смуга пропускання (пропускна здатність).

Для вирішення завдання масштабування та маршрутизації в мережі, необхідно організувати комутацію між кластерами, або ж графами. Приклад топології і потенційної задачі маршрутизації наведено на рис. 3.1. Для пошуку оптимальних маршрутів від командного вузла до інших кластерів або кінцевих пристроїв можуть використовуватись алгоритм Дейкстри або Беллмана-Форда. В четвертому розділі кваліфікаційної роботи наведене програмне забезпечення, яке вирішує задачі з побудови моделі графів, а також розрахунку найкоротших маршрутів з кожної точки мережі до іншої.

3.2 Основні елементу кластера, структура елементів

Безпроводові сенсорні мережі складаються з низки ключових компонентів, включаючи кінцеві пристрої та командні вузли. Кінцеві пристрої в цих мережах виконують роль як сенсорів, так і виконавчих контролерів. Стандартна конфігурація кінцевого пристрою включає декілька основних елементів. Перший з них - система зв'язку, яка дозволяє пристрою взаємодіяти з іншими вузлами мережі. Наступний - мікропроцесор, призначений для обробки даних або команд. Останній ключовий елемент - це джерело живлення, необхідне для підтримки роботи всіх компонентів пристрою.

На рис. 3.3 зображено структурну схему типового кінцевого пристрою для ОЕС, що відображає організацію та взаємодію вищезазначених елементів. Наведена схема демонструє взаємозв’язок компонентів для забезпечення функціонування пристрою в рамках безпроводової сенсорної мережі.

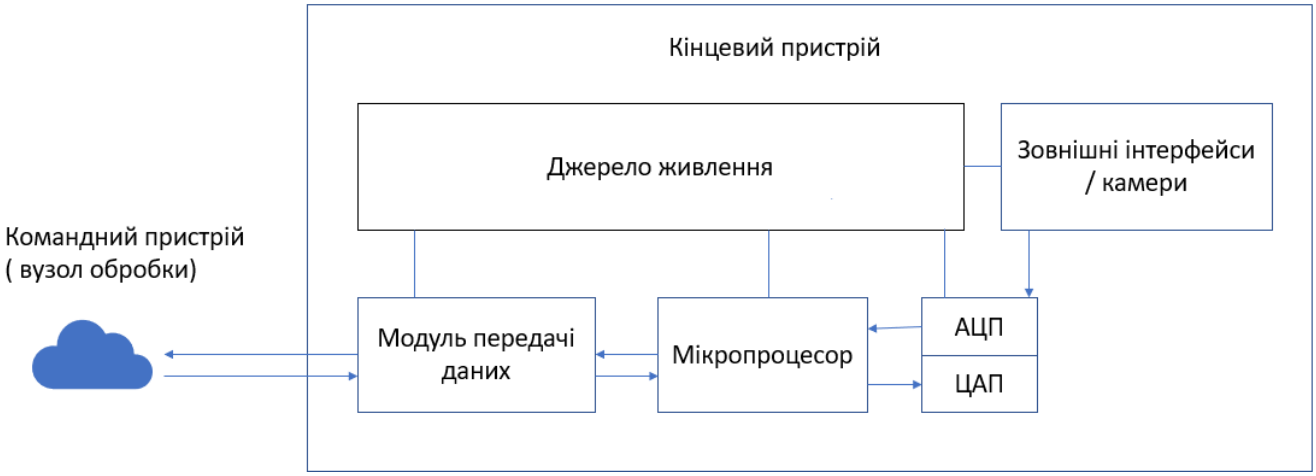


Рисунок 3.3 – Структурна схема кінцевого пристрою

Через зовнішні інтерфейси можуть підключатись серводвигуни або інші елементи. Дані підключених девайсів поступають до аналого-цифрового перетворювача, що дозволяє перекодувати сигнал з аналогового на цифровий. Далі ці дані передаються на мікропроцесор, або інший девайс. На фінальному етапі модуль комунікації даних виконує передачу даних. На рис. 3.4 зображено схему організації кінцевого пристрою або ж звичайного вузла БСМ ОЕС. Він складається з оптико-електронної станції, мікроконтролеру та мережевого модулю. Мікроконтролер потрібен для збереження мережевої інформації.

Рисунок 3.4 – Схема організації кінцевого пристрою

Як можна побачити з рис. 3.1, кінцеві пристрої приєднуються до командного вузла. Схему організації командного вузла наведено на рис. 3.5. Командний вузел має сервер обробки даних та підключенний мережевий модуль.

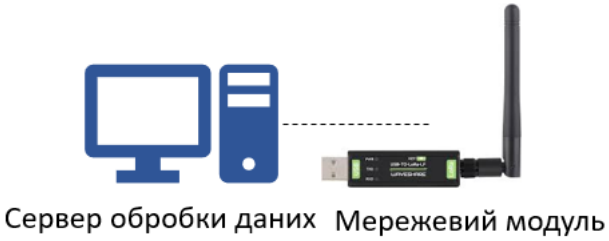


Рисунок 3.5 – Схема організації командного вузла з модулем SX1262

Для отримання сумісності вже з існуючею конфігурацію, можуть використовуватись модулі EBYTE E32. Схема підключення зображена на рис. 3.6.

Рисунок 3.6 – Організація командого вузла з модулем EBYTE E32

Комірка кластеру може складатись з одного або декількох мережевих девайсів. Приклад топології наведено на рис. 3.6. В середині кластерної зони 2, знаходиться командний пристрій 2.3, що виконує задачі комунікації між іншими комірками кластера, а також тримає зв’язок з двома пристроями (2.2 та 2.1), що не мають зв’язку з загальною мережею. Таке розподілення допомагає зменшити навантаження на БСМ з ОЕС.

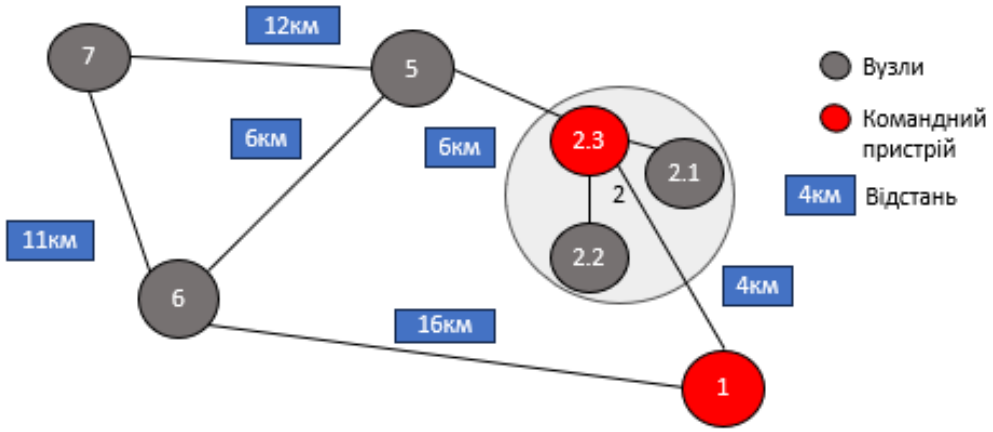


Рисунок 3.7 – Приклад топології мережі з двома командними вузлами

З рис. 3.7 можна побачити, що відстань між вузлами 1 та 6 дорівнює 16 кілометрам. Ця відстань більша, ніж технологія LoRa підтримує. Рішенням в даному випадку буде застосувати вузол-ретранслятор, який буде виконувати тільки задачі маршрутизації в мережі. Ретранслятори в мережі LoRa відіграють вирішальну роль у підвищенні ефективності та масштабованості безпроводових комунікаційних систем, особливо в контексті організації сенсорної мережі. Ці пристрої призначені для підсилення та перенаправлення сигналів між кінцевими вузлами та центральними мережевими станціями, забезпечуючи посилення сигналу та збільшення зони покриття, а також збільшення надійності зв'язку. На рисунку 3.8 наведено топологію з доданим вузлом ретранслятором (7).

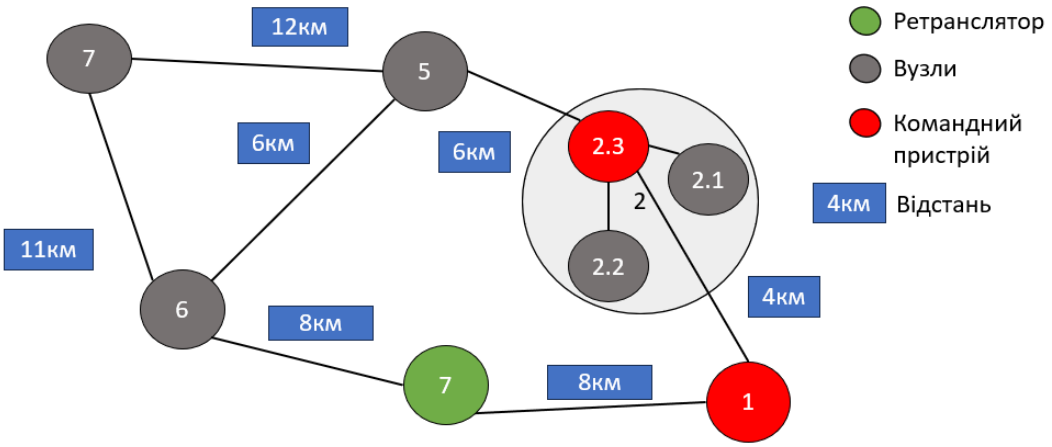


Рисунок 3.8 – Топологія з доданим ретранслятором

До додавання вузла 7, маршрут з точки 1 до 6 відбувався через вершини 2.3 та 5, дистанція складала 16 км. Якщо проаналізувати оновлену топологію, то

маршрут з вузла 1 до вузла 6 буде мати лише один вузол посередник. Таким чином, завдяки додаванню ретранслятора, вдалось зменшити навантаження на інші вузли кластеру.

3.3 Алгоритми створення кластерів у безпроводових сенсорних мережах

У сфері безпроводових сенсорних мереж існує значна кількість різноманітних протоколів маршрутизації. Серед них важливе місце займають ієрархічні протоколи, такі як PEGASIS, LEACH, TEEN, що організовують вузли в кластери за ієрархією. Ці протоколи спрямовані перш за все на підвищення енергоефективності мережі, забезпечуючи оптимальну доставку даних до базових станцій шляхом групування вузлів в кластери.

Розробка ефективних алгоритмів кластеризації є однією з актуальних задач у галузі БСМ. Ефективний алгоритм це ключовий чинник для підвищення продуктивності та зменшення енергоспоживання мереж.

Як вже згадувалось у першому та другому розділі кваліфікаційної роботи однією з можливих топологій мережі LoRa є "Кластерне дерево". Під час формування мережі, координатор відправляє широкомовні повідомлення та команди всім сусіднім пристроям. Після цього вузли надсилають запит на долучення до кластеру. Якщо мережевий координатор дає дозвіл, то встановлюється зв'язок з вузлом, який починає періодично відсилати команди для приєднання інших пристроїв. Такий підхід до формування мережі дозволяє оптимізувати процес маршрутизації та енергоспоживання у мережах LoRa, а також сприяє підвищенню масштабованості та гнучкості мережевої структури. Алгоритм формування кластеру наведено на рис. 3.9. Формування вузлів в кластер спирається на такі критерії як географічний радіус та енергетичний баланс. Лідером кластеру може стати лише командний вузол, що містить сервер для оброки даних. Командний вузол обирається тільки в ручному режимі і не може бути сформований автоматично на поточний момент. Таким чином кластер та комірки кластеру будуть формуватися за географічним принципом навколо командного пристрою. В даному випадку поріг додавання девайсів до комірки кластеру становить 40% від максимальної дальності роботи пристрою. Цей поріг може бути змінено у налаштуваннях розробленого програмного забезпечення. Поріг у 40% зумовлений споживанням енергій та умовами позиціонування оптико-електронних станції.

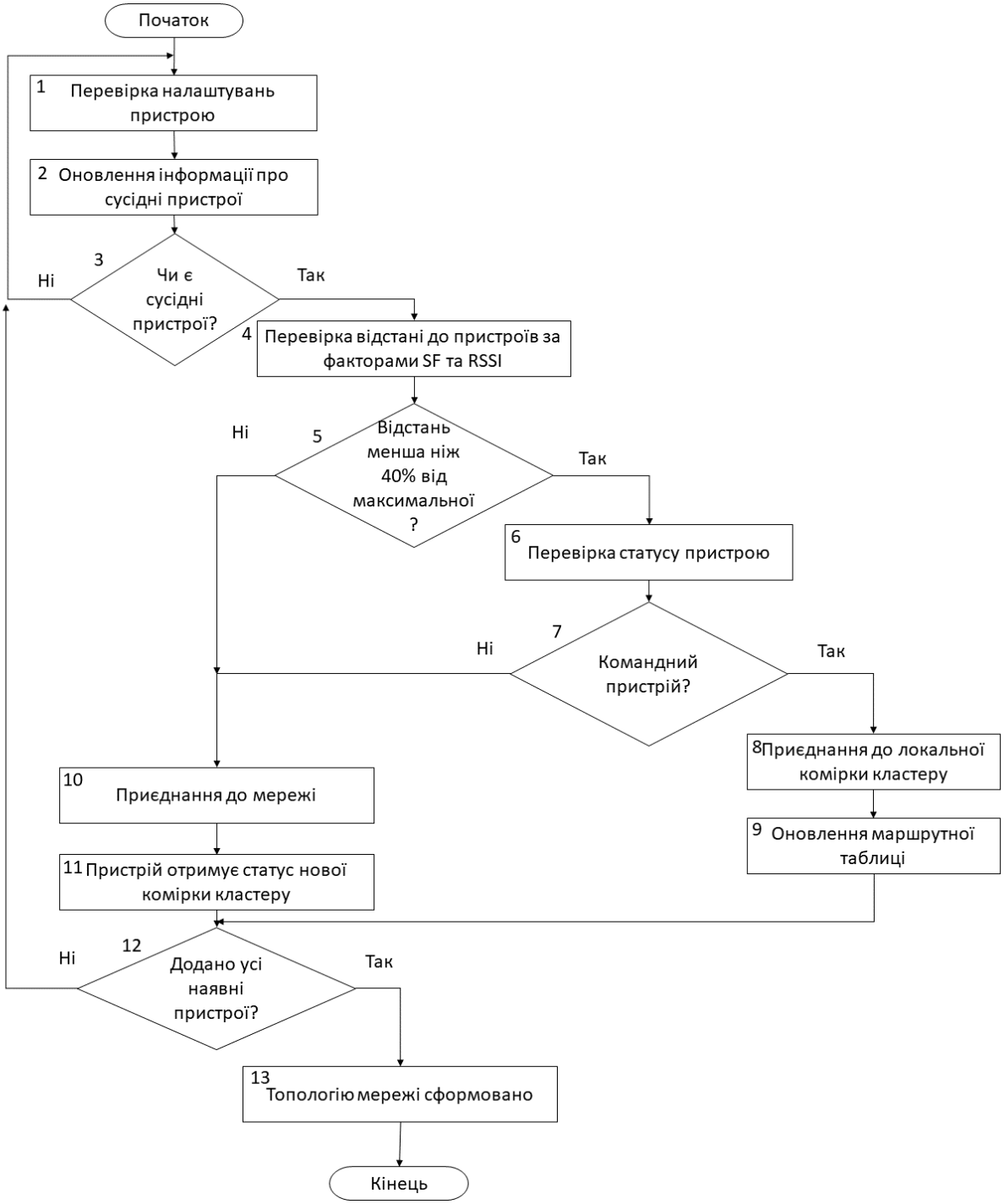


Рисунок 3.9 – Алгоритм формування кластерів в БСМ з ОЕС

Весь трафік обробляється в координаторі топології, у випадку з ОЕС координатором виступає командний вузол. Розроблене програмне забезпечення, для вирішення задач маршрутизації створює таблицю маршрутизації на командному вузлі, а також мікроконтролерах інших пристроїв, що створюють комірки кластерів. Ця таблиця використовується для спілкування з кінцевими пристроями. В таблиці утримається інформація об номері кінцевого пристрою,

адресі, каналі. При використанні номера ОЕС з таблиці, в пакети даних підставляється адреса та канал з таблиці. Приклад маршрутної таблиці заповнено в таблиці 3.2. Важливим показником в даному випадку є RSSI (Received Signal Strength Indicator), що вимірює рівень потужності сигналу, який отримує приймач, та є ключовим показником, що відображає якість зв'язку між двома пристроями. У LoRa, RSSI часто використовується для оцінки відстані між передавачем та приймачем, оскільки сила сигналу зменшується зі збільшенням відстані. Це означає, що нижчий показник RSSI може вказувати на більшу відстань між вузлами або наявність перешкод, які ослаблюють сигнал. Характеристики сили сигналу може допомогти визначити оптимальні місця розміщення шлюзів LoRa, тому цей показник враховується при формуванні кластеру. Слід зауважити, що на RSSI впливають зовнішні фактори, такі як перешкоди, інтерференція та інші ефекти розповсюдження радіочастотного сигналу.RSSI вимірюється за логарифмічною шкалою у децибелах над рівнем шуму (дБм). Пристрої LoRa визначають RSSI автоматично під час прийому даних.

Таблиця 3.2 – Приклад таблиці маршрутизації

Вузол похо-дження	Кінцевий вузол	ТТЛ	Spreading Factor	RSSI
0x01	0x03	300	SF12	-73
0x02	0x03	250	SF7	-75
0x05	0x03	1	SF7	-73

За результатами досліджень та проектування було розглянуто можливість створення кластерів у БСМ ОЕС. На рисунку 3.1 розглянуто ситуація з одноранговою мережею, а на рисунку 3.7 зображено приклад задачі маршрутизації при багаторанговій мережі. Алгоритм організації кластеру наведено на рисунку 3.9. Тип організації мережі має лінійну залежність від відстані, тобто якщо залежні вузли знаходяться на невеликій відстані один від одного, 25% або 40% від максимального радіуса дії – то можливо організувати кластер з декількома пристроями, з яких тільки один буде мати зв’язок з іншою коміркою кластеру. Додавання пристроїв до однієї комірки кластеру допоможе отримати кращу енергоефективність.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОРГАНІЗАЦІЇ БЕЗПРОВОДОВОЇ СЕНСОРНОЇ МЕРЕЖІ ОПТИКО-ЕЛЕКТРОННИХ СТАНЦІЇ ТА РОЗРАХУНКУ МАРШРУТІВ ІЗ ЗАСТОСУВАННЯМ ТЕХНОЛОГІЇ LORA

4.1 Алгоритм ініціалізації пристроїв у безпроводовій сенсорній мережі LoRa із використанням чипу Semtech SX1262

Алгоритм ініціалізації мережі є критично важливим у контексті створенні та управлінні будь-якою мережевою інфраструктурою, особливо в контексті складних і розподілених систем, таких як безпроводові сенсорні мережі. Цей алгоритм відіграє вирішальну роль у визначенні первинної конфігурації мережі, встановленні з'єднань між вузлами та оптимізації загальної продуктивності мережевих операцій.

Процес ініціалізації розпочинається з виявлення та ідентифікації всіх вузлів та обладнання, що входять до складу мережі. Це включає визначення фізичних та логічних адрес, конфігурацію каналів зв'язку та встановлення необхідних параметрів для забезпечення зв'язку між вузлами. Особливо це стосується безпроводових мереж, де точність налаштувань впливає на ефективність використання радіочастотного спектру, мінімізацію інтерференції та загальну завадохазищенність. На рис. 4.1 зображено алгоритм ініціалізації мережевих пристроїв з моменту ввімкнення живлення. Як можна побачити з алгоритму, одразу після ініціалізації мережі – пристроїв сповіщають інші сусідні пристрої про свою адресу. Якщо відповідь отримано, то далі аналізується можливість входження до комірки кластеру при відповіді певним умовам розташування та сили сигналу, або ж створення нової комірки кластеру. Якщо налаштування завершено – мережа готова до функціонування. Слід брати до уваги, що за потреби можуть бути виконані додаткові налаштувань в ручному порядку.

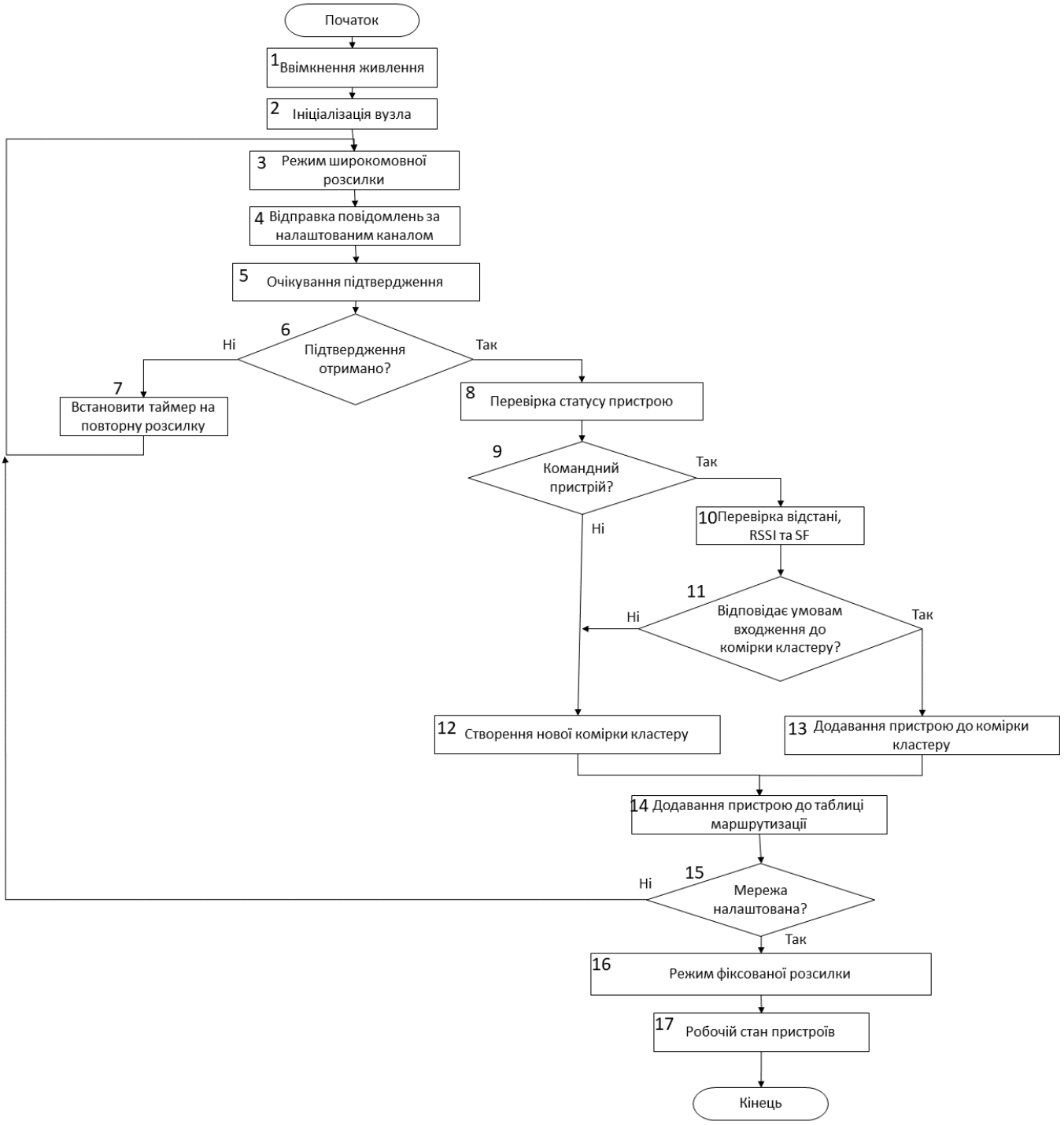


Рисунок 4.1 – Алгоритм ініціалізації пристроїв у мережі

Наведений на рис. 4.1 алгоритм демонструє можливості автоматичного налаштування мережі шляхом відсилання даних про мережевій пристрій у широкомовному режимі в рамках поточного каналу роботи. Також існує можливість виконати автоматичне налаштування мережі через виклик процесу ініціалізації за допомогою команди *addStation(00;0000;00)* [1,3].

4.2 Алгоритм отримання пакетів у безпроводовій сенсорній мережі LoRa

Щоб забезпечити певний рівень автоматизації мережевого протоколу, було змінено поведінку протоколу LoRa за замовчуванням. На рис. 4.2 наведено алгоритм отримання пакетів, а також маршрутизації в мережі.

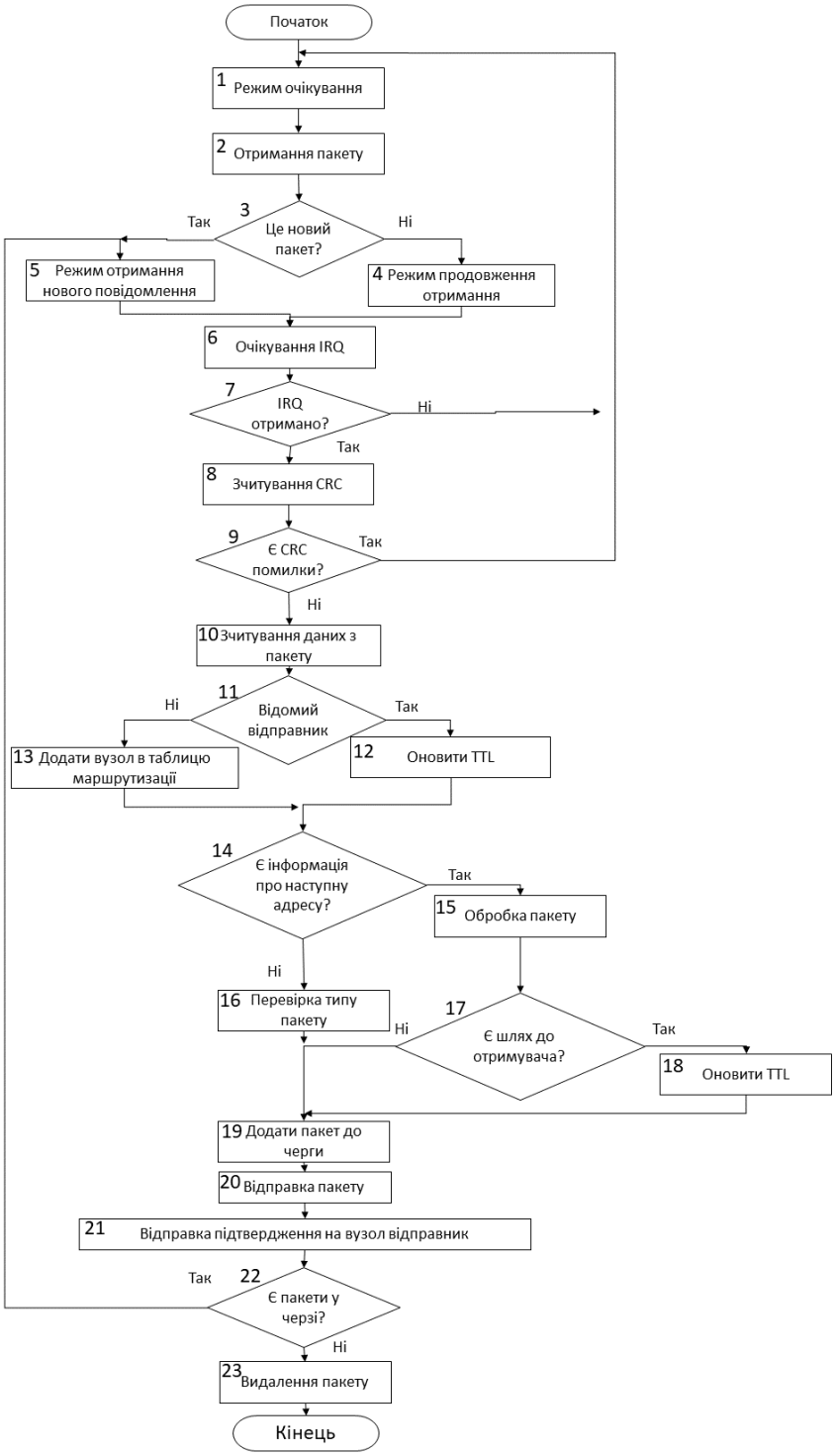


Рисунок 4.2 – Алгоритм отримання повідомлень та оновлення таблиці маршрутизації

На рис. 4.2 наведено алгоритм, при якому мережа вже ініціалізована та відомі маршрути. При отримані пакета розпочинається перевірка даних. Для мережі важливо знати, це нова передача або продовження передачі великого повідомлення з черги. Після перевірки очікується підтвердження що пакет отримано, передачу завершено і пакет не містить помилок або пошкоджених даних. Якщо відправник пакета невідомий, то таблиці маршрутизації записується адреса та інші відомості про відправника, а у іншому випадку просто оновлюється TTL (Time-To-Live), показник скільки часу цей маршрут буде актуальним.

Якщо отриманий пакет містить в собі інформацію про наступну адресу отримувача – пакет буде пересланий далі або ж буде лише оброблений на цьому вузлі. Для відправлення пакету далі по мережі формується черга. Після відправлення пакету, на вузол відправник повідомляється інформація що пакет було переслано далі. Це може бути корисним при пошуку проблем у мережі.

Алгоритми з рис. 4.1 та рис. 4.2 дозволяють автоматизувати процеси ініціалізації та маршрутизації у мережі. Розподіл задач маршрутизації між кінцевими пристроями дозволяє зменшити навантаження на командрий пристрій і особливо на маршрути передачі даних, що дуже впливає на організацію безпроводової сенсорної мережі.

4.3 Формування оточення програмного забезпечення для управління мережею з оптико-електронних станції

Для виконання задач кваліфікаційної роботи було розроблене програмне забезпечення у вигляді веб-застосунку, яке візуалізує розташування оптико-електронних станції на картах OpenStreetMap.

OpenStreetMap (OSM) являє собою проект цифрового картографування, що базується на принципах відкритих даних і колаборативної роботи. Він був створений з метою надання вільного доступу до геопросторових даних усім користовучам. OSM можна охарактеризувати як відкриту альтернативу комерційним картографічним сервісам, що дає користувачам можливість створювати, редагувати та використовувати географічні дані у вільному доступі.

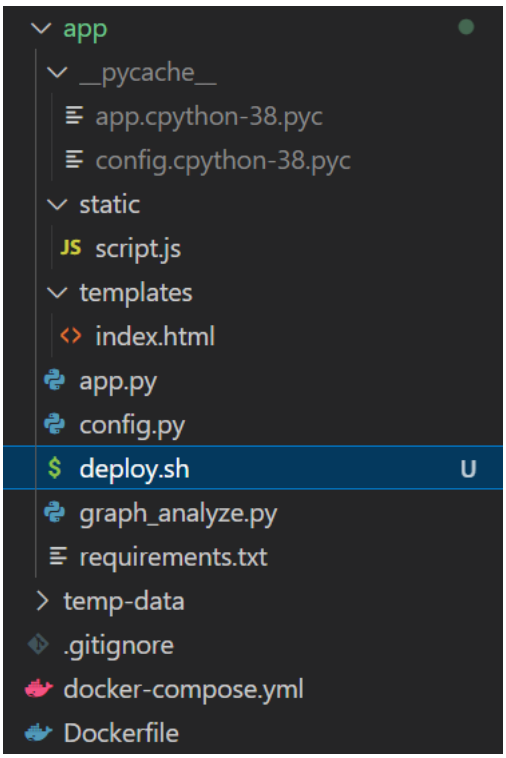


Рисунок 4.3 – Структура директорії проекту

Структура директорій програмного забезпечення наведена на рис. 4.3. Файли проекту зберігаються у директорії app, а файли docker-compose.yml та Dockerfile містять в собі інструкції для розгортування віртуального образу з усіма необхідними бібліотеками та залежностями. Веб-застосунку повністю запускається на платформі Docker.

Docker є високоефективною платформою для автоматизації розгортання, управління та масштабування застосунків у контейнеризованому середовищі. Він використовує технологію контейнеризації, що дозволяє ізолювати застосунки та їх залежності у легкі, переносні контейнери. Ці контейнери виконуються на одній і тій же хост-операційній системі, але вони відокремлені один від одного та від самої хост-системи. Це забезпечує безпеку, консистентність та ізоляцію ресурсів. Однією з ключових переваг Docker у цілях виконання кваліфікаційної роботи та розгортання веб-застосунків є його легкість та гнучкість. Docker дозволяє швидко створити, протестувати та розгорнути застосунок, гарантуючи, що він буде працювати однаково в будь-якому іншому середовищі, тому повністю відсутня залежність від локальних умов.

Такий підхід значно спрощує управління інфраструктурою та етап розгортки програмного забеспечення, а також знижує ризики, пов'язані з несумісністю середовищ. Контейнери Docker потребують значно менше ресурсів,

ніж традиційні віртуальні машини, оскільки вони ділять ядро операційної системи хоста, а не емулюють цілу ОС. Це дозволяє запускати більше застосунків на одному і тому ж апаратному обладнанні, зменшуючи витрати на інфраструктуру. Зміст docker-compose файлу наведено на рис. 4.4.

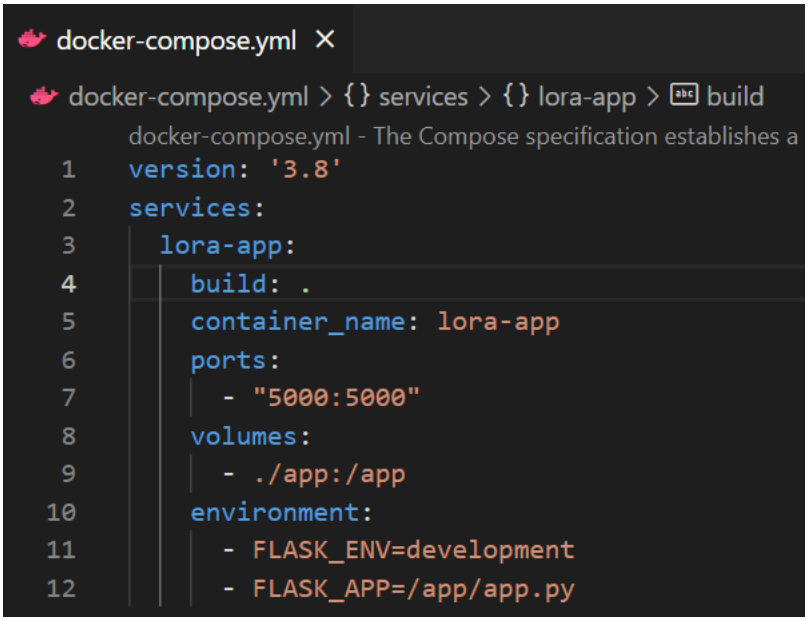


Рисунок 4.4 – Структура docker-compose файлу

Цей текстовий файл містить в собі ключові інструкції для ядра Docker, а саме версію, опис сервісів що повинні бути розгорнути, для кожного окремого сервісу можна прописати мережеві та інші налаштування. Розгоратння оточення виконується командой *docker-compose up -d*. Ця команда ініціює процес розгортання застосунку, описаного у docker-compose.yml, виконуючи ряд кроків для запуску та координації роботи всіх контейнерів, необхідних для застосунку. Якщо базовий образ не було скомпільовано, то ядро Docker зробить це. Завдяки використанню ключа -d, контейнер буде запущено у фоновому режимі, відокремленому від поточного вікна терміналу.

Поточний файл конфігурації написаний у версії 3.8, описує сервіс з веб-застосунком lora-app, використовуючи ряд параметрів для конфігурації відповідного контейнера:

- вказано version: '3.8', що позначає використання специфікації Docker Compose версії 3.8. Ця версія включає підтримку новіших функцій та можливостей Docker;

- в секції services визначено один сервіс lora-app, що описує спосіб розгортання та управління контейнером для застосунку LoRa;
- інструкція build: . вказує на те, що образ для контейнера має бути зібраний з Dockerfile, розташованого у поточній директорії;
- рядок container_name: lora-app задає ім'я контейнера, що полегшує його ідентифікацію та управління;
- вказівка ports: - "5000:5000" означає, що порт 5000 всередині контейнера буде прив'язано до порту 5000 на хост-машині, дозволяючи зовнішнім користувачам звертатися до застосунку. Видимість застосунку залежить від налаштувань firewall та анти-вірусного програмного забезпечення;
- параметер volumes: - ./app:/app створює спільний об'єм між локальною директорією `./app` та директорією `/app` всередині контейнера. Це забезпечує синхронізацію файлів між хостом та контейнером, що є корисним для розробки та тестування, оскільки дозволяє змінювати код без необхідності перебудови контейнера;
- у секції environment визначено дві змінні середовища: FLASK_ENV=development та FLASK_APP=/app/app.py. Ці змінні використовуються для конфігурації поведінки Flask застосунку в контейнері.

Наступний важливий файл для оточення це Dockerfile. Він є текстовим файлом, який містить набір інструкцій та команд, які використовуються Docker для автоматичного створення образу контейнера. Цей файл визначає середовище, необхідне для запуску та виконання застосунку, включаючи операційну систему, бібліотеки, залежності, конфігураційні файли та інші ресурси. Dockerfile дозволяє стандартизувати та автоматизувати процес створення образів, забезпечуючи консистентність та відтворюваність середовища застосунку. Зміст Dockerfile наведено на рис 4.5.

Рисунок 4.5 – Зміст Dockerfile

В даному випадку Dockerfile містить наступну інформацію:

- FROM python:3.8-slim – ця інструкція вказує на використання базового образу Python версії 3.8 у "slim" варіанті, що забезпечує легковаговий фундамент для контейнера;
- COPY app/requirements.txt /app/requirements.txt – команда копіює файл `requirements.txt` з локальної директорії `app` у директорію `/app` всередині контейнера. Цей файл містить список залежностей Python застосунку;
- RUN pip install --no-cache-dir -r /app/requirements.txt - ця команда використовує менеджер пакетів pip для встановлення залежностей, перерахованих у файлі requirements.txt, наведено на рис 4.6. Опція --no-cache-dir запобігає зберіганню кешу, що зменшує розмір віртуального образу;
- EXPOSE 5000 - ця інструкція вказує Docker, що застосунок слухає порт 5000. Хоча `EXPOSE` не публікує порт автоматично, він інформує Docker про те, що застосунок використовує цей порт;
- ENTRYPOINT ["tail", "-f", "/dev/null"] – встановлює команду, яка буде виконуватися за замовчуванням при запуску контейнера. У даному випадку, використання tail -f /dev/null фактично утримує контейнер у робочому стані. Це використовується оскільки застосунок запускається через окремий bash-скріпт.

На рис. 4.6 наведено перелік бібліотек, що використовуються для реалізації даного проекту.

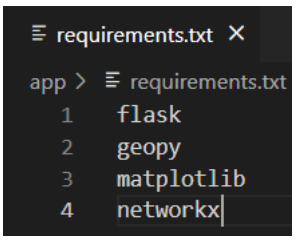


Рисунок 4.6 – Бібліотеки Python для віртуального образу

Flask є мікрофреймворком для розробки веб-додатків на Python, який заснований на двох ключових бібліотеках: Werkzeug, інструментарій WSGI (Web Server Gateway Interface), та Jinja2, система шаблонів. Flask пропонує мінімалістичний підхід до розробки, надаючи базовий набір функцій для веб-додатків із можливістю розширення за допомогою численних розширень. Він підтримує маршрутизацію запитів, сесії, обробку шаблонів та інші ключові аспекти веб-додатків, одночасно забезпечуючи гнучкість та легкість у використанні.

GeoPy є бібліотекою Python, яка дозволяє розробникам отримувати доступ до різних геокодувальних служб та обчислювати геодезичні відстані між точками. Вона підтримує ряд відомих геокодувальних сервісів, включаючи Google Geocoding API, Nominatim від OpenStreetMap, Bing Maps API та інші. GeoPy дозволяє здійснювати геокодування, обернене геокодування (знаходження адрес за координатами) та обчислення відстаней з врахуванням кривизни Землі.

Matplotlib - це бібліотека для створення статичних, інтерактивних та анімованих візуалізацій у Python. Як фундаментальний інструмент для візуального аналізу даних, Matplotlib надає обширні можливості для створення графіків та діаграм різного типу, включаючи лінійні графіки, гістограми, кругові діаграми, діаграми розкиду та багато інших. Бібліотека пропонує гнучкий інтерфейс та широкі можливості для налаштування візуалізацій, від простих графіків до складних композитних візуалізацій.

NetworkX - це бібліотека Python, призначена для створення, маніпулювання та вивчення структури, динаміки та функцій складних мереж. З її допомогою можна створювати графи, які включають вузли та ребра, а також виконувати різноманітні аналізи та обчислення на графах, наприклад, пошук найкоротших шляхів, класифікацію вузлів, аналіз ступеня зв'язності та багато іншого. NetworkX відома своєю здатністю обробляти великі мережі та надає інструменти для

візуалізації графів, хоча для більш складних візуалізацій часто використовуються інші бібліотеки, такі як Matplotlib.

4.4 Інтерфейс програмного забезпечення для управління мережею з оптико-електронних станції

Інтерфейс програмного забезпечення написано мовою верстки HTML, а основні розрахунки виконуються на зворотній стороні застосунку мовою програмування Python. Для функціонування внутріньої логики програмного забезпечення та отримання зв’язку front-енду та back-енду використовується мова JavaScript. Інтерфейс програмного забезпечення наведено на рис 4.7.

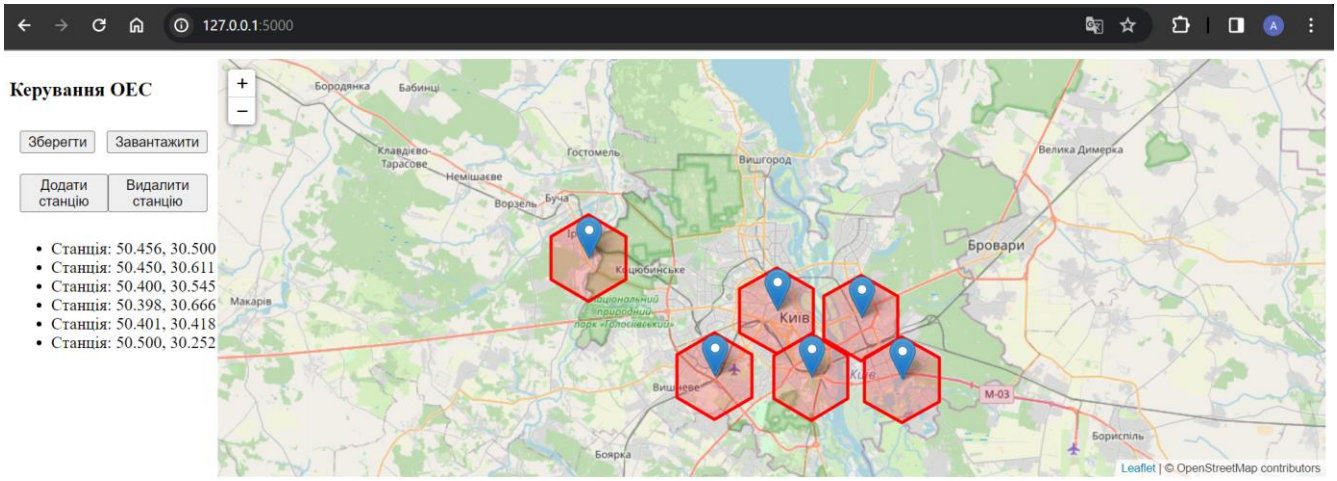


Рисунок 4.7 – Інтерфейс веб-застосунку

Як можна побачити з рис. 4.7, застосунок розгорнуто на локальному комп’ютері, про це свідчить адреса 127.0.0.1, а також застосунок доступний за портом 5000, як і було описано в інструкціях для Docker. Використання порту 5000 обумовлено конфігурацією веб-серверу Flask, але може бути змінено в файлі app.py.

Ліву частину екрану застосунку займає панель керування. За допомогою кнопок управління можна виконувати наступні функції:

- додати станцію;
- видалити станцію;
- зберегти конфігурацію;
- завантажити конфігурацію.

На правій стороні екрану відображається інтекрактивна карта OpenStreetMap з використанням фреймворку Laflet. Кожна оптико-електронна станція відображається синьою міткою, а комірка кластеру – червоним шестикутником. Форма шестикутнику обрано з метою зображення зони покриття з меншими спотвореннями, а також з метою спрощення візуалізації. В поточній реалізації автоматично не рахуються інтерфераційні вади, що задаються рел'ефом або ж високими спорудами. Це може мати ефект при проектуванні мереж у міській забудові, але цей недолік можна компенсувати у файлі конфігурації застосунку. На рис. 4.8 наведено зміст файлу конфігурації, в якому можна змінювати параметри далекобійності, розміру кластерів, порогу масштабування та додавання ретрансляторів.

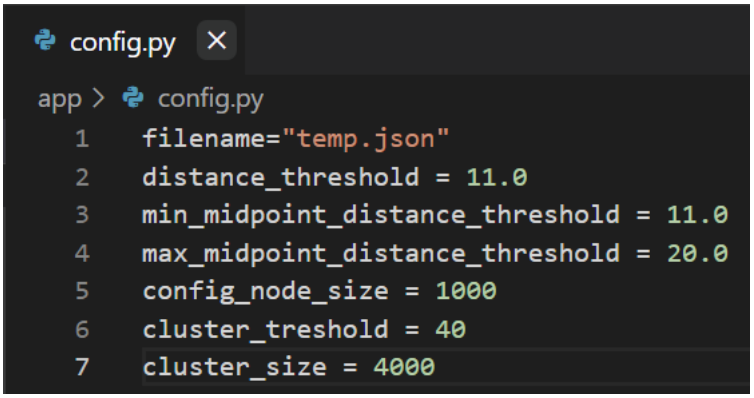


Рисунок 4.8 – Зміст файлу конфігурації

Для виконання операції з додавання станції, користувачу достатньо нажати на кнопку «додати станцію» та наступним кроком розташувати прогнозовану або поточну позицію ОЕС на мапі у правій частині екрану. При потребі, масштаб відображення може бути змінено через активні клавіши. Після додавання станції, її координати у форматі (широта; довгота) будуть навдені у панелі керування під кнопками управління. Слід зауважити, що у застосунку присутня топографічна інформація об усіх містах та об'ектах земної кулі. Наведена розробка ніяким чином не обмежена в географічному застосувані. Для видалення станції зі списку та карти, достатньо нажати кнопку «Видалити станцію», а потім на мапі обрати станцію.

Поточна конфігурація то розташування пристроїв може бутти збережена як конфігурціоний файл у форматі JSON. В файлі зберігаються координати кожної станції,а також інформація про зону покриття станції. При виконанні операції зі

збереження або завантаження файлу конфігуарції відкривається діалог. На рис. 4.9 наведено вікно діалогу.

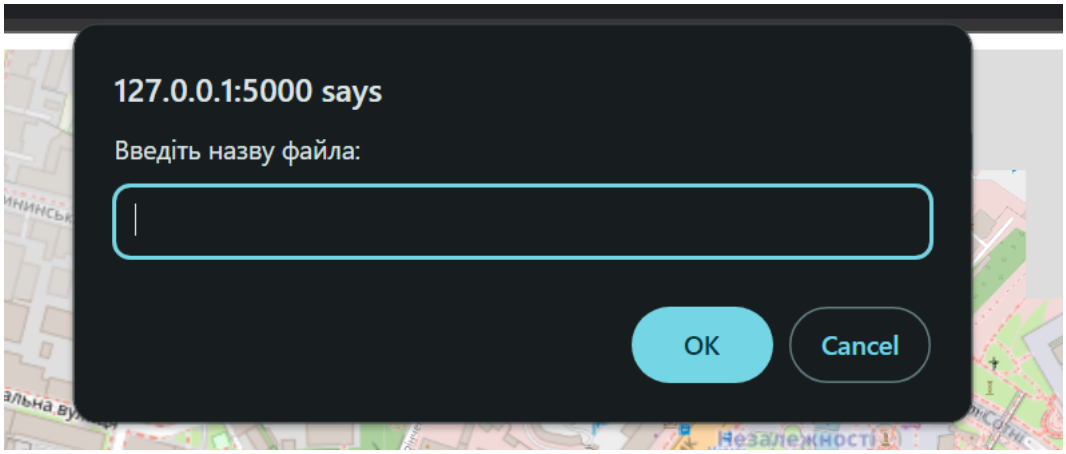


Рисунок 4.9 – Вікно для збереження та завантаження файлу конфігурація

Ключові залежності та змінні, яки використовуються при обчисленнях містяться в файлі config.py та зображені на рис. 4.10.

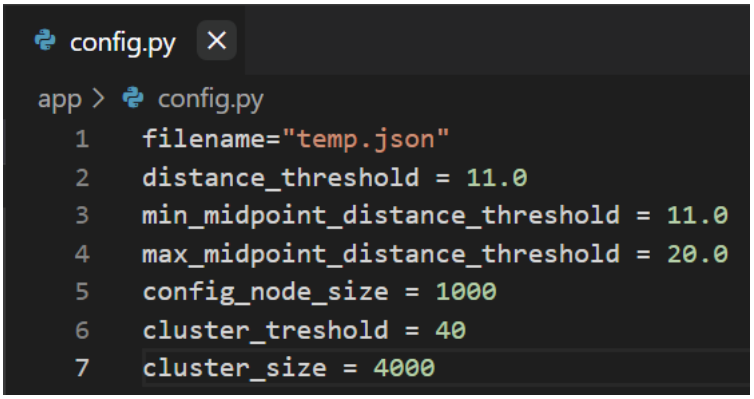


Рисунок 4.10 – Зміст файлу конфігурації

Перелік змінних та їх значення виглядає таким чином:

- filename відповідає за назву файлу, який буде автоматично створюватись при збереженні конфігурації. Цей файл буде дублікатом для оригінального файлу збереженного користувачем. Він також використовується для автоматичного обчислення маршрутів в топології мережі.
- distance_threshold – це зміна, яка характеризує максимальну відстань між вузлами мережі. Іншими словами це максимальний радіус дії поточних мережевих модулів Semtech SX1262. Величина цієї змінної – кілометри.

- min_midpoint_distance_threshold – характеризує мінімальну відстань, від якої буде створюватись вузол ретранслятор. Величина цієї змінної – кілометри.
- max_midpoint_distance_threshold – характеризує максимальну відстань, від якої буде створюватись вузол ретранслятор. Величина цієї змінної – кілометри. Значення цієї змінної описує верхню межу, тобто якщо між вузлами відстань більша ніж значення змінної – вузол не буде створено. Основна мета цих обмежень це організація одночасно декількох незалежних кластерів.
- config_node_size виконує роль показника масштабу для автоматичного малювання графу.
- cluster_threshold – ця змінна описує поріг відстані, у процентах на додавання пристроїв до комірки кластеру. Цей процесс детальніше описано у третьому розділі квалифікаційної роботи
- cluster_size – це значення для радіусу внутрішньої окружності для побудови шестикутників на карті веб-застосунку. Це значення вимірюється в метрах.

Основний файл застосунку – це app.py. В ньому описані основні функції та методи, а також уся логика програмного забезпечення. Цей файл запускається за замовчуванням при старті віртуального контейнеру, або ж може бути розгорнуто в мануальному порядку за командою: *python3 app.py*. Для запуску в ручному режимі необхідно підключитись до терміналу контейнера.

Робота будь-якого застосунка на мові Python розпочинається з процесу імпорту бібліотек. На рис. 4.11 наведено імпорт встановлених залежностей, необхідних для функціонування веб-застосунку.

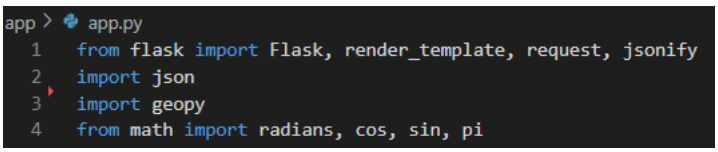


Рисунок 4.11 – Імпорт бібліотек для веб-застосунку

Після імпорту бібліотек, зазвичай описуються класи та функції програмного забезпечення. В даному випадку було розроблено декілька функції для обчислень. На рис. 4.12 наведено функцію, що виконує розрахунки координат кожної вершини шестикутника, який буде відображатись на карту веб-застосунку.

```python
def calculate_hexagon(lat, lng, radius):
    coords = []
    start_point = (lat, lng)

    for i in range(6):
        angle = 60 * i  # Кут для кожної вершини (60 градусів)
        radian = radians(angle)  # Переведення в радіани

        new_lat = lat + cos(radian) * (radius / 111320)  # 111320 - приблизна довжина градуса широти в метрах
        new_lng = lng + sin(radian) * (radius / (40075000 * cos(radians(lat)) / 360))  # Довжина градуса довготи в метрах

        coords.append((new_lat, new_lng))

    return coords
```

Рисунок 4.12 – Функція розрахунку зони покриття кластеру

Як можна побачити з рис. 4.12, функція calculate_hexagon реалізує алгоритм для генерації координат шестикутника (гексагона) на основі заданої географічної точки та радіусу. Цей алгоритм використовує геодезичні обчислення для визначення шести вершин шестикутника, розташованих на однаковій відстані від центральної точки, заданої широтою (lat) та довготою (lng). В даному випадку центральна точка шестикутника ініціалізується як start_point з вхідними координатами lat та lng, які передаються до програми одразу після додавання ОЕС на карту. Потім, використовуючи цикл, функція обчислює координати шести вершин шестикутника. Для кожної вершини, визначається кут відхилення від центральної точки, що дорівнює 60 градусам (кутова частина правильного шестикутника), який переводиться у радіани для подальших обчислень.

Функція використовує геодезичні припущення для обчислення нових координат широти та довготи. Довжина градуса широти в метрах приблизно становить 111320 метрів, а довжина градуса довготи залежить від широти і обчислюється як частка від загальної довжини екватора (приблизно 40075000 метрів), помножена на косинус радіана широти. Ці величини використовуються для визначення зміщення кожної вершини шестикутника відносно центральної точки, з урахуванням радіусу шестикутника.

Результатом виконання функції є список кортежів, де кожен кортеж містить координати однієї з шести вершин шестикутника. Ці координати використовуються для візуалізації шестикутників на карті веб-застосунку. Такий підхід дозволяє виконувати точні геометричні розрахунки та моделювання, що є важливими у військовії галузі.

Наступний етап роботи застосунку, це ініціалізація веб-застосунку Flask, створення масиву для збережння інформації про станції, а також запис радіусу кластерау з конфігураціоного файла config.py. Цей етап наведено на рис 4.13.

```
app = Flask(__name__)

points = []
scale = cluster_size
```

Рисунок 4.13 – Ініціалізація застосунку Flask

Після створення застосунку Flask і пустогу масиву даних, слідує блок коду, який являє собою серію маршрутів веб-сервісу, написаних за допомогою фреймворку Flask для Python. Обробка маршрутів відповідає за обробку дій користувача та роботу з геопросторовими даними. Код наведено на рис. 4.14.

```
@app.route('/')
def index():
    return render_template('index.html')

@app.route('/add_point', methods=['POST'])
def add_point():
    data = request.json
    points.append(data)
    return jsonify(success=True)

@app.route('/remove_point', methods=['POST'])
def remove_point():
    data = request.json
    points.remove(data)
    return jsonify(success=True)

@app.route('/save_points', methods=['POST'])
def save_points():
    data = request.json
    filename = data['filename']
    points = data['points']
    if not points:
        return jsonify({'status': 'empty points list'})
    with open(filename, 'w') as file:
        file.write(json.dumps(points))
    with open("temp.json", 'w') as file2:
        file2.write(json.dumps(points))
    return jsonify({'status': 'success'})

@app.route('/load_points')
def load_points():
    filename = request.args.get('filename')
    with open(filename, 'r') as file:
        points = json.load(file)
        print(f' Loading poits: {points}')
    return jsonify({'points': points})

@app.route('/calculate_hexagon', methods=['POST'])
def calculate_hexagon_route():
    data = request.json
    lat = data['lat']
    lng = data['lng']
    zoom = scale
    coords = calculate_hexagon(lat, lng, zoom)
    return jsonify(coords)
```

Рисунок 4.14 – Основні маршрути веб-застосунку

В частині @app.route('/') визначає головну сторінку веб-додатку. Функція index відповідає за відображення головної HTML-сторінки (index.html). Це базова

точка входу для користувачів, з якої вони можуть взаємодіяти з іншими функціями додатку.

Маршрут @app.route('/add_point', methods=['POST']) обробляє додавання нової точки. Функція add_point отримує дані у форматі JSON через POST-запит, додає ці дані до списку points та повертає відповідь у форматі JSON, підтверджуючи успішне додавання.

У маршруті @app.route('/remove_point', methods=['POST']) визначена функція remove_point, яка виконує видалення точки. Вона отримує дані у форматі JSON та видаляє вказану точку зі списку points. Після видалення, функція відправляє підтвердження успішного виконання операції у вигляді відповіді JSON.

Маршрут @app.route('/save_points', methods=['POST']) включає функцію save_points, яка займається збереженням списку точок. При отриманні даних у форматі JSON, функція записує ці дані у файл, ім'я якого також передається у запиті. У разі відсутності точок у списку, відправляється відповідь з повідомленням про порожній список. Цей маршрут забезпечує збереження даних на сервері для подальшого використання або аналізу.

Функція під маршрутом @app.route('/load_points') - load_points - обробляє завантаження списку точок з файлу. Ім'я файлу отримується через параметри запиту, а дані точок завантажуються та відправляються клієнту у форматі JSON. Це дозволяє користувачам відновлювати раніше збережені дані.

Останній з маршрутів @app.route('/calculate_hexagon', methods=['POST']) визначає функцію calculate_hexagon_route, яка призначена для обчислення координат шестикутника з вказаною центральною точкою та радіусом. Дані (широта, довгота, радіус) отримуються у форматі JSON через POST-запит. Функція викликає допоміжну функцію calculate_hexagon, яка була описана раніше. Вона генерує координати вершин шестикутника, та повертає ці координати у відповіді JSON.

Розглянемо детальніше головний HTML файл веб-застосунку, який викликається першим маршрутом. Цей файл наведено на рис. 4.15, він є інструментом для візуалізації та керування станціями, підключеними через систему LoRa.

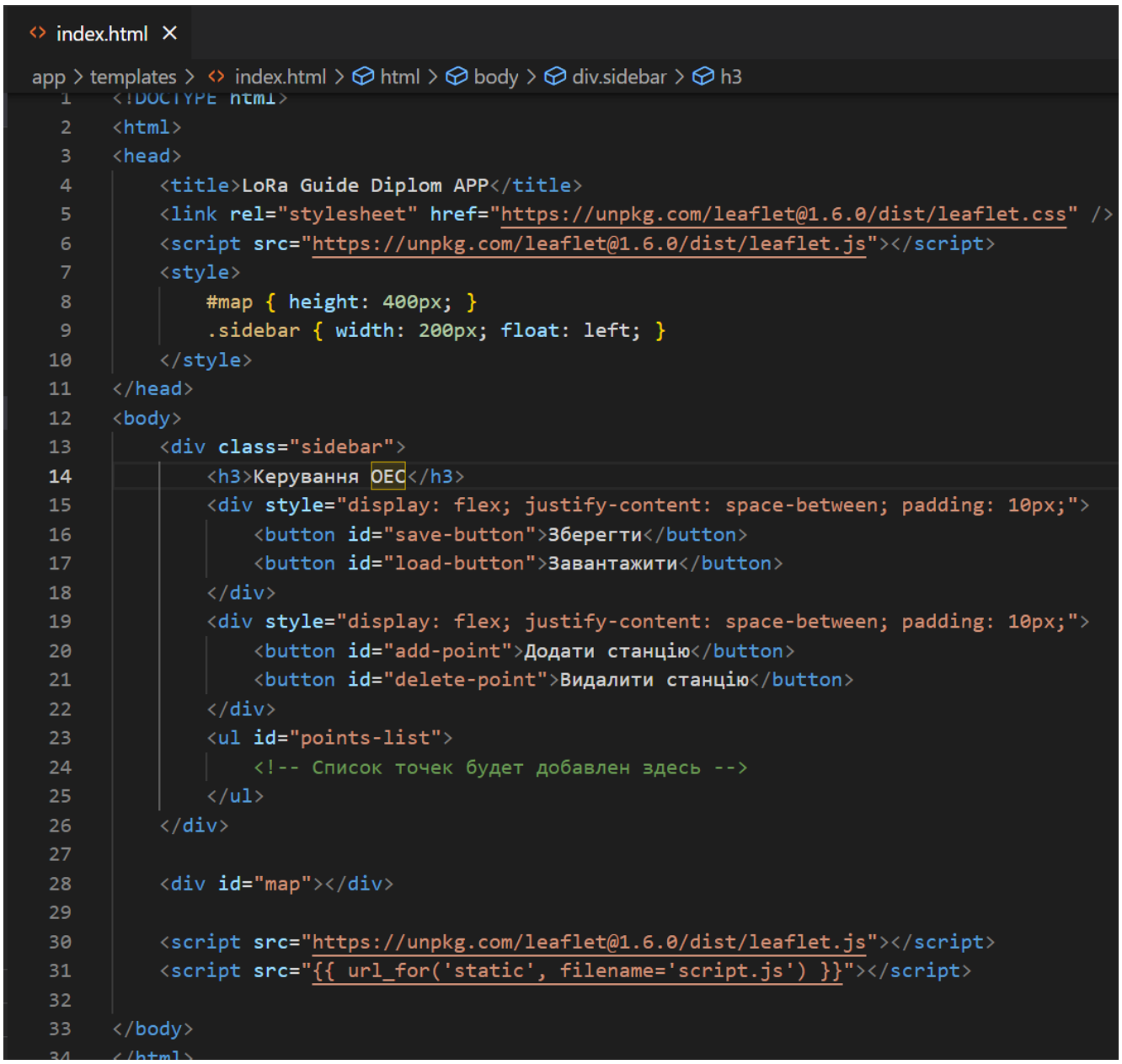


Рисунок 4.15 – Структура веб-сторінки застосунку

Цей код описує задачі інтерактивного планування мережі LoRa. Файл містить базову структуру веб-документа, включаючи елементи head та body.

В ньому міститься посилання на зовнішній CSS файл Leaflet, що є відкритою бібліотекою для інтерактивних карт. В HTML файлах можливе використання зовнішнього файлу стилів, а в даном випадку є вбудовані стилі CSS для визначення висоти карти (#map) та стилів для бічної панелі (sidebar).

Секція body є основною частиною файлу. Вона містить div з класом sidebar, який використовується для відображення інтерфейсу управління. Цей блок включає кнопки для збереження та завантаження даних, а також додавання та

видалення станцій. Також в цьому блоці є ul з ідентифікатором points-list призначений для динамічного відображення списку станцій.

Фінальною секцією є підключення JavaScript файлів. В розробленій конфігурації підключаються два скрипти. Це скрипт Leaflet, необхідний для роботи інтерактивної карти і зовнішній JavaScript файл, що містить логіку для керування елементами інтерфейсу та інтеграції з картами Leaflet. На рис. 4.16 наведено JavaScript функцію яка відповідає за збереження файлів, а на рис. 4.17 функція що відповідає за завантаження файлів.

```
function saveToFile(filename) {
    console.log(points);
    var dataToSave = { filename: filename, points: points };
    fetch('/save_points', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify(dataToSave),
    })
    .then(response => response.json())
    .then(data => console.log('Success:', data))
    .catch((error) => console.error('Error:', error));
}
```

Рисунок 4.16 – JavaScript функція для збереження конфігурації в файл

```
function loadFromFile(filename) {
    fetch('/load_points?filename=' + encodeURIComponent(filename))
        .then(response => response.json())
        .then(data => {
            if (!data.points) {
                throw new Error('Невірний формат даних');
            }
            clearMap();
            points = data.points;
            points.forEach(function(point) {
                if (point.type === 'point') {
                    L.marker([point.lat, point.lng]).addTo(map);
                } else if (point.type === 'hexagon') {
                    drawHexagonOnMap(map, point.center, point.radius);
                }
            });
        })
        .catch(error => console.error('Error:', error));
}
```

Рисунок 4.17 – JavaScript функція для завантаження конфігурації з файлу

Дві JavaScript функції, saveToFile і loadFromFile, взаємодіють з сервером для зберігання та завантаження даних, відповідно, у контексті веб-карти.

Функція saveToFile приймає аргумент filename і відправляє об'єкт dataToSave, що містить filename та змінну points, на сервер через POST-запит. Запит виконується до ендпоінта /save_points, використовуючи fetch API з JSON-форматованим тілом запиту. У відповідь сервера функція реагує логуванням результату або помилки.

Функція loadFromFile запитує з сервера дані за допомогою GET-запиту, використовуючи ім'я файлу filename як параметр запиту. В отриманих даних очікується поле points, що містить точки або шестикутники. Після очищення карти ця функція додає маркери або шестикутники на карту залежно від типу кожної точки (точка або шестикутник). У випадку помилки або невірного формату даних, виконується логування помилки.

Таким чином було розробленно інтерактивний веб-застосунок, який дозволяє взаємодіяти з картами OpenStreetMap у реальному масштабі часу. Завдяки використанням платформи Docker було досягнуто універсальність та компактність розгортання застосунку. З використанням декількох мов програмування одразу, вдалось досягти поставленної мети та реалізувати додаток, що спрощує масштабування мережі та візуалізує розташування ОЕС.

4.5 Програмне забезпечення для автоматичного розрахунку позиціонування ретрансляторів безпроводової сенсорної мережі

Розрахунок оптимальних маршрутів є ключовим елементом у плануванні та управлінні мережами, особливо при умові масштабування та кластерізаціїї. Цей процес включає в себе визначення найефективніших шляхів між мереживими пристроями. Оптимальні маршрути не лише сприяють економії часу та ресурсів, але й зменшують навантаження на мережу.

Додавання ретрансляторів є важливим процесом для підвищення якості та надійності безпроводових комунікацій, особливо в областях де є різок втрати сигналу. Ретранслятори виконують функцію посилення та пересилання трафіку, забезпечуючи більш широке покриття та кращу якість зв'язку.

Щоб вірищи задачу масштабування та розрахунку позиції ретрансляторів, а також обчислення оптимальних за відстанню маршрутів, були розроблені додаткові програмні модулі, які додаються до веб-застосунку у вигляді функцій. Як можна побачити з оригінальної топології, зображенної на рис. 4.18 та рис. 4.19,

одна ОЕС з лівої частини віддалена від інших та не має прямого маршруту до інших комірок кластеру.

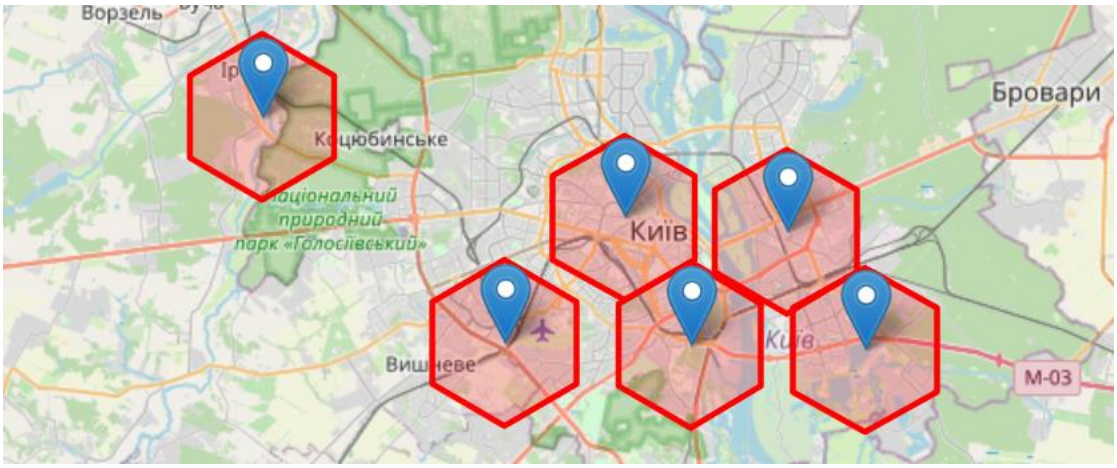


Рисунок 4.18 – Візуалізація розташувань ОЕС разом з комірками кластера на мапі

Розроблене програмне забезпечення виконує автоматичну побудову графої моделі, грунтуючись на координатах та топології мережі в збереженому JSON файлі. На рис. 4.19 наведено графову топологію, що була автоматично згенерована програмним забезпеченням. Між доступними пристроями, були побудовані маршрути, відстань яких можна побачити. До 5 вершини маршрут відсутній, оскільки вона знаходиться поза межою зони покриття БСМ.

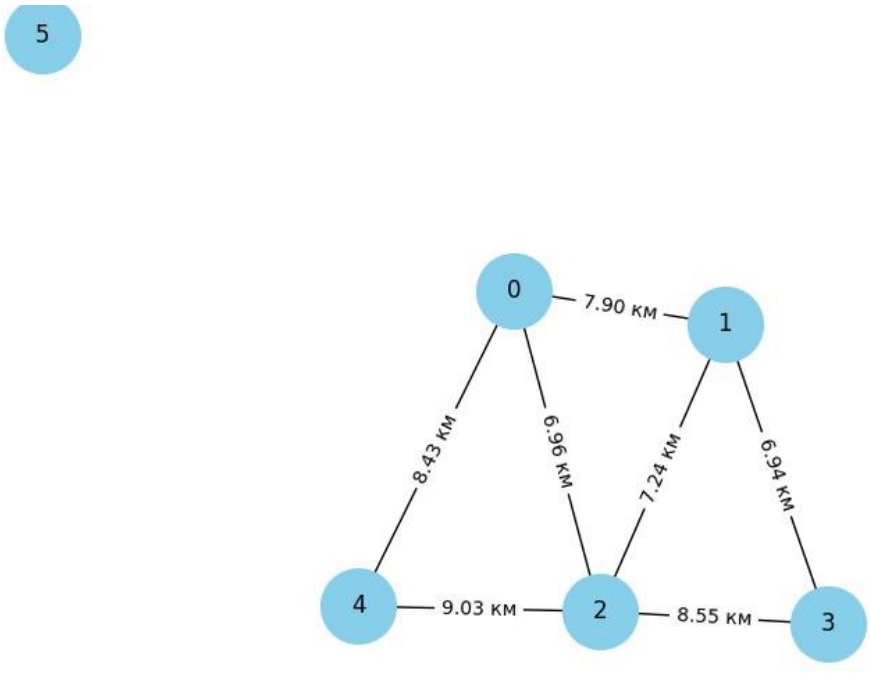
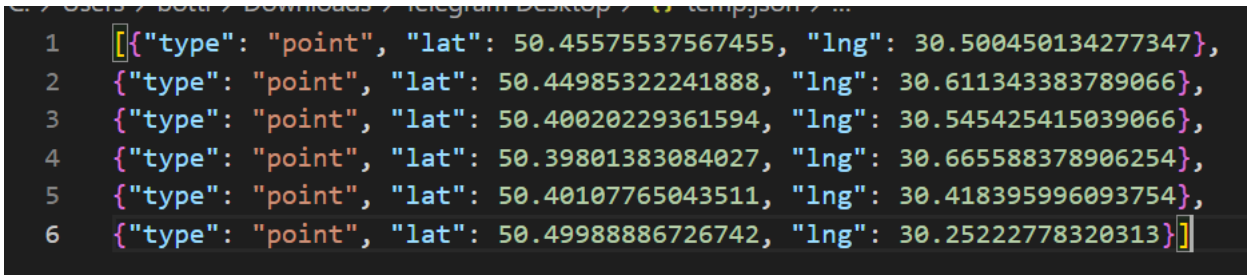


Рисунок 4.19 – Топологія мережі у вигляді графу

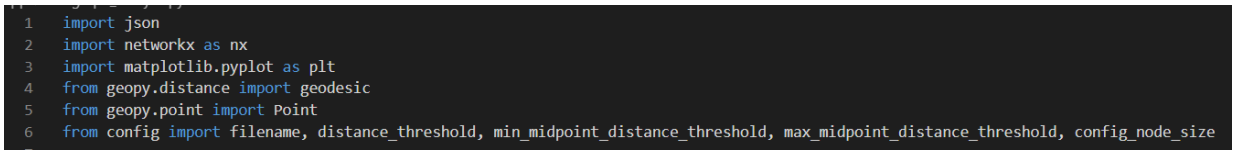


```
[{"type": "point", "lat": 50.45575537567455, "lng": 30.500450134277347},
{"type": "point", "lat": 50.44985322241888, "lng": 30.611343383789066},
{"type": "point", "lat": 50.40020229361594, "lng": 30.545425415039066},
{"type": "point", "lat": 50.39801383084027, "lng": 30.665588378906254},
{"type": "point", "lat": 50.40107765043511, "lng": 30.418395996093754},
{"type": "point", "lat": 50.49988886726742, "lng": 30.25222778320313}]
```


Рисунок 4.20 – Файл конфігурації мережі

Як можна побачити з рис. 4.20, файл містить 6 рядків, при чому кожний з них відповідає за окрему станцію. Це вже отформотований вигляд файлу, оскільки були видалені дані про зону покриття кластеру (координати шестикутника). Для кожної станції описані широта та довгота.

Модулі для розрахунку позицій ретрансляторів винесено в окремий файл під назвою graph_analyze.py. Імпорт бібліотек та конфігурації можна побачити на рис. 4.21.

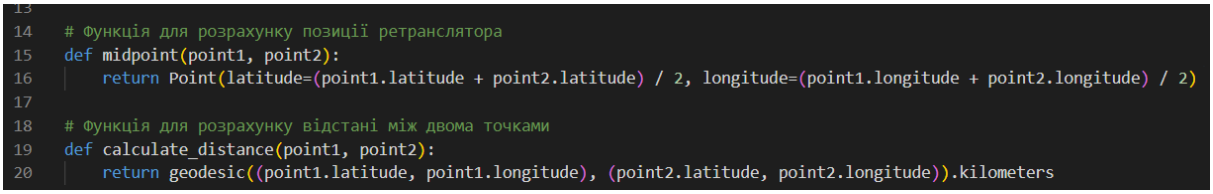


```python
import json
import networkx as nx
import matplotlib.pyplot as plt
from geopy.distance import geodesic
from geopy.point import Point
from config import filename, distance_threshold, min_midpoint_distance_threshold, max_midpoint_distance_threshold, config_node_size
```

Рисунок 4.21 – Імпорт бібліотек

Оточення Docker залишається повністю без змін, оскільки все залежності було інстальованно під час проектування веб-застосунку.

При виконанні обчислень координат розташування ретрансляторів, використовуються дві нові функцій, код яких наведено на рис. 4.22.



```python
# Функція для розрахунку позиції ретранслятора
def midpoint(point1, point2):
    return Point(latitude=(point1.latitude + point2.latitude) / 2, longitude=(point1.longitude + point2.longitude) / 2)

# Функція для розрахунку відстані між двома точками
def calculate_distance(point1, point2):
    return geodesic((point1.latitude, point1.longitude), (point2.latitude, point2.longitude)).kilometers
```

Рисунок 4.22 – Функцій обчислення координат та відстані

Перша функція, midpoint, призначена для розрахунку середньої точки (географічного центру) між двома заданими точками. Функція приймає два аргументи, point1 та point2, які представляють географічні координати (широту та довготу) двох ОЕС. Вона обчислює середнє значення широти та довготи цих двох

точок, тим самим визначаючи координати точки, що знаходиться прямо посередині між ними. Це використовується для визначення оптимального місця розташування ретранслятора в заданому регіоні, забезпечуючи рівномірне покриття сигналом між двома вже існуючими станціями.

Друга функція, calculate_distance, використовується для визначення відстані між двома географічними точками. Функція також приймає дві точки з географічними координатами як вхідні параметри. Вона обчислює відстань між цими точками, використовуючи геодезичний метод, який враховує кривизну Землі, що надає більш точний результат порівняно з простим евклідовим відстанню. Результат повертається у кілометрах.

Дані про розташування ОЕС зберігаються у файлі разом з іншою інформацією, що може завадити розрахункам. Оскільки оригінальний файл конфігурації містить інформацію про зону покриття, то його потрібно відформатувати та видалити зайву інформацію. Фрагмент коду, який виконує операцію форматування наведено на рис 4.23.

```
9   with open(filename, 'r') as file:
10      data = json.load(file)
11  filtered_data = [item for item in data if isinstance(item, dict) and item.get("type") == "point"]
12  filtered_json_data = json.dumps(filtered_data)
```

Рисунок 4.23 – Фрагмент коду для форматування початкових даних

Цей програмний модуль демонструє процес читання, фільтрації та серіалізації даних у форматі JSON. Спочатку використовуючи контекстний менеджер with, код відкриває файл, ім'я якого зберігається в змінній filename, у режимі читання. Використання контекстного менеджера забезпечує коректне закриття файлу після завершення операцій з ним. Далі, дані з файла завантажуються у форматі JSON за допомогою функції json.load, результатом чого є об'єкт Python, який зберігається у змінній data. Наступним кроком є фільтрація даних: зі списку data вибираються лише ті елементи, які є словниками і мають ключ "type" зі значенням "point". Цей фільтрований список зберігається у змінній filtered_data. На заключному етапі, фільтрований список filtered_data серіалізується назад у формат JSON за допомогою функції json.dumps, результатом чого є рядок у форматі JSON, що містить лише відфільтровані дані.

Маючи вже готові, отформатовані та серіалізовані дані, програма може обчислювати оптимальні маршрути. Функція, яка займається обчисленням усіх можливих, а далі й коротших маршрутів наведена на рис. 4.24.

```
def find_all_routes(G):
    routes = []
    for source in G.nodes():
        for target in G.nodes():
            if source != target and nx.has_path(G, source, target):
                path = nx.shortest_path(G, source=source, target=target, weight='weight')
                path_length = sum(calculate_distance(Point(G.nodes[path[i]]['pos']), Point(G.nodes[path[i + 1]]['pos'])) for i in range(len(path) - 1))
                routes.append((path, path_length))
    return routes
```

Рисунок 4.24 – Функція пошуку всіх маршрутів

Функція find_all_routes призначена для пошуку та обчислення усіх можливих маршрутів у графі G. Граф G є вхідним параметром функції і представляє собою мережу точок та їх з'єднань. Функція ініціює порожній список routes, який буде використовуватися для зберігання інформації про маршрути та їх загальні довжини. Далі вона перебирає всі вузли графа G двічі: один раз для визначення вихідної точки маршруту (source) та другий раз для визначення кінцевої точки (target). Для кожної пари вузлів source і target функція перевіряє, чи існує між ними шлях у графі, використовуючи функцію nx.has_path з бібліотеки NetworkX. Якщо шлях існує, функція використовує nx.shortest_path для обчислення найкоротшого шляху між цими точками, враховуючи вагу зв’язків, в даному випадку – це відстань

Після обчислення найкоротшого шляху, функція розраховує його загальну довжину. Це відбувається шляхом обчислення відстані між кожною парою послідовних вузлів у маршруті. Для цього використовується функція calculate_distance, яка приймає координати двох точок і повертає відстань між ними. Кожен маршрут разом з його загальною довжиною додається до списку routes. В кінці, функція повертає список routes, що містить всі можливі маршрути в графі G та їх відповідні загальні довжини. Додатком до цієї функції, є інша, що аналізує маршрути за алгоритмом Дейкстри та записує лише оптимальні. Код наведено на рис. 4.25. Алгоритм Дейкстри - це реалізація задачі пошуку найкоротшого шляху в графі, який знаходить мінімальну відстань від однієї вершини, званої "стартовою", до всіх інших вершин. Він працює тільки для графів з не від'ємною вагою ребер, працює за наступним чином:

- ініціалізація: Спочатку алгоритм ініціалізує набір відстаней від початкової вершини до кожної іншої вершини графа. Відстань до початкової вершини встановлюється як 0, а відстані до всіх інших вершин - як нескінченність;
- вибір вершини: Алгоритм обирає вершину з найменшою тимчасовою відстанню, яка ще не була відвідана, і розглядає всі її сусідні вершини. Цю вершину часто називають "поточною вершиною";
- релаксація ребер: Для кожного сусіда поточної вершини алгоритм оновлює відстань, якщо знайдено коротший шлях через поточну вершину. Це означає, що якщо сума відстані від початкової вершини до поточної вершини і вага ребра, що веде до сусіда, менша за поточну відстань до цього сусіда, тоді відстань до сусіда оновлюється;
- позначення вершини: Після того, як відстані до всіх сусідів були перевірені та потрібні оновлені, поточна вершина вважається "відвіданою", і вона більше не буде розглядатися для релаксації ребер;
- повторення: Кроки 2-4 повторюються, доки всі вершини не будуть відвідані;
- кінцевий результат: Після завершення алгоритму маємо набір найкоротших відстаней від стартової вершини до кожної іншої вершини в графі.

Функція save_optimal_routes_to_file використовується для знаходження та збереження оптимальних маршрутів між парами вузлів у графі G до текстового файлу. Вона ітерує через усі вузли графа, ігноруючи вузли ретранслятори. Для кожної пари реальних пристроїв ОЕС вона використовує алгоритм Дейкстри (реалізований функцією nx.single_source_dijkstra з бібліотеки NetworkX) для обчислення найкоротшого шляху та його загальної довжини, зважуючи краї вузлів.

```
def save_optimal_routes_to_file(G, node_colors, file_name):
    with open(file_name, 'w') as file:
        for source in G.nodes():
            if node_colors[source] == 'green':
                continue
            for target in G.nodes():
                if source != target and node_colors[target] != 'green':
                    try:
                        length, path = nx.single_source_dijkstra(G, source, target, weight='weight')
                        route_description = f"Маршрут від {source} до {target}: {path}, Відстань: {length:.2f} км\n"
                        file.write(route_description)
                    except nx.NetworkXNoPath:
                        pass
```

Рисунок 4.25 – Код функції обчислення оптимальних маршрутів

У випадку знаходження маршруту, інформація про нього (включаючи вузли шляху та загальну відстань) форматується у вигляді рядка та записується до файлу. Якщо маршрут між вузлами відсутній, обробляється виняток і програма продовжує виконання без збереження маршруту.

Усі раніше наведені функції з цього підрозділу виконуються разом у головній функції, яка обчислює граф, розташовує вузли ретранслятори, генерує новий JSON файл для завантаження у веб-застосунок та у фіналі записує оптимальні маршрути до файлу. Код наведено на рис. 4.26 та рис. 4.27.

```python
def create_graph_with_midpoints(json_data, distance_threshold, min_midpoint_distance_threshold, max_midpoint_distance_threshold):
    points = json.loads(json_data)
    G = nx.Graph()
    node_colors = ['skyblue' for _ in points]
    for i in range(len(points)):
        G.add_node(i, pos=(points[i]['lng'], points[i]['lat']), type='static-point')
        for j in range(i + 1, len(points)):
            point1 = Point(points[i]['lat'], points[i]['lng'])
            point2 = Point(points[j]['lat'], points[j]['lng'])
            distance = calculate_distance(point1, point2)
            if distance <= distance_threshold:
                G.add_edge(i, j, weight=distance)

    isolated_nodes = [node for node in G.nodes() if G.degree(node) == 0]

    new_points = []
    for i in isolated_nodes:
        for j in range(len(points)):
            if i != j:
                point1 = Point(points[i]['lat'], points[i]['lng'])
                point2 = Point(points[j]['lat'], points[j]['lng'])
                distance = calculate_distance(point1, point2)
                if min_midpoint_distance_threshold <= distance <= max_midpoint_distance_threshold:
                    point1 = Point(points[i]['lat'], points[i]['lng'])
                    point2 = Point(points[j]['lat'], points[j]['lng'])
                    mid_point = midpoint(point1, point2)
                    new_points.append({"type": "mid-point", "lat": mid_point.latitude, "lng": mid_point.longitude})
                    new_node_index = len(points) + len(new_points) - 1
                    G.add_node(new_node_index, pos=(mid_point.longitude, mid_point.latitude), type='mid-point')
                    node_colors.append('green')
                    half_distance = distance / 2
                    G.add_edge(i, new_node_index, weight=half_distance)
                    G.add_edge(new_node_index, j, weight=half_distance)
```

Рисунок 4.26 – Код для обчислення графів ч.1

```python
                    G.add_edge(new_node_index, j, weight=half_distance)
    pos = nx.get_node_attributes(G, 'pos')
    labels = nx.get_edge_attributes(G, 'weight')
    nx.draw(G, pos, with_labels=True, node_color=node_colors, node_size=1500)
    nx.draw_networkx_edge_labels(G, pos, edge_labels={k: f"{v:.2f} км" for k, v in labels.items()})

    save_optimal_routes_to_file(G, node_colors,'optimal_routes.txt')

    save_graph_to_json(G, 'graph_with_midpoints.json')

    plt.title("Графове зображення з новими точками")
    plt.savefig("graph_with_midpoints_3.png", format="png")

    all_routes = find_all_routes(G)
    for route, length in all_routes:
        print(f"Маршрут: {route}, Довжина: {length:.2f} км")

    print("Граф з ретрансляторами з згенеровано!")
```

Рисунок 4.27 - Код для обчислення графів ч.2

Функція create_graph_with_midpoints виконує складний аналіз та візуалізацію мережі, використовуючи графи для моделювання з'єднань між точками, з урахуванням додавання оптимальних місць розташування ретрансляторів. Функція приймає чотири параметри: json_data (рядок JSON з початковими точками), distance_threshold (максимальна допустима відстань між точками для безпосереднього з'єднання), min_midpoint_distance_threshold і max_midpoint_distance_threshold (мінімальна та максимальна відстані для визначення місця ретранслятора).

Спочатку, дані з json_data конвертуються в Python-об'єкти, після чого створюється граф G за допомогою бібліотеки NetworkX. Для кожної точки в даних створюється вузол графа. Потім функція перебирає всі пари точок, розраховуючи відстань між ними. Якщо ця відстань менша або дорівнює distance_threshold, між вузлами створюється зв'язок. Особливість цієї функції полягає в обробці ізольованих вузлів, тобто таких, що не мають прямих зв'язків із іншими. Для кожного ізольованого вузла функція шукає точки, відстань до яких знаходиться у межах заданих порогів min_midpoint_distance_threshold і max_midpoint_distance_threshold. Для відповідних пар точок розраховується середня точка (ретранслятор), яка додається до графа.

Після цього відбувається візуалізація створеного графа з використанням NetworkX, де точки та ретранслятори мають різні кольори. Граф зберігається у форматі JSON та у вигляді зображення. Також функція знаходить усі можливі маршрути в графі, розраховує їхню довжину та виводить цю інформацію. Функція завершується повідомленням про успішне генерування графа з ретрансляторами. Для визову функції виконується команда create_graph_with_midpoints(filtered_json_data, distance_threshold, min_midpoint_distance_threshold,max_midpoint_distance_threshold), що додається до функції збереження конфігурації в файл. Такі дії дозволять автоматично проводити розрахунки, будувати графи мережі та обчислювати оптимальні маршрути. Результати обчислення графу з ретрансяторами наведено на рис. 4.28.

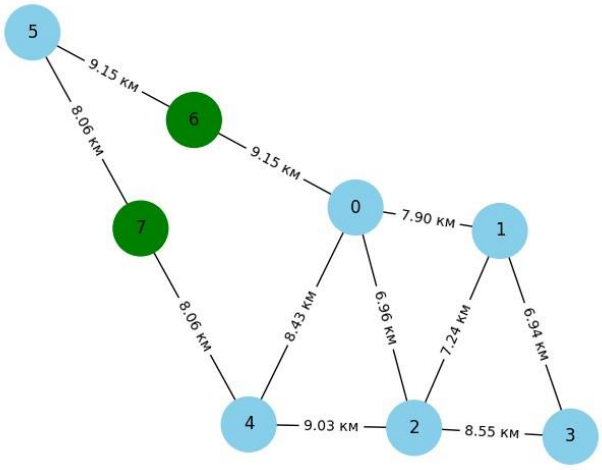


Рисунок 4.28 – Графова задача з ретрансляторами

Як можна побачити з рис. 4.28, вузли ретранслятори позначені зеленим кольором, а кінцеві пристрої ОЕС – синім. Алгоритм запропонував додати два вузли посередника – це точки 6 та 7. Завдяки ним з'являється можливість об'єднати усі вузли в безпроводової сенсорної мережі до одного кластеру і збільшити масштаб охоплення. Вже раніше було зазначено, що після виконання обчислень формується оновлений JSON файл конфігурації мережі. Код функції наведено на рис. 4.29.

```
def save_graph_to_json(G, file_name):
    graph_data = []
    for node_id, node_data in G.nodes(data=True):
        coordinates = node_data.get('pos', (None, None))
        node_type = node_data.get('type', 'unknown')
        graph_data.append({
            'id': node_id,
            'type': node_type,
            'lat': coordinates[1],
            'lng': coordinates[0]
        })

    with open(file_name, 'w') as file:
        json.dump(graph_data, file, indent=4)
```

Рисунок 4.29 – Функція збереження графу у форматі JSON

Функція save_graph_to_json призначена для серіалізації графа G у формат JSON та його збереження у файл. Граф G передається як вхідний параметр разом із назвою файлу file_name, куди має бути збережений граф. Функція ітерує крізь усі вузли графа, збираючи дані про кожен вузол, такі як його унікальний ідентифікатор (ID), тип вузла та його координати (широту та довготу). Ця інформація зберігається у списку graph_data у форматі словників Python. Після збору інформації про всі вузли, виконується серіалізація цього списку в JSON за

допомогою функції json.dump, і дані записуються у файл з назвою, заданою у file_name. Використання контекстного менеджера with забезпечує коректне відкриття та закриття файлу під час процесу запису. Результат формування нового файлу наведено на рис. 4.30.

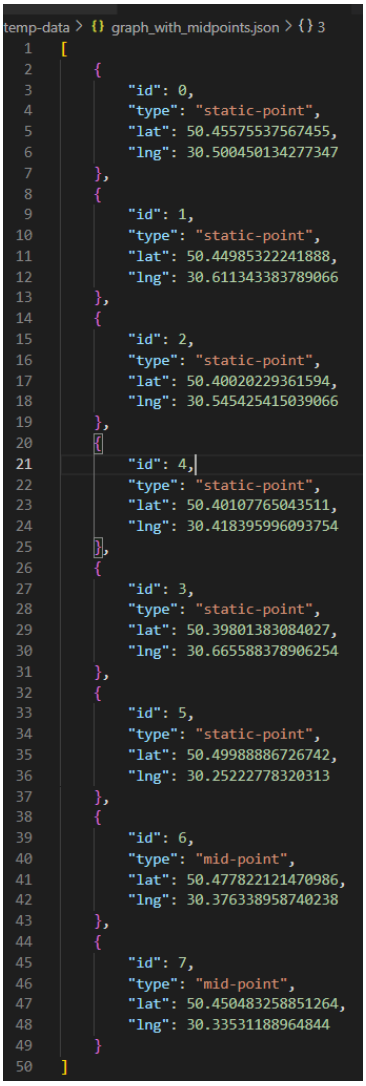


Рисунок 4.30 – Згенерований файл з даними мережі

Після всіх обчислень генерується файл з назвою optimal_routes.txt, до якого записуються усі наявні маршрути з кінцевих пристроїв до інших вузлів мережі. Маршрути від ретрансляторів та до ретрансляторів ігноруються, але переприймои через них враховуються. Оскільки ретранслятор неповноціний пристрій на відміну від ОЕС, то рахувати маршрути тільки до нього не є ефективним. Обчислені маршрути для топології з рис. 4.28 наведено на рис 4.31.

```
Маршрут від 0 до 1: [0, 1], Відстань: 7.90 км
Маршрут від 0 до 2: [0, 2], Відстань: 6.96 км
Маршрут від 0 до 4: [0, 4], Відстань: 8.43 км
Маршрут від 0 до 3: [0, 1, 3], Відстань: 14.84 км
Маршрут від 0 до 5: [0, 6, 5], Відстань: 18.29 км
Маршрут від 1 до 0: [1, 0], Відстань: 7.90 км
Маршрут від 1 до 2: [1, 2], Відстань: 7.24 км
Маршрут від 1 до 4: [1, 2, 4], Відстань: 16.27 км
Маршрут від 1 до 3: [1, 3], Відстань: 6.94 км
Маршрут від 1 до 5: [1, 0, 6, 5], Відстань: 26.19 км
Маршрут від 2 до 0: [2, 0], Відстань: 6.96 км
Маршрут від 2 до 1: [2, 1], Відстань: 7.24 км
Маршрут від 2 до 4: [2, 4], Відстань: 9.03 км
Маршрут від 2 до 3: [2, 3], Відстань: 8.55 км
Маршрут від 2 до 5: [2, 4, 7, 5], Відстань: 25.16 км
Маршрут від 4 до 0: [4, 0], Відстань: 8.43 км
Маршрут від 4 до 1: [4, 2, 1], Відстань: 16.27 км
Маршрут від 4 до 2: [4, 2], Відстань: 9.03 км
Маршрут від 4 до 3: [4, 2, 3], Відстань: 17.58 км
Маршрут від 4 до 5: [4, 7, 5], Відстань: 16.13 км
Маршрут від 3 до 0: [3, 1, 0], Відстань: 14.84 км
Маршрут від 3 до 1: [3, 1], Відстань: 6.94 км
Маршрут від 3 до 2: [3, 2], Відстань: 8.55 км
Маршрут від 3 до 4: [3, 2, 4], Відстань: 17.58 км
Маршрут від 3 до 5: [3, 1, 0, 6, 5], Відстань: 33.13 км
Маршрут від 5 до 0: [5, 6, 0], Відстань: 18.29 км
Маршрут від 5 до 1: [5, 6, 0, 1], Відстань: 26.19 км
Маршрут від 5 до 2: [5, 7, 4, 2], Відстань: 25.16 км
Маршрут від 5 до 4: [5, 7, 4], Відстань: 16.13 км
Маршрут від 5 до 3: [5, 6, 0, 1, 3], Відстань: 33.13 км
```

Рисунок 4.31 – Прораховані оптимальні маршрути

Таким чином, розроблене програмне забезпечення забезпечує повний цикл проектування безпроводої сенсорної мережі оптико-електронних станції. Веб-застосунок дозволяє візуалізувати розташування пристроїв спостереження, командних вузлів та ретрансляторів. Завдяки функціям збереження та завантаження, можливо поширювати конфігурацію мережі та редагувати її в ручному режимі. Друга частина програмного забезпечення дозволяє вирішити задачу маршрутизації, візуалізвуати топологію, прорахувати усі наявні оптимальні за критерієм дальності маршрути та запропонувати розташування вузлів ретрансляторів в необхідних позиціях для забезпечення ефективного функціонування топології та усіх наявних мережевих пристроїв.

ВИСНОВКИ

Результатом проведення досліджень та виконання кваліфікаційної роботи є вирішення задачі по розробці методу масштабування безпроводової сенсорної мережі оптико-електронних станцій із застосування технології LoRa.

В першому розділі кваліфікаційної роботи для виконання поставлених задач було проведено порівняльний аналіз технологій організації безпроводових сенсорних мереж ZigBee, Sigfox та Lora. За результатами аналізу та порівняння за швидкістю, далекобійністю, завадостійкістю та іншими показниками можна виділити повну перевагу технології LoRa над іншими технологіями і можна зробити висновок, що технологія LoRa відповідає усім поставленим. Але для повноцінного функціонування мережі з ОЕС було використано протокол комутації LoRaWAN з модифікаціями для забезпечення умов кластеризації, а також підготовлено алгоритми формування кластерів, алгоритм управління девайсами у комірках кластерів. Для планування масштабування мережі було розглянуто основні топології з усіма перевагами та недоліками, загальну класифікацію мереж за масштабами, а також сформовано вимоги до безпроводової сенсорної мережі.

Для організації безпроводової сенсорної мережі було обрано модулі зв’язку Semtech SX1262 з підтримкою технологій LoRa. В другому розділі кваліфікаційної роботи було досліджено технологію LoRa і особливості модуляції з розкладанням сигналу на частини. Модуляція LoRa дозволяє мережевим пристроям працювати у середмісті не зважаючи на інші сигнали, перешкоди, завади та погодні умови. Було виконано порівняння модулів SX1262 та SX1278. Модуль SX1262 має кращу енергоефективність, ширший діапазон частот, який забезпечує більшу гнучкість та адаптивність. Також слід зазначити, що він обладнаний USB роз’ємом, що збільшує універсальність застосування з різними модифікаціями мікроконтролерів. Для виконання поставлених задач квалифікційної роботи та виконання масштабування рекомендовано використовувати режим широкомовної розсилки при ініціалізації мережі. Пристрої повинні надавати інформацію про себе до наближених мережевих модулів для заповнення маршрутних таблиць серед інших модулів зв’язку. Після встановлення з’єднання пристрої перейдуть у режим фіксованої розсилки, що

дозволить виконувати задачі комутації даних. Для виконання задач виокремлення мережевих пакетів поміж інших було створено окремий клас комутаційних пакетів. Протокол LoRaWAN рекомендується використовувати в якості протоколу високого рівня для організації безпроводової сенсорної мережі оптико-електронних станций.

В третьому розділі кваліфікаційної роботи було розглянуто можливість організації кластерів у безпроводових мережах ОЕС. На рисунку 3.1 розглянуто топологію однорангової мережі з декількома кластерами, а на рисунку 3.7 зображено приклад задачі маршрутизації при організації багаторангової мережі. Для виконання задачі кластеризації було розроблено алгоритм, який наведено на рисунку 3.9. Тип організації мережі має лінійну залежність від відстані, тобто якщо відстань між вузлами 40% від максимального радіуса дії – то можливо організувати комірку кластеру з декількома пристроями, з яких тільки один буде мати зв’язок з іншою коміркою кластеру. Організація мережі таким чином допоможе отримати кращу енергоефективність.

Під час виконання кваліфікаційної роботи було розроблено інструмент планування мережі у вигляді веб-застосунку, який дозволяє спланувати та спрогнозувати навантаження на елементи безпроводової мережі оптико-електронних станції, а також завчасно прорахувати зону покриття та слабкі місця. В четвертому розділі представлено алгоритм ініціалізації мережі (рис. 4.1) та алгоритм обробки пакетів і маршрутизації (рис. 4.2). Завдяки реалізованим алгоритмам та програмному забезпеченню вирішується питання масштабування мережі LoRa та об’єднання комірок кластерів у суцільну єдину мережу. Завдяки програмам розрахунку одразу можливо дізнатись точки навантаження, прорахувати швидкість мережі, знайти позиції для ретрансляторів та спланувати наперед механізми резервування. Розроблений веб-застосунок дозволяє візуалізувати розташування ОЕС на карті, побудувати графову модель мережі і в автоматичному порядку розрахувати позиції для розташування вузлів ретрансляторів. Також для організований мережі автоматично будуються оптимальні маршрути за алгоритмом Дейкстри.

Таким чином, під час виконання кваліфікаційної роботи було досліджено технічні особливості модулів зв’язку та метод масштабування безпроводової сенсорної мережі з використанням цих модулів. Було розроблено алгоритми та програмне забезпечення для розрахунку топології мережі з автоматичним позиціонуванням ретрансляторів між кластерами у мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Шлома О. К., Шостко І.С. Алгоритм дистанційного керування приводами лазерної оптико-електронної станції по can-шині. *Перспективи розвитку інфокомунікацій та інформаційно-вимірювальних технологій*. – 2020. – №4. – С. 36–37.

2. Шлома О. К., Волотка В.С. Огляд інновації в області передачі даних на прикладі протоколів зв'язку. *Стан, досягнення і перспективи інформаційних систем і технологій*. – 2020. – №2. – С. 103–104.

3. Шлома О.К., Шостко І.С., Майба М.А., Дробяз М.О Масштабування мережі оптико-електронних станції із застосуванням технології lora. *Проблеми електромагнітної сумісності перспективних бепроводових мереж зв'язку (ЕМС-2022)»*. тези доп. Міжнар. наук.-практ Харків: ХНУРЕ – 2022. С. 90-93.

4. Шлома О.К., Волотка В.С., Стамбулжі М.М. Індустрія 5.0-найближче майбутнє? .*Сучасний рух науки*: тези доп. XII Міжнар. наук.-практ. інтернет-конференції. – 2021.–Т. 1.–493 с

5. Закон України «Про захист інформації в інформаційно-\ телекомунікаційних системах» URL: – 1994. – Режим доступу до ресурсу: http://zakon5.rada.gov.ua/laws/show/80/94-%D0%B2%D1%80 (дата звернення 10.12.2023).

6. Шостко І.С.,Тевяшев А.Д., Землянной О.В. Сетевая оптико-электронная система мониторинга воздушного пространства *Transfer of Innovative Technologies*. – 2020. – №4. – С.106-107.

7. Шостко І.С., Шлома О.К., Цибільников Д.І. Метод налаштування й управління режимами роботи безпроводової сенсорної мережі оптико-електронних станцій із застосуванням технології LORA *Проблеми телекомунікацій*, 2022, No. 1(30), С. 32–56

8. Шостко І.С.,Тевяшев А.Д., Кудя Ю.Є., Методы позиционирования узлов беспроводной сенсорной сети. *Проблеми телекомунікацій*, 2021.–Т. 1.–С.68-69.

9. Shostko I.S. Applications of Multipath Routing for Energy Balancing in Sensor Networks. *Scientific and Technical Journal «radioelectronics & informatics»*. – 2015. – Р1 №1. – 2015. – Р. 12-16.

10. Shloma O. K. Analysis of Innovation in the Field of Data Transfer on Example of Segment Routing. *computer and information systems and technologie*s. – 2020.

11. Shloma O.K., Shostko I.S., Tsybulnikov D.M. The Method of Controlling The Wireless Sensor Network of Optical-Electronic Stations Using LoRa Technology. *IEEE 9th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T),* 2022. P. 583 - 586

12. SX1278 Wireless Module E32 Series URL: – 2020. – Режим доступу до ресурсу: https://ebyte.com (дата звернення 21.12.2023).

13. E32 LORA tranceiver module URL: – 2020. – Режим доступу до ресурсу: https://www.matrixtsl.com (дата звернення 21.12.2023).

14. Arduino UNO documentation URL: Режим доступу: https://doc.arduino.ua/ru/hardware/UNO (дата звернення 21.12.2023).

15. LoRa Modulation Basics AN1200.22 / URL: – 2020. – Режим доступу до ресурсу: https://semtech.com (дата звернення 21.12.2023).