МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет

імені Петра Могили

Факультет комп'ютерних наук

Кафедра комп'ютерної інженерії

ДОПУЩЕНО ДО ЗАХИСТУ

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

_____ І. М. Журавська

« __ » _____ 2023 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі

Спеціальність 123 Комп'ютерна інженерія

123 – КБР.ПЗ.00 – 405.21910509

Студент _____М. О. Жуланов
підпис
«__» _____ 202__ р.

Керівник д-р техн. наук, проф. _____ С. В. Пузирьов
підпис
«__» _____ 202__ р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили
Факультет комп'ютерних наук
Кафедра комп'ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри _____ І. М. Журавська

«____»_____20_ р.

ЗАВДАННЯ
на виконання кваліфікаційної бакалаврської роботи

Видано студенту групи 405 факультету комп'ютерних наук

_____Жуланову Максиму Олеговичу_____
(прізвище, ім'я, по-батькові студента)

1. Тема кваліфікаційної роботи: Програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі

Затверджена наказом по ЧНУ від «__» _____ 2023 р. № _____

2. Строк представлення кваліфікаційної роботи «____» _____ 2023 р.
3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Комплекс на базі mesh-мережі, що забезпечує надійний голосовий зв'язок на відстані до 200м та не вимагає централізованого управління.

4. Перелік питань, що підлягають розробці

Дослідження основних відомостей та принципів побудови mesh-мережі. Реалізація апаратної частини mesh-мережі. Реалізація програмної частини mesh-мережі. Аналіз отриманих результатів. Дослідження перспектив розвитку mesh-мереж. Спеціальна частина з охорони праці. Висновки. Перелік джерел посилання.

5. Перелік графічних матеріалів

– зображення компонентів апаратної частини комплексу, позначення виходів, загальні/структурні схеми;

– логічна схема модему LoRa, вміст пакету, блок-схеми алгоритму передачі/прийому пакету;

– часова діаграма сигналів інтерфейсів;

– візуалізація Mesh-мережі.

6. Завдання до спеціальної частини

Розглянути основні державні норми України, щодо праці в умовах використання комп'ютерів та екраних пристроїв загалом, щодо вентиляції та кондиціонування, організація повітрообміну, норм шумів та вібрацій. Ознайомитись з правами та нормами праці в умовах використання комп'ютерів та екраних пристроїв загалом.

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
А. О. Алексєєва, к.т.н., доцент	кафедра екології Медичного інституту ЧНУ імені Петра Могили	Спеціальна частина з охорони праці

Керівник роботи
_ канд. фіз.-мат. наук, доцент, С. В. Пузирьов_____
(посада, прізвище, ім'я, по батькові)

_____
(підпис)

Завдання прийнято до виконання

_ Жуланов Максим Олегович_____
(прізвище, ім'я, по батькові студента)

_____
(підпис)

Дата видачі завдання «____» _____ 20_____ р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної бакалаврської роботи

Тема: Програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КБР	20.12.2022	26.12.2022	Виконано
2	Огляд літератури за темою роботи	15.01.2023	01.02.2023	Виконано
3	Складання календарного плану КБР	02.02.2023	03.02.2023	Виконано
4	Аналіз предметної області	03.02.2023	09.02.2023	Виконано
5	Розробка проектних рішень	10.02.2023	10.03.2023	Виконано
6	Моделювання апаратної частини	11.03.2023	11.04.2023	Виконано
7	Розробка програмного забезпечення та тестування	12.04.2023	12.05.2023	Виконано
8	Написання розділу з охорони праці	13.05.2023	19.05.2023	Виконано
9	Відгук керівника КБР	19.05.2023	20.05.2023	Виконано
10	Оформлення КР та презентації	20.05.2023	30.05.2023	Виконано
11	Перший передзахист КБР	30.05.2023	31.05.2023	Виконано
12	Рецензування	31.05.2023	13.06.2023	Виконано
13	Другий передзахист КБР	13.06.2023	14.06.2023	Виконано
14	Завершення оформлення КБР та презентації	14.06.2023	27.06.2023	Виконано
15	Захист кваліфікаційної роботи	27.06.2023	28.06.2023	Виконано

Розробив здобувач ВО Жуланов Максим Олегович _____ 

(*прізвище, ім'я, по батькові)* *(підпис)*

«__» _____ 20__ р.

Керівник роботи канд. фіз.-мат. наук, доцент, С. В. Пузирьов____ _____

(підпис)

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі»

Студент 405 гр.: Жуланов Максим Олегович

Керівник: канд. фіз.-мат. наук, доцент, С. В. Пузирьов

Бакалаврська робота присвячена розробці програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі. Розглянуто наявні рішення для голосового зв’язку та порядок їх побудови. Практичне значення результатів дослідження та розроблення полягає в тому що, на підставі результатів можливе створення приладу, вдосконалення програмно-апаратного комплексу, засвоєння навичок процесу проектування, застосування та подальше використання результатів роботи у майбутніх дослідженнях розробленого комплексу.

Пояснювальна записка бакалаврської роботи складається зі вступу, трьох розділів, висновків та додатків. У вступі визначається актуальність теми, сформульовані мета, об’єкт, предмет, завдання дослідження та розроблення бакалаврської роботи. У першому розділі проводиться аналіз існуючих рішень, дослідження основних відомостей, принципів побудови mesh-мереж та обираються необхідні компоненти. У другому розділі розглянуто вибір та реалізацію апаратної частини мережі. У третьому розділі описано хід розробки програмного забезпечення для комплексу. У висновках наведено аналіз виконаної роботи та отриманих результатів дослідження.

У додатку А наведено звіт з перевірки роботи на плагіат. У додатку Б наведено програмний код, що використовувався в проекті. В цілому, бакалаврська робота без додатків містить 85 сторінок, 31 рисунок, 4 таблиці, 20 джерел посилань.

Ключові слова: mesh-мережа, протокол маршрутизації, периферійний інтерфейс, мікроконтролер, трансивер.

ABSTRACT

of the Bachelor’s Thesis

«Software and hardware complex for voice connection based on a mesh-network»

Student: Zhulanov Maksym Olegovich

Supervisor: PhD in Physics and Mathematics, associate professor, S.V. Puzyrov

The bachelor's thesis is devoted to the development of software and hardware complex for voice communication based on a mesh-network. The thesis examines the available solutions for voice communication and the order of their construction. The practical value of the results of research and development lies in the fact that, based on the results, it is possible to create a device, improve the hardware and software complex , master the skills of the design and application process, and use the results of work in future studies of the developed complex.

The explanatory note of the bachelor thesis consists of an introduction, main body(three chapters), conclusions and appendixes. The introduction determines the relevance of the topic, formulates the goal, the object, the subject, the research and development task of the bachelor's thesis. In the first chapter of the main body, an analysis of existing solutions is carried out, the basic information and the principles of mesh-network construction are explored, and the necessary components are selected. The second chapter of the main body deals with the selection and implementation of the hardware part of the network. The third chapter of the main body describes the software development process for the complex. The conclusions provide the analysis of the performed work and the obtained research results.

Appendix A contains a plagiarism check repor. Appendix B contains the program code used in the project. In general, the bachelor's thesis without appendixes contains 85 pages, 31 illustrations, 4 tables, 20 reference sources.

Keywords: mesh network, routing protocol, peripheral interface, microcontroller, transceiver.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КПК – Кишеньковий Персональний Комп'ютер
МНС – Міністерство Надзвичайних Ситуацій
СКЖ – Система Контролю Живлення

AODV – Ad hoc On-Demand Distance Vector
ASCII – American Standard Code for Information Interchange
BATMAN – Better Approach to Mobile Ad-hoc Networking
CRC – Cyclic redundancy check
CSS – Cascading Style Sheets
DDNS – Dynamic DNS
DHCP – Dynamic Host Configuration Protocol
DMZ – Demilitarized Zone
DNS – Domain Name System
DNS – Domain Name System
DoS/DDoS – Distributed Denial of Service
DSR – Dynamic Source Routing
EEPROM – Electrically Erasable Programmable Read-Only Memory
FIFO – First In First Out
FSK – Frequency Shift Keying
FTDI – Future Technology Devices International
FTP – File Transfer Protocol
GPS – Global Positioning System
HTML – HyperText Markup Language
I²C – Inter-Integrated Circuit
IDE – Integrated Development Environment
IEEE – Institute of Electrical and Electronics Engineers
IoT – Internet of Things
ISP – In-System Programming

JTAG – Joint Test Action Group

LCD – Liquid Crystal Display

LoRaWAN– Long Range Wide Area Network

MISO – Master In Slave Out

MOSI – Master Out Slave In

MQTT – Message Queue Telemetry Transport

OLSR – Optimized Link State Routing Protocol

OOK – On-Off Keying

PDA – Personal Digital Assistant

PSTN – Public Switched Telephone Network

RADIUS – Remote Authentication Dial-In User Service

RAM – Random Access Memory

SCL – Serial Clock

SDA – Serial Data

SPI – Serial Peripheral Interface

SPIFFS – Serial Peripheral Interface Flash File System

SS – Slave Select

SSG **–** Secure Service Gateway

SSID – Service Set Identifier

TCP – Transmission Control Protocol

UART – Universal Asynchronous Receiver-Transmitter

USB – Universal Serial Bus

VoIP – Voice over Internet Protocol

VPN – Virtual Private Network

Wi-Fi – Wireless Fidelity

WMN – Wireless Mesh Network

WPA2-PSK– Wi-Fi Protected Access 2 with Pre-Shared Key

ВСТУП

У сучасному світі, що стрімко розвивається, постійно зростає потреба в ефективних засобах комунікації, особливо в сфері голосового зв'язку. Завдяки новітнім технологіям та інноваціям, комунікаційні системи поступово стають все більш надійними, доступними та зручними у використанні. Одним із перспективних напрямків розвитку голосового зв'язку є впровадження програмно-апаратних комплексів на базі mesh-мереж.

Mesh-мережі – це безпровідні, розподілені мережі, які забезпечують надійне та стабільне з'єднання між пристроями, незалежно від інфраструктури традиційних телекомунікаційних мереж. Відмінність таких мереж полягає в тому, що вони працюють на принципі взаємодії пристроїв один з одним, уникаючи єдиних центральних вузлів або станцій. Це дозволяє підвищити стабільність роботи мережі, покращити якість зв'язку та забезпечити більш гнучке масштабування.

На сьогоднішній день існує ряд наукових робіт та практичних розробок, присвячених використанню mesh-мереж у різних сферах діяльності. Проте, дослідження програмно-апаратних комплексів для голосового зв'язку на базі цих мереж залишається актуальним завданням, оскільки ще не вирішено ряд проблем, пов'язаних з якістю передачі голосу, оптимізацією маршрутизації та енергоефективністю пристроїв.

У рамках даної роботи розглядається програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі, який включає аналіз поточного стану розвитку технологій, виявлення проблем та прогалин у знаннях та розробку нових методів та рішень для покращення якості голосового зв'язку.

Мета: Розробка програмно-апаратного комплексу для голосового зв'язку на основі mesh-мережі, який забезпечує високу якість передачі голосу, оптимальну маршрутизацію пакетів даних та енергоефективність використання пристроїв.

Об'єкт: Програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі, що охоплює апаратні та програмні складові, їх взаємодію та інтеграцію в робочій системі.

Предмет: Методи, алгоритми, технології та моделі, що використовуються для покращення ефективності, якості та енергоефективності програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі. Це включає аналіз технічних характеристик мережі, оптимізацію маршрутизації пакетів даних, використання сучасних кодеків та алгоритмів стиснення, розробку енергоефективних рішень для пристроїв мережі та інше.

Для досягнення поставленої мети в роботі ставляться такі основні **завдання**:

– аналіз основних відомостей та принципів побудови mesh-мереж, включаючи технічні характеристики, елементи мережі, особливості та функції мережі Wi-Fi Mesh, вибір протоколу маршрутизації та аналіз існуючих рішень;

– реалізація апаратної частини mesh-мережі, включаючи вибір платформ для побудови мережі, опис платформи для мережевого шлюзу з контролером, використання трансивера RFM95 як основи мережі, опис Wi-Fi модуля ESP8266 та використання LCD-дисплею;

– реалізація програмної частини mesh-мережі, яка включає розробку програмної частини для мережевого шлюзу, налаштування послідовного периферійного інтерфейсу, роботу з EEPROM пам'яттю, керування трансивером RFM95, використання ESP8266 як основи для mesh-шлюзу, роботу з протоколом MQTT, підключення шлюзу до WMN, аналіз отриманого результату та перспективи розвитку mesh-мереж.

Враховуючи особливості кваліфікаційної роботи, можна виділити наступні методи дослідження:

– спостереження. Дослідження існуючих теоретичних матеріалів, дослідження та аналіз функціонування сучасних mesh-мереж і систем голосового зв'язку;

– аналіз. Систематичний розбір технічних характеристик, особливостей мережі, протоколів маршрутизації, платформ та компонентів програмно-апаратного комплексу;

– декомпозиція. Розчленування програмно-апаратного комплексу на окремі складові, з метою їх детального вивчення та оптимізації;

– агрегація. Інтеграція різних складових програмно-апаратного комплексу в єдину систему та перевірка її функціональності та продуктивності;

– моделювання. Створення математичних, програмних та інженерних моделей для оцінки роботи системи, емуляція роботи програмно-апаратного комплексу в різних умовах та експериментальне перевірка результатів.

Новизна дослідження полягає в наступних аспектах:

– розробка програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі, який є ефективним і надійним рішенням у порівнянні з традиційними системами зв'язку;

– впровадження нових алгоритмів і методів маршрутизації, що забезпечують оптимізацію передачі даних та підвищення стабільності роботи mesh-мережі;

– розробка енергоефективних рішень для пристроїв мережі, що знижують енергоспоживання та забезпечують тривалу автономну роботу системи;

– створення модульної архітектури програмно-апаратного комплексу, що дозволяє легко налаштовувати, розширювати та модифікувати систему відповідно до потреб користувачів.

1 ОСНОВНІ ВІДОМОСТІ ТА ПРИНЦИПИ ПОБУДОВИ MESH-МЕРЕЖ

1.1 Загальні відомості про Mesh-мережі

Mesh-мережі – це новітній тип бездротових мереж, який відрізняється від традиційних зіркових топологій за своєю структурою та робочими принципами [1]. Вони характеризуються динамічним підключенням вузлів, автоматичною маршрутизацією трафіку та високою стійкістю до збоїв. У даному розділі ми розглянемо основні аспекти, що стосуються mesh-мереж, їхні переваги та недоліки, а також можливості їх застосування. Така мережа складається з вузлів, які можуть виконувати різні функції: передавати, отримувати або маршрутизувати дані. Вони можуть працювати як точки доступу, клієнти або ретранслятори. Особливість mesh-мережі полягає в тому, що вона є самоналаштовуваною, тобто вона автоматично визначає найкращі маршрути для передачі даних між вузлами.

Топологія WMN (Wireless Mesh Network) відрізняється від типових мереж 802.11a/b/g, оскільки базується на децентралізованій організації мережі (рис. 1.1).

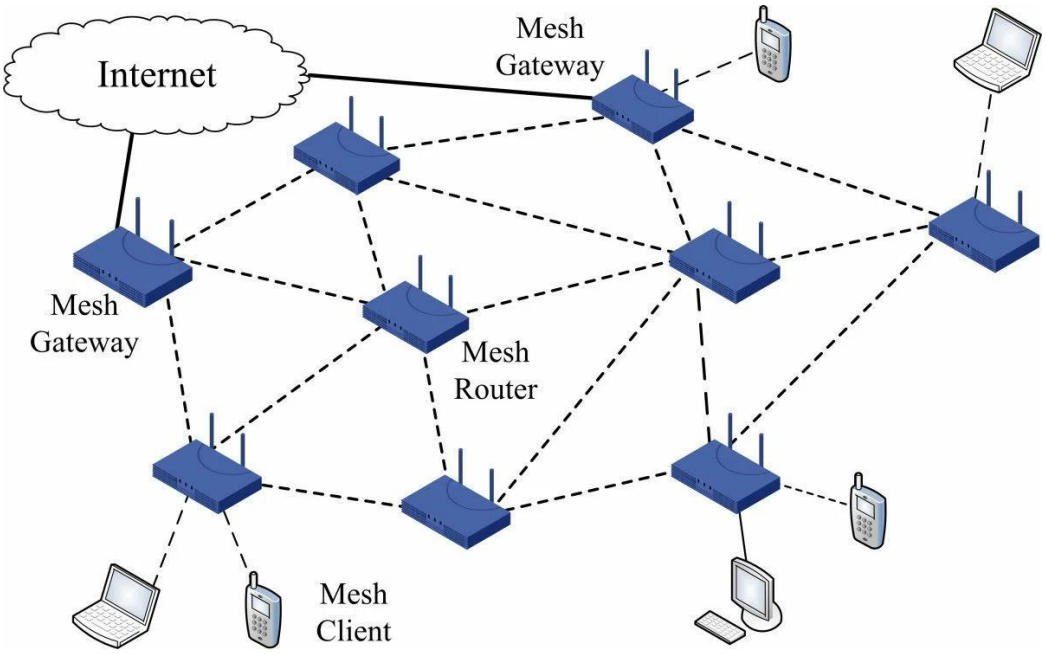


Рисунок 1.1 – Топологія Mesh-мережі

За розподіленим принципом, вона надає можливість створювати багаторівневі мережі, в яких кожен вузол має функції маршрутизатора і може використовувати різні шляхи для передачі пакетів. Це дозволяє створювати самовстановлювальні та самовідновлювальні сегменти мережі. Якщо один з вузлів відмовляє, інші вузли автоматично знаходять нові шляхи для передачі даних.

WMN можна умовно розділити на дві категорії: стаціонарні та мобільні. Стаціонарні мережі мають постійне розташування вузлів, тоді як мобільні мережі можуть часто змінювати своє місце розташування.

Мережу можна побудувати як сукупність кластерних зон, і кількість таких зон теоретично необмежена. Кластерна організація сприяє полегшенню управління мережею, оскільки можна зосередитися на окремих зонах без необхідності враховувати всю мережу в цілому.

Переваги Mesh-мережі включають наступні аспекти [2]:

– висока стійкість. Mesh-мережі характеризуються високою стійкістю до збоїв завдяки наявності кількох шляхів передачі даних між вузлами. Це дозволяє мережі продовжувати роботу навіть у разі відмови одного або кількох вузлів;

– самоналаштування. Mesh-мережі автоматично визначають найкращі маршрути для передачі даних між вузлами, що спрощує налаштування та розширення мережі;

– масштабованість. Mesh-мережі легко масштабуються, оскільки нові вузли можуть бути додані без зміни загальної конфігурації мережі;

– висока пропускна здатність. Завдяки кільком незалежним маршрутам передачі даних між вузлами, mesh-мережі забезпечують високу пропускну здатність та менші затримки в порівнянні з традиційними мережами.

До недоліків Mesh-мережі можна віднести наступні:

– низька надійність. Мережа може стати менш надійною, якщо багато вузлів буде відсутнім або вимкненим. Це може призвести до порушення зв'язку і недоступності деяких частин мережі;

– низька ефективність в умовах великого обсягу трафіку. Оскільки кожен вузол в Mesh-мережі має функції маршрутизатора, пакети доводиться проходити через декілька вузлів, перш ніж дійдуть до призначення. Це може призвести до затримок і зниження швидкості передачі даних [3];

– вразливість до атак типу «черв'яки». Mesh-мережі можуть стати вразливими до атак типу «черв'яки», оскільки пакети можуть поширюватися в мережі без обмежень. Це може дозволити шкідливим елементам використовувати мережу для шкідливих дій;

– висока складність побудови та управління. Побудова та управління Mesh-мережею може бути складною через необхідність координації багатьох вузлів, настройки маршрутизації та моніторингу трафіку.

1.2 Технічні характеристики мережі

Одним із головних принципів побудови mesh-мережі є принцип самоорганізації архітектури, що забезпечує такі можливості, як реалізацію топології мережі «кожен з кожним»; стійкість мережі при відмові окремих компонентів; масштабованість мережі; динамічну маршрутизацію трафіку; контроль стану мережі та ін. Mesh-технологія стає особливо необхідною за відсутності провідної інфраструктури для з'єднання станцій.

Топологія Mesh ґрунтується на децентралізованій схемі організації мережі, на відміну від типових мереж 802.11 a/b/g, які створюються за централізованим принципом. Точки доступу, що працюють у Mesh-мережах, не лише надають послуги абонентського доступу, але й виконують функції маршрутизаторів/ретрансляторів для інших точок доступу тієї самої мережі.

Завдяки цьому з'являється можливість створення сегмента широкосмугової мережі, що самовстановлюється і самовідновлюється.

Mesh-мережі будуються як сукупність кластерів (рис. 1.2). Територія покриття поділяється на кластерні зони, кількість яких теоретично не обмежена. В одному кластері розміщується від 8 до 16 точок доступу. Одна з таких точок є вузловою (gateway) і підключається до магістрального інформаційного каналу за допомогою кабелю (оптичного або електричного) або радіоканалу (з використанням систем широкосмугового доступу). Вузлові точки доступу, як і інші точки доступу (nodes) в кластері, з'єднуються між собою (з найближчими сусідами) по транспортному радіоканалу. Залежно від конкретного рішення точки доступу, можуть виконувати функції ретранслятора (транспортний канал) або функції ретранслятора та абонентської точки доступу. Особливістю Mesh є використання спеціальних протоколів, що дозволяють кожній точці доступу створювати таблиці абонентів мережі з контролем стану транспортного каналу та підтримкою динамічної маршрутизації трафіку оптимальним маршрутом між сусідніми точками. При відмові будь-якої з них відбувається автоматичне перенаправлення трафіку іншим маршрутом, що гарантує не просто доставку трафіку адресату, а доставку за мінімальний час.

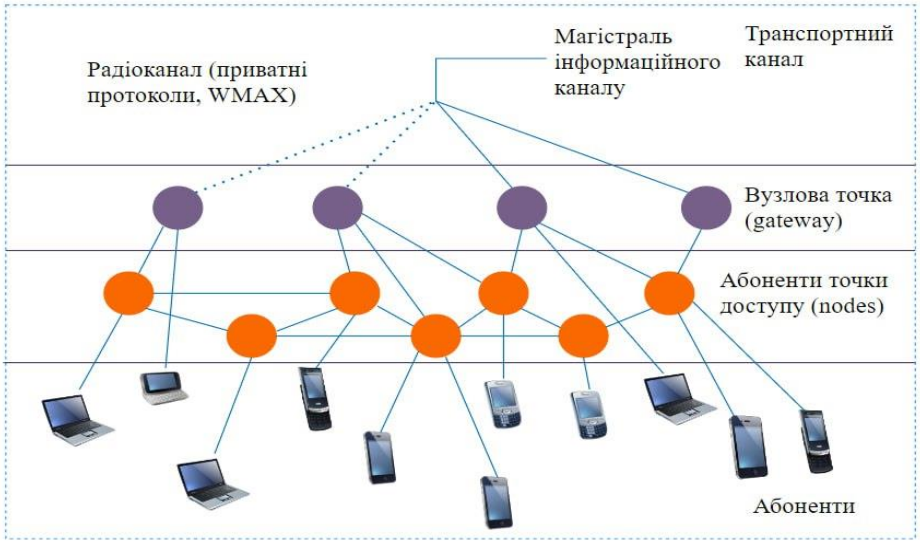


Рисунок 1.2 – Структурна схема Mesh-мережі

Процедура розширення мережі в межах кластера обмежується встановленням нових точок доступу, інтеграція яких у існуючу мережу відбувається автоматично.

Недолік подібних мереж у тому, що вони використовують проміжні пункти передачі даних; це може спричинити затримку під час пересилання інформації та, як наслідок, знизити якість трафіку реального часу (наприклад, мови або відео). У зв'язку з цим є обмеження на кількість точок доступу в одному кластері.

Mesh-топологія дозволяє реалізувати унікальні за своїми можливостями мережі муніципального призначення, орієнтовані служби оперативного реагування (поліція, «Швидка допомога», МНС) [4].

Основу мережі складають вузлові та абонентські точки доступу, що розміщуються на вулиці (як правило, вздовж доріг) та організуючі зони інформаційного покриття, в яких забезпечується підключення абонентів зі стандартними Wi-Fi-адаптерами. Додатково точки доступу можуть використовуватися для організації управління рухом (світлофори) та збору відеоінформації, з підключенням відеокамер по дротовому або бездротовому інтерфейсу. Підключення користувачів, розташованих усередині приміщень, до зовнішньої мережі здійснюється за допомогою внутрішньо-офісних точок доступу, які характеризуються зниженою вихідною потужністю та «кімнатним» виконанням корпусу.

Найбільший інтерес мають мобільні точки доступу, призначені для експлуатації в автомобілях. Використання цих пристроїв не тільки збільшує радіус дії між точками доступу до 800-1200 метрів, а й дозволяє організувати:

– інформаційне забезпечення користувачів усередині автомобіля при дротовому або бездротовому підключенні кінцевих пристроїв (ноутбук, PDA тощо);

– інформаційне покриття у радіусі 300 м навколо автомобіля для абонентів зі стандартними Wi-Fi-адаптерами 802.11 b/g;

– контроль положення автомобіля при використанні вбудованого в точку доступу приймача GPS.

Застосування мобільних точок доступу дозволяє організувати оперативне розширення зони покриття або збільшення інформаційної ємності мережі за рахунок концентрації обладнаних автомобілів у гарячих точках. Механізми самоорганізації Mesh-мережі дозволяють за мінімальний час (визначається часом прибуття автомобілів, обладнаних Mesh-точками доступу) організовувати зону Wi-Fi з передачею оперативної аудіо- та відеоінформації на центральний пульт.

Аналіз створення та розвитку Mesh-мереж показує, що існує стійка тенденція об'єднання абонентських та муніципальних мереж. Найчастіше мережі, побудовані за муніципальним замовленням, доповнюються згодом точками доступу та експлуатуються операторами в об'єднаному «муніципально-абонентському» режимі.

1.3 Елементи мережі

Система Wi-Fi Mesh складається з кількох компонентів, що забезпечують її роботу. Основними компонентами є клієнтська частина, точки доступу, станційна частина та білінгова система провайдера.

Клієнтська частина складається з різних пристроїв, таких як ноутбуки, КПК, смартфони та мобільні телефони, які мають підтримку Wi-Fi технології.

Точки доступу є найважливішою складовою системи Wi-Fi Mesh, які дозволяють підключатися до мережі. Розрізняють RAP-точки доступу з дротовим з'єднанням та MAP-точки доступу з бездротовим з'єднанням. Кожна точка доступу приєднується до дротової мережі через одну або кілька точок доступу з провідним підключенням, так звану RAP.

Станційна частина включає в себе систему управління мережею, комутатор, контролер для управління точками доступу та SSG (Secure Service

Gateway). Ці компоненти забезпечують роботу мережі, моніторинг, налаштування та безпеку.

Існуюча білінгова система провайдера складається з системи авторизації користувачів і елементів мережі (сервер RADIUS), системи розподілу IP адрес користувачів і елементів мережі (сервер DHCP), системи завантаження точок доступу (сервер FTP) та системи доменних імен (сервер DNS). Ці компоненти забезпечують авторизацію та розподіл ресурсів мережі між користувачами [5].

Усі компоненти системи Wi-Fi Mesh повинні працювати відповідно до встановлених стандартів і протоколів, що забезпечують безпеку, швидкість та надійність мережі. Загальна робота цих компонентів дозволяє забезпечити ефективну роботу мережі та забезпечити якість передачі даних.

Крім того, важливим аспектом є технічні характеристики кожного компонента системи. Наприклад, точки доступу повинні мати відповідну швидкість передачі даних, що залежить від їх мережевих параметрів та технології бездротового зв'язку, а також можливість роботи в різних режимах (режим точки доступу, маршрутизатора або моста).

Система управління мережею має бути здатна контролювати роботу всіх компонентів мережі, налаштовувати їх та моніторити стан мережі. Комутатор повинен мати високу пропускну здатність і підтримувати різні стандарти мережевих протоколів.

Контролер для управління точками доступу має бути здатний підтримувати багато точок доступу і здійснювати централізоване управління ними. SSG має забезпечувати захист мережі від різних загроз, зокрема зловмисного програмного забезпечення та атак з зовнішньої мережі.

У системі Wi-Fi Mesh також важливо правильно налаштувати білінгову систему провайдера, яка відповідає за авторизацію та розподіл ресурсів мережі між користувачами. Система авторизації користувачів і елементів мережі (сервер RADIUS) має бути здатна вести журналізацію дій

користувачів, а система розподілу IP адрес користувачів і елементів мережі (сервер DHCP) має бути здатна розподіляти IP адреси відповідно до вимог мережі.

1.4 Особливості та функції мережі WI-FI Mesh

Wi-Fi Mesh є одним з типів mesh-мережі, який базується на бездротовому стандарті Wi-Fi. Він характеризується високою пропускною здатністю, стійкістю та легкістю налаштування.

Канали доступу

Канали доступу – це ресурси мережі Wi-Fi Mesh, які використовуються для передачі даних між точками доступу та клієнтськими пристроями. У даному розділі розглянемо особливості каналів доступу до мережі Wi-Fi Mesh, їх класифікацію та вплив на продуктивність мережі [6].

У мережі Wi-Fi Mesh існують два типи каналів доступу – фізичний та логічний.

Фізичний канал доступу – це частотний діапазон, який використовується для передачі даних між точками доступу та клієнтськими пристроями. Цей діапазон може бути розділений на під діапазони, що використовуються для передачі даних різного типу. Наприклад, діапазон 2.4 ГГц використовується для передачі даних на великі відстані, а діапазон 5 ГГц використовується для передачі даних на короткі відстані та з більшою швидкістю.

Логічний канал доступу – це віртуальний канал, що використовується для передачі даних між точками доступу та клієнтськими пристроями. У мережі Wi-Fi Mesh кожен вузол може мати кілька логічних каналів доступу, що дозволяє одночасно передавати різні види даних.

Безпека Wi-Fi Mesh

Однією з основних проблем, пов'язаних з використанням мереж Wi-Fi, є безпека. З моменту появи Wi-Fi-мереж у 90-х роках минулого століття,

з'явилися численні загрози безпеці мережі, такі як перехоплення даних, злам паролів та інших заходів захисту, зловмисні програми та багато іншого. У цьому розділі ми розглянемо основні загрози безпеці мережі Wi-Fi Mesh та засоби їх усунення.

Загрози безпеці мережі Wi-Fi Mesh [6]:

– перехоплення даних. Є однією з найбільш поширених загроз безпеці мереж Wi-Fi. На даний момент, за допомогою спеціальних програм та обладнання, зловмисники можуть легко перехоплювати даних, що передаються між точками доступу та клієнтськими пристроями;

– атаки «людина посередині». Цей тип атак полягає в тому, що зловмисник перехоплює дані між двома пристроями, вносить зміни в ці дані, та передає їх далі. Найбільш поширеними видами атаки «людина посередині» є атаки Man-in-the-Middle (MITM) та Man-in-the-Browser (MITB);

– несанкціонований доступ. Ця загроза полягає в тому, що зловмисник може отримати доступ до мережі Wi-Fi Mesh без дозволу власника мережі. Це може призвести до крадіжки даних, встановлення шкідливих програм та інших злочинних дій;

– злам паролів. Є однією з найбільш поширених загроз безпеці мереж Wi-Fi. За допомогою спеціальних програм зловмисники можуть перебирати паролі, використовуючи для цього словники та різні комбінації символів.

Відмовостійкість

Відмовостійкість – це властивість мережі, що характеризує її здатність зберігати працездатність в разі відмови одного або декількох її компонентів.

У мережі Wi-Fi Mesh кожна точка доступу має функції маршрутизатора, що дозволяє використовувати різні шляхи для передачі пакетів. Ця особливість відкриває можливість створювати самовстановлювальні та самовідновлювальні сегменти мережі.

Основні принципи відмовостійкості в мережі Wi-Fi Mesh включають [7]:

– взаємозамінність компонентів. Для забезпечення відмовостійкості, мережа повинна мати достатню кількість компонентів, які можуть виконувати ті ж функції. Наприклад, в мережі Wi-Fi Mesh встановлення декількох точок доступу забезпечує забезпечує резервне підключення в разі відмови однієї з точок доступу;

– система резервування. Для забезпечення надійності роботи, мережа повинна мати систему резервування, яка дозволяє забезпечити перехід на резервні компоненти в разі відмови основних. Наприклад, в мережі Wi-Fi Mesh може бути використана система автоматичного відновлення підключення до точок доступу;

– моніторинг та діагностика. Моніторинг та діагностика мережі дозволяє вчасно виявляти проблеми та вирішувати їх. В мережі Wi-Fi Mesh для моніторингу можуть використовуватися системи, які автоматично виявляють несправність компонентів мережі та забезпечують повідомлення про них адміністратору мережі;

– система автоматичного відновлення. В мережі Wi-Fi Mesh можуть бути використані системи автоматичного відновлення підключення до точок доступу. Якщо одна з точок доступу відмовляє, система може автоматично переключити підключення на іншу точку доступу;

– захист мережі від атак. Для забезпечення відмовостійкості мережі Wi-Fi Mesh важливо забезпечити її захист від різних типів атак, таких як DoS (Denial of Service), DDoS (Distributed Denial of Service), а також захист від хакерських атак на точки доступу. Для цього можуть бути використані різноманітні захисти, такі як WPA2-PSK (Wi-Fi Protected Access 2 with Pre-Shared Key), фільтрація мак-адрес, VPN-з'єднання, файрволи та інші;

– швидке відновлення підключення. Для забезпечення швидкого відновлення роботи мережі після відмови компонента, мережа повинна мати

систему автоматичного відновлення підключення. Наприклад, в мережі Wi-Fi Mesh може бути використана система автоматичного відновлення підключення до точок доступу;

– система резервування живлення. Для забезпечення відмовостійкості мережі важливо мати систему резервування живлення. Це можуть бути батареї, джерела живлення з автоматичним переключенням та інші засоби, які забезпечують роботу мережі в разі відмови головного джерела живлення.

Отже, відмінність Wi-Fi Mesh полягає в його відмінній від традиційних бездротових мереж структурі, яка забезпечує високу відмовостійкість.

Керованість

Керованість є однією з важливих характеристик мережі Wi-Fi Mesh, яка описує рівень контролю та управління мережею з боку адміністратора. Керованість може бути реалізована за допомогою різних механізмів, які забезпечують взаємодію між різними компонентами мережі та дозволяють здійснювати контроль за її роботою.

Основні функції керування мережею Wi-Fi Mesh включають [8]:

– керування ресурсами мережі. Адміністратор мережі може визначати доступні ресурси та їх розподіл між користувачами мережі. Таким чином, можна забезпечити рівномірний доступ до ресурсів та підтримку потреб користувачів;

– керування доступом до мережі. Адміністратор мережі може встановлювати правила доступу до мережі, забезпечуючи контроль за безпекою мережі та запобігаючи несанкціонованому доступу користувачів;

– керування якістю обслуговування. Адміністратор мережі може встановлювати правила пріоритезації трафіку та регулювання пропускної здатності мережі, забезпечуючи оптимальні умови для передачі різних типів даних;

– моніторинг та діагностика мережі. Адміністратор мережі може використовувати різні інструменти для моніторингу та діагностики мережі, що дозволяє вчасно виявляти та вирішувати проблеми, що виникають у мережі;

– керування безпекою мережі. Адміністратор мережі може встановлювати правила безпеки, такі як аутентифікація та шифрування, забезпечуючи захист мережі від несанкціонованого доступу та злому.

Покриття Wi-Fi Mesh

Покриття Wi-Fi Mesh є одним з головних параметрів, що визначають якість роботи мережі та її ефективність. Оскільки мережа Wi-Fi Mesh базується на використанні бездротового зв'язку, то її покриття залежить від ряду факторів, зокрема [9]:

– фізичних перешкод – будь-які перешкоди на шляху поширення радіохвиль можуть знижувати якість сигналу та знижувати радіус покриття;

– метеоумов – погода, зокрема дощ, сніг, туман та інші атмосферні явища можуть впливати на якість сигналу та знижувати радіус покриття;

– технічних параметрів обладнання – якість сигналу та радіус покриття можуть залежати від технічних параметрів обладнання, зокрема потужності передавача, чутливості приймача, антен та інших параметрів;

– частотного діапазону – радіус покриття може відрізнятися в залежності від частотного діапазону, що використовується.

Для оцінки покриття мережі Wi-Fi Mesh використовуються різні методи, зокрема [10]:

– метод вимірювання сигналу – полягає у вимірюванні сигналу на різних відстанях від точки доступу та аналізі його параметрів, зокрема потужності та рівня шуму;

– метод моделювання – полягає у створенні математичної моделі мережі та проведенні розрахунків радіуса покриття;

– метод емпіричного дослідження – полягає у встановленні точок доступу на території та вимірюванні якості зв'язку на різних відстанях.

У табл. 1.1 наведено радіус покриття каналів доступу для різних радіочастотних середовищ.

Таблиця 1.1 – Покриття для доступу

Радіочастотне середовище	**Радіус покриття каналів доступу**
2,4 ГГц	до 40 метрів
5 ГГц	до 70 метрів
60 ГГц	до 200 метрів

Як можна побачити із табл. 1.1, радіус покриття залежить від частоти та може досягати значень від 40 до 200 метрів. Оскільки мережа Wi-Fi Mesh базується на використанні різних частотних діапазонів, то при побудові мережі необхідно враховувати можливості та обмеження кожного діапазону.

1.5 Вибір протоколу маршрутизації

Вибір правильного протоколу маршрутизації є важливим аспектом при розробці програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі. У даному розділі ми розглянемо різні протоколи маршрутизації, їх особливості та причини вибору протоколу радіозв'язку IEEE 802.11g для даної задачі.

Протоколи маршрутизації відповідають за визначення оптимальних маршрутів передачі даних між вузлами мережі. У mesh-мережах використовуються спеціальні протоколи, що працюють у динамічних умовах та забезпечують стабільність передачі даних. Розглянемо деякі з них:

1. Optimized Link State Routing (OLSR). OLSR є протоколом маршрутизації, заснованим на принципі передачі стану зв'язку (link state) [11]. Він використовує механізм оптимізації шляхів та працює в режимі

«проактивно», підтримуючи актуальну інформацію про маршрути в мережі. Це забезпечує швидку передачу даних та мінімальні затримки.

2. Ad hoc On-Demand Distance Vector (AODV). AODV є протоколом маршрутизації, заснованим на принципі вектора відстані (distance vector). Він працює в режимі «реактивно», будуючи маршрути лише за потреби. Такий підхід зменшує обсяг передаваної керуючої інформації та знижує навантаження на мережу.

3. Better Approach to Mobile Ad-hoc Networking (BATMAN). BATMAN є реактивним протоколом маршрутизації, який використовує алгоритм повідомлень про доступність (availability) для визначення маршрутів у мережі. Він підтримує маршрутизацію на рівні кінцевих вузлів, що дозволяє забезпечити більш стабільну роботу мережі в динамічних умовах.

4. Dynamic Source Routing (DSR). DSR є реактивним протоколом маршрутизації, який заснований на принципі маршрутизації з джерелом (source routing) [12]. У такому підході вузол-джерело визначає маршрут передачі даних, що забезпечує гнучкість управління трафіком та зменшує кількість керуючої інформації, яка передається по мережі;

5. протокол радіозв'язку IEEE 802.11g. Протокол радіозв'язку IEEE 802.11g є стандартом бездротового зв'язку, що забезпечує передачу даних на частоті 2,4 ГГц та швидкість до 54 Мбіт/с [13]. Цей стандарт є сумісним із старішим стандартом IEEE 802.11b та новішими стандартами, такими як IEEE 802.11n і IEEE 802.11ac. Розглянемо деякі переваги IEEE 802.11g:

– швидкість передачі даних. IEEE 802.11g забезпечує високу швидкість передачі даних, що дозволяє забезпечити голосовий зв'язок відмінної якості, без значних затримок та перерв;

– сумісність. Оскільки IEEE 802.11g сумісний з іншими стандартами бездротового зв'язку, це дозволяє легко інтегрувати

програмно-апаратний комплекс на базі mesh-мережі з існуючими Wi-Fi мережами та пристроями;

– доступність та вартість. Протокол IEEE 802.11g широко використовується в різних областях, що забезпечує доступність обладнання та рішень, заснованих на цьому стандарті. Це дозволяє знизити вартість розробки та впровадження програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі.

При виборі протоколу маршрутизації для програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі варто враховувати такі фактори:

– швидкість передачі даних. Протокол маршрутизації повинен забезпечувати достатню швидкість передачі даних для голосового зв'язку відмінної якості;

– мінімізація затримок. Голосовий зв'язок вимагає мінімальних затримок, тому протокол маршрутизації повинен оптимізувати передачу даних для забезпечення мінімальних затримок;

– стійкість та надійність. Протокол маршрутизації повинен забезпечувати стійку та надійну роботу мережі, навіть у динамічних умовах з відмовами вузлів або змінами топології мережі;

– масштабованість. Протокол маршрутизації повинен дозволяти легко розширювати мережу, додавати нові вузли та пристрої.

1.6 Аналіз існуючих рішень

Для розробки програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі важливо ретельно проаналізувати існуючі рішення та технології на ринку. Це дасть змогу визначити переваги та недоліки кожного рішення, а також знайти оптимальний варіант для впровадження в проект.

Традиційні методи голосового зв'язку

Традиційно голосовий зв'язок забезпечувався через стаціонарні телефонні мережі (PSTN) або мобільні мережі. Проте ці методи мають ряд обмежень:

– висока вартість інфраструктури та експлуатації;

– обмежена покриття, особливо в віддалених та малонаселених регіонах;

– відсутність гнучкості та масштабованості.

Voice over IP (VoIP)

Voice over IP (VoIP) – це технологія передачі голосових даних через Інтернет або інші IP-мережі. VoIP має ряд переваг, таких як:

– низькі витрати на дзвінки, особливо на міжнародні дзвінки;

– висока якість звуку;

– широкий вибір функцій, таких як групові дзвінки, відеодзвінки та передача файлів.

Mesh-мережі для голосового зв'язку

Mesh-мережі є альтернативою традиційним методам голосового зв'язку, таким як PSTN, мобільні мережі та VoIP. Mesh-мережі мають ряд переваг, що роблять їх привабливим рішенням для голосового зв'язку:

– автономність. Можуть працювати без постійного з'єднання з Інтернетом або іншими мережами, що робить їх відмінним варіантом для віддалених або ізольованих регіонів;

– масштабованість. Легко масштабуються, тому що кожен вузол мережі може діяти як ретранслятор, забезпечуючи зв'язок для інших вузлів;

– самоорганізація. Можуть автоматично реорганізуватися в разі змін у топології, що забезпечує високу стабільність та відмовостійкість;

– висока ефективність. Використовують кілька шляхів для передачі даних, що може забезпечити оптимальний варіант навіть при перешкодах або втратах вузлів;

– відкриті стандарти та протоколи. Зазвичай використовують відкриті стандарти та протоколи, що забезпечують широку сумісність та можливість інтеграції з іншими системами.

Для аналізу існуючих рішень на базі mesh-мережі, можна розглянемо декілька відомих продуктів та технологій:

– OpenMesh. Це відкритий проект, який розробляє програмне забезпечення та апаратні компоненти для створення mesh-мереж. OpenMesh використовує відкриті стандарти та протоколи, що забезпечують широку сумісність та гнучкість;

– goTenna Mesh. Це пристрій, який дозволяє користувачам створювати приватні mesh-мережі для обміну текстовими повідомленнями та GPS-координатами без залежності від мобільних операторів або доступу до Інтернету. goTenna Mesh працює на радіочастоті UHF, що забезпечує відносно велику дальність зв'язку та низьку потребу в енергії;

– FireChat. Безкоштовний мобільний додаток, що дозволяє користувачам створювати mesh-мережі для обміну текстовими повідомленнями, використовуючи Bluetooth та Wi-Fi без потреби в мобільному зв'язку або доступі до Інтернету;

– Commotion Wireless: Це відкритий проект, який розробляє програмне забезпечення для створення комунікаційних мереж на базі mesh-технологій. Commotion Wireless використовує відкриті стандарти та протоколи, що забезпечують сумісність з різноманітним обладнанням та системами.

Аналіз існуючих рішень на базі mesh-мережі показує, що mesh-мережі мають великий потенціал для реалізації голосового зв'язку, особливо в ситуаціях з обмеженим доступом до традиційних мереж або з поганим зв'язком. Враховуючи відкриті стандарти та протоколи, використані в більшості mesh-мереж, можливо розробити програмно-апаратний комплекс, що буде сумісний з різним обладнанням та системами.

Однак, для досягнення найкращих результатів у реалізації голосового зв'язку на базі mesh-мережі, важливо врахувати декілька ключових аспектів:

– вибір оптимальних радіочастот та протоколів маршрутизації для забезпечення високої якості зв'язку та мінімізації затримок. Рішення, що використовують IEEE 802.11g, можуть забезпечити надійний зв'язок та відмінну пропускну здатність;

– розробка ефективних методів компресії голосових даних та зменшення впливу перешкод на якість звуку. Використання спеціалізованих кодеків та алгоритмів може допомогти оптимізувати передачу голосових даних у mesh-мережах;

– забезпечення безпеки та приватності комунікацій. Враховуючи відкриту природу mesh-мереж, важливо розробити механізми захисту від несанкціонованого доступу та зловмисного втручання;

– розробка програмного забезпечення та апаратного обладнання, що є сумісними з різними пристроями та системами. Це дозволить користувачам легко інтегрувати програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі в свої існуючі інфраструктури;

– моніторинг та управління mesh-мережею. Розробка інструментів для моніторингу та управління mesh-мережею може допомогти операторам і адміністраторам виявляти та усувати проблеми, а також оптимізувати роботу мережі.

Висновки до розділу 1

У даному розділі було досліджено основні відомості та принципи побудови mesh-мереж, технічні характеристики мережі, елементи мережі, особливості та функції мережі Wi-Fi Mesh, вибір протоколу маршрутизації та аналіз існуючих рішень.

На основі проведеного аналізу, можна зробити наступні висновки:

– Mesh-мережі мають великий потенціал для реалізації голосового зв'язку, особливо в ситуаціях з обмеженим доступом до традиційних мереж або з поганим зв'язком. Вони забезпечують гнучкість та масштабованість, що дозволяє створити стабільні та надійні комунікаційні системи;

– вибір оптимального протоколу маршрутизації є важливим кроком у побудові ефективної mesh-мережі для голосового зв'язку. Відкриті стандарти та протоколи, використані в більшості mesh-мереж, сприяють сумісності з різноманітним обладнанням та системами;

– дослідження існуючих рішень, таких як OpenMesh, goTenna Mesh, FireChat, та Commotion Wireless, показали, що існують значні різниці між ними у функціональності, сумісності та простоті використання. Жодне з них не враховує всі потреби для створення програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі;

– розробка програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі вимагає врахування ряду ключових аспектів, таких як оптимізація радіочастот та протоколів маршрутизації, розробка ефективних методів компресії голосових даних, забезпечення безпеки та приватності комунікацій, сумісність з різними пристроями та системами, а також моніторинг та управління mesh-мережею.

2 РЕАЛІЗАЦІЯ АПАРАТНОЇ ЧАСТИНИ MESH-МЕРЕЖІ

2.1 Вибір платформ для побудови Mesh-мережі

Mesh-мережі стали популярним рішенням для голосового зв'язку в різних сферах, таких як екстрені служби, військові операції, зв'язок в місцях з відсутністю інфраструктури та багатьох інших. При виборі платформи для побудови mesh-мережі необхідно враховувати декілька факторів, таких як доступність апаратних та програмних ресурсів, потреби користувачів, енергозбереження та надійність.

2.2 Опис платформи для мережевого шлюзу з використанням контролеру

Для забезпечення належного рівня надійності системи голосового зв'язку з використанням Mesh-технологій, розроблено систему на двох різних платформах. Основні модулі, які забезпечують підтримку маршрутизації Mesh-топології, працюватимуть на платформі Arduino Nano, тоді як шлюз, який забезпечує доступ WMN до мережі Інтернет та керування кінцевими пристроями за допомогою смартфону, реалізовано на платформі ESP8266. Для радіомодуля обрано поширений LoRa(Long Range) трансивер RFM95, а для виводу інформації про стан роботи шлюзу – LCD-дисплей, що може відображати шістнадцять символів у два рядки, та модуль 2PH84388A для його підключення. Обрані платформи через гнучкість, простоту в програмуванні та наявність великої кількості бібліотек та розширень.

Arduino Nano – це компактна платформа, що містить мікроконтролер ATmega328p із 14 цифровими входами/виходами, 8 аналоговими входами, шиною SPI та кварцовим резонатором на 16 МГц (рис. 2.1) [14]. Ширина і довжина друкованої плати складають відповідно 0.0185 м і 0.043 м. Плата може живитися від USB або зовнішнього джерела живлення від 7.2 до 15 В (вивід VIN) або від 2.7 до 5 В (вивід +5V). Є вбудований лінійний

стабілізатор на 5 В, але не рекомендується перевищувати поріг в 15 В, оскільки модуль може вийти з ладу. Кнопка, що міститься на платі, під'єднана до RESET мікроконтролера і дозволяє перезавантажувати його. Вбудований USB-UART перетворювач на базі мікросхеми FTDI FT232RL подається у випадку, якщо платформа живиться від USB, а на виході 3V3 формується за допомогою мікросхем FTDI.

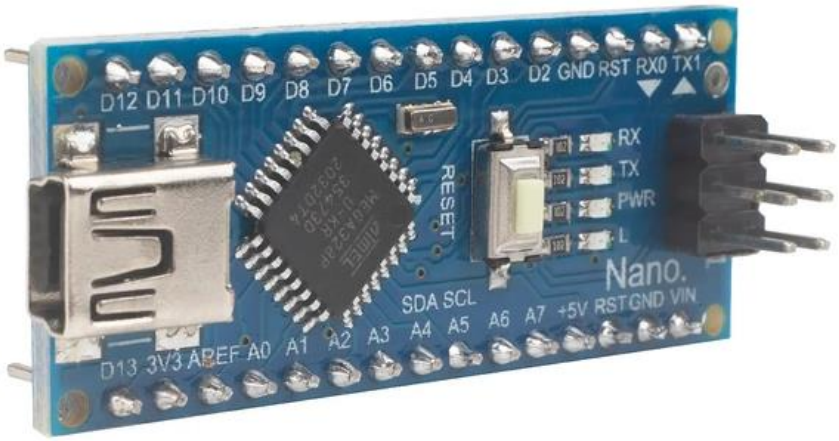


Рисунок 2.1 – Зовнішній вигляд платформи Arduino Nano

Таблиця 2.1 – Основні технічні характеристики Arduino Nano

Характеристика	**Значення**
Мікроконтролер	ATmega328p
Напруга живлення	7.2 – 15 В (вивід VIN), 2.7 – 5 В (вивід +5V)
Вбудований лінійний стабілізатор	Так, на 5 В
Ширина друкованої плати	0.0185 м (1.85 см)
Довжина друкованої плати	0.043 м (4.3 см)
Цифрові входи/виходи	14
Аналогові входи	8
Шина SPI	Так
Кварцовий резонатор	16000000 Гц (16 МГц)
Вбудований USB-UART	Так, на базі мікросхеми FTDI

Характеристика	Значення
перетворювач	FT232RL
Кнопка RESET	Так
Наявність вбудованого світлодіоду	Так
Розміри світлодіоду	0.005 м (5 мм)
Напруга на виході 3V3	Формується за допомогою мікросхем FTDI

ATmega328p – це 8-бітний мікроконтролер, що входить до сімейства AVR виробництва компанії Microchip Technology (раніше – Atmel). Він є одним з найпоширеніших мікроконтролерів у світі і використовується в різних пристроях та системах, включаючи Arduino.

ATmega328p має такі основні технічні характеристики:

– тактова частота: 20000000 Гц (20 МГц);

– об'єм пам'яті програм: 32000 байт = 32 Кбайт (з яких 0.5 Кбайт зайняті заголовком);

– об'єм оперативної пам'яті: 2000 байт (2 Кбайт);

– кількість вхідних/вихідних ліній (GPIO): 23;

– кількість аналогових вхідних ліній: 6;

– кількість таймерів: 3;

– кількість зовнішніх переривань: 2;

– інтерфейси: USART, SPI, I²C;

– інтерфейс програмування та налагодження: ISP (In-System Programming), JTAG (Joint Test Action Group).

Мікроконтролер ATmega328p може працювати в широкому діапазоні напруг живлення від 1.8 до 5.5 В, що дозволяє використовувати його в різних пристроях, включаючи портативні пристрої, датчики, системи автоматизації та інші [15].

Також варто зазначити, що ATmega328p має вбудований апаратний дивайдер для вимірювання напруги на аналогових входах, який дозволяє виконувати точні вимірювання і забезпечує високу якість звуку у системах зв'язку. Окрім того, ATmega328p має вбудований програмований генератор кварцевих коливань і вбудований затримувач часу (Real Time Clock), що дозволяє створювати системи, які мають точну взаємодію з часом.

Основна функція ядра мікроконтролера ATmega328p полягає в забезпеченні правильного виконання програми, що передбачає доступ до пам'яті, виконання обчислень, контроль за периферійними пристроями та обробку переривань. Ці функції забезпечують надійну та стабільну роботу мікроконтролера в різних пристроях та системах, включаючи Arduino. Загальну схему мікроконтролера ATmega328p зображено на рис. 2.2.

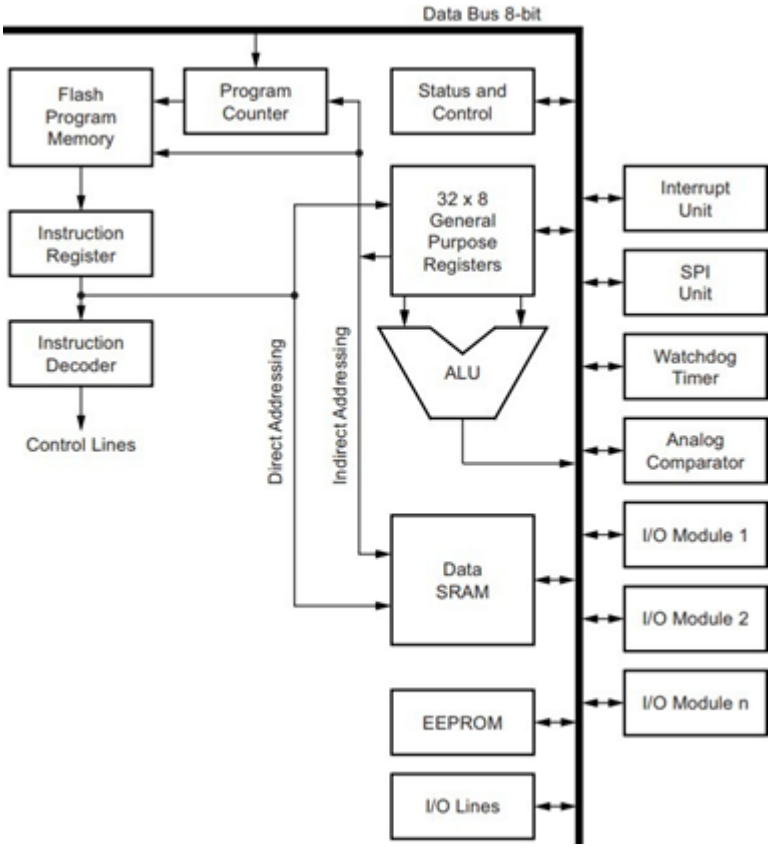


Рисунок 2.2 – Загальна схема мікроконтролера ATmega328p

На платі Arduino присутня можливість обміну інформацією з іншими пристроями, включаючи інші плати Arduino, комп'ютери або мікроконтролери. Це забезпечується за допомогою цифрових виводів RX і

TX, які дозволяють здійснювати зв’язок завдяки вбудованому UART-інтерфейсу [16]. Крім того, за допомогою стандартної бібліотеки SoftwareSerial, можна перетворити будь-який цифровий вивід на віртуальний послідовний інтерфейс. В середовищі Arduino IDE також є монітор послідовного порту, який дозволяє відправляти або приймати текст у вигляді ASCII символів через USB-інтерфейс, що спрощує налагодження та взаємодію з пристроями, що підключені до плати. Повний перелік всіх виходів та їх призначення для платформи Arduino Nano на базі мікроконтролера ATmega328p зображено на рис. 2.3.

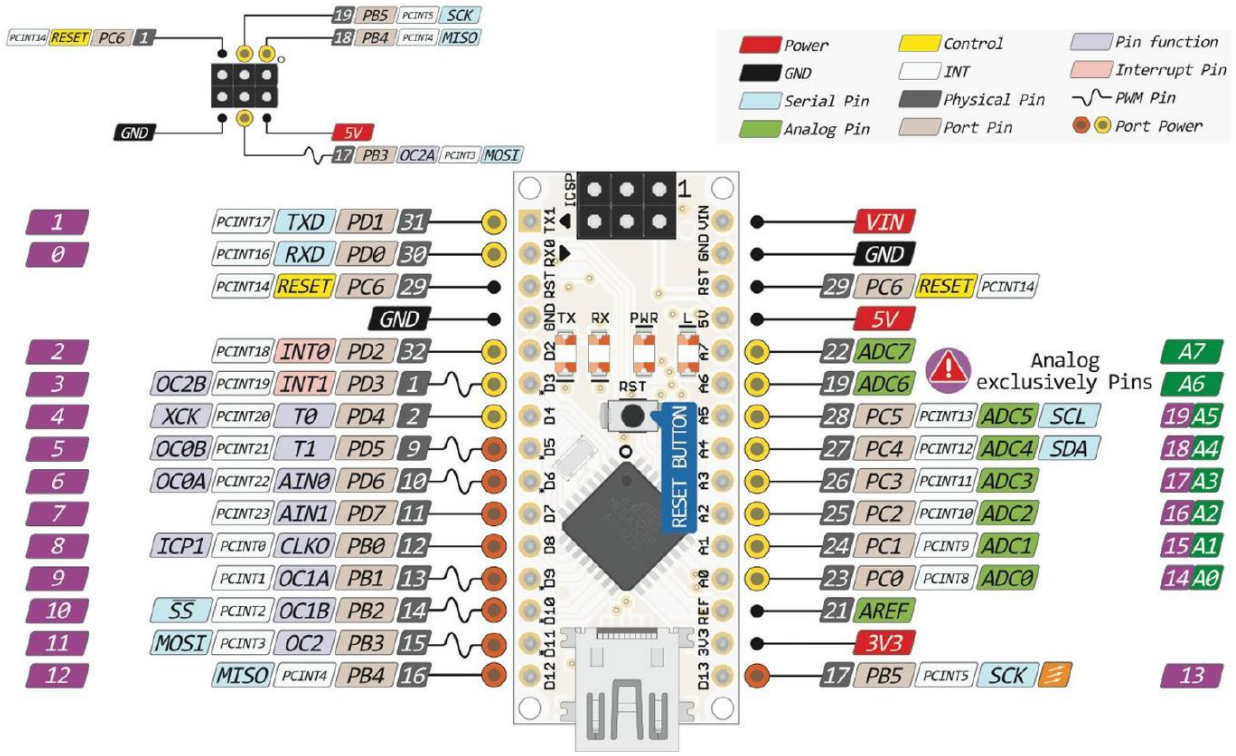


Рисунок 2.3 – Позначення виходів платформи Arduino Nano

Arduino Nano має кілька переваг серед аналогів, зокрема:

– компактний розмір, що робить його ідеальним для використання в проектах з обмеженим простором;

– легкість у використанні та програмуванні завдяки дружньому для користувачів середовищу Arduino IDE та великій кількості доступних бібліотек та розширень;

– низька вартість порівняно з іншими мікроконтролерами.

До недоліків можна віднести:

– обмежена пам'ять та швидкість процесора можуть бути недостатніми для деяких складних проектів, що вимагають великої кількості обчислень та багато ресурсів;

– обмежена кількість пінів в порівнянні з більшими мікроконтролерами, що може бути проблемою для деяких проектів;

– є певні обмеження щодо живлення, особливо при використанні вбудованого стабілізатора напруги, що може призвести до неправильної роботи або пошкодження плати.

Загалом, Arduino Nano є корисним та економічним рішенням для багатьох простих проектів, але для складніших та більш вимогливих проектів можуть бути кращі альтернативи.

2.3 Трансивер RFM95, як основа Mesh-мережі

RFM95 є радіопередавачем, що використовує технологію LoRa для передачі даних на великі відстані з малим енергоспоживанням (рис. 2.4). Він працює на частотах від 868000000 Гц до 915000000 Гц (від 868 МГц до 915 МГц), що дає змогу використовувати його в різних регіонах світу.

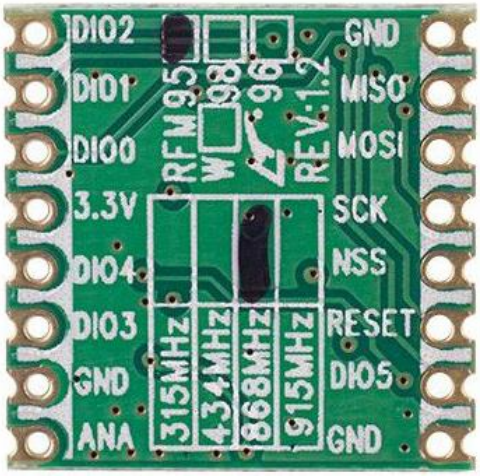


Рисунок 2.4 – Зовнішній вигляд радіопередавача RFM95

RFM95 має високу чутливість (-148 дБм) та потужність передачі до 20 дБм, що дозволяє передавати дані на відстань до 10000 м (10 км) в

залежності від умов оточення. Також, він підтримує вбудовану функцію автоматичного вибору частоти (AFC), яка дозволяє автоматично підтримувати точність частоти, що дуже важливо при передачі на великі відстані.

RFM95 має також вбудований декодер/кодер даних, що дозволяє передавати дані в різних форматах, включаючи бездротові мережі (wireless mesh networks), що є важливим для розробки складних систем з багатьма вузлами [17]. Крім того, він підтримує такі функції, як контроль передачі, кодування помилок та детектування колізій, що дозволяє забезпечити надійну та безперебійну передачу даних.

RFM95 може працювати в різних режимах, включаючи режим прийому, передачі та сну. Це дозволяє ефективно використовувати енергію та продовжувати тривалість роботи від батареї.

RFM95 є досить поширеним та добре документованим пристроєм, що дає змогу легко інтегрувати його в різні проекти. Він є частиною великої сімейства радіочастотних пристроїв, що розробляються компанією Semtech, та має велику кількість доступних бібліотек та прикладів програмного забезпечення для різних платформ, таких як Arduino, Raspberry Pi та інші.

Таблиця 2.2 – Основні технічні характеристики RFM95

Характеристика	**Значення**
Робоча частота	від 868000000 Гц до 915000000 Гц (від 868 МГц до 915 МГц)
Максимальна потужність відправлення	20 дБм
Чутливість	-148 дБм
Швидкість передачі даних	до 300000 біт/с (до 300 кбіт/с)
Інтерфейс зв'язку	SPI
Довжина пакету даних	до 256 байт

Джерело живлення	1.8-3.7 В
Характеристика	**Значення**
Підтримка LoraWAN	Так
Розмір	0.0161 x 0.0161 м (16.1 x 16.1 мм)
Вага	0.0016 кг (1.6 г)
Документація та підтримка	Існує відкрита документація та доступні бібліотеки для різних платформ

Основною функцією радіопередавача RFM95 є забезпечення зв'язку за допомогою модема LoRa, який базується на мікросхемі RF96. RFM95 здатен працювати на широкому спектрі частот, має високий рівень захисту від завад та споживає мінімальну потужність. З цим радіопередавачем можна створювати надійні та швидкі мережі зв'язку з великою зоною покриття та мінімальним енергоспоживанням. Архітектура RFM95 зображена на рис. 2.5.

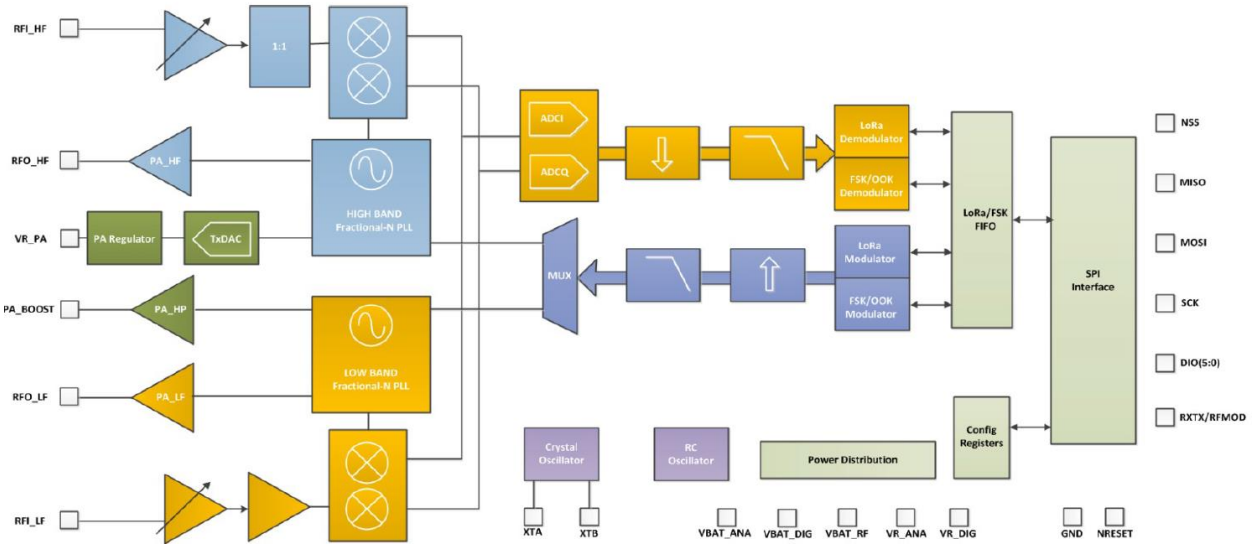


Рисунок 2.5 – Архітектура радіопередавача RFM95

Радіопередавач RFM95 оснащений модемом LoRa на базі мікросхеми RF96, що дозволяє забезпечувати зв’язок у широкому спектрі частот та високий захист від завад. Модуль дозволяє налаштовувати пропускну здатність, спектр поширення, коефіцієнт поширення та швидкість

виправлення помилок, а кожен коефіцієнт розширення є ортогональним, що дозволяє одночасно приймати дані з різних передавачів одним каналом зв’язку. Блок-схема модему LoRa має незалежний буфер даних FIFO з подвійним портом, до якого можна отримати доступ через інтерфейс SPI [18]. Це дозволяє швидко переключатися між FSK та LoRa модемами, а переключення можна здійснювати в режимі сну. Процес модуляції та демодуляції є комерційною таємницею, але відомо, що він використовує форму модуляції з розширеним спектром, в поєднанні з кодуванням циклічного виправлення помилок. Логічну схему модему LoRa зображено на рис. 2.6.

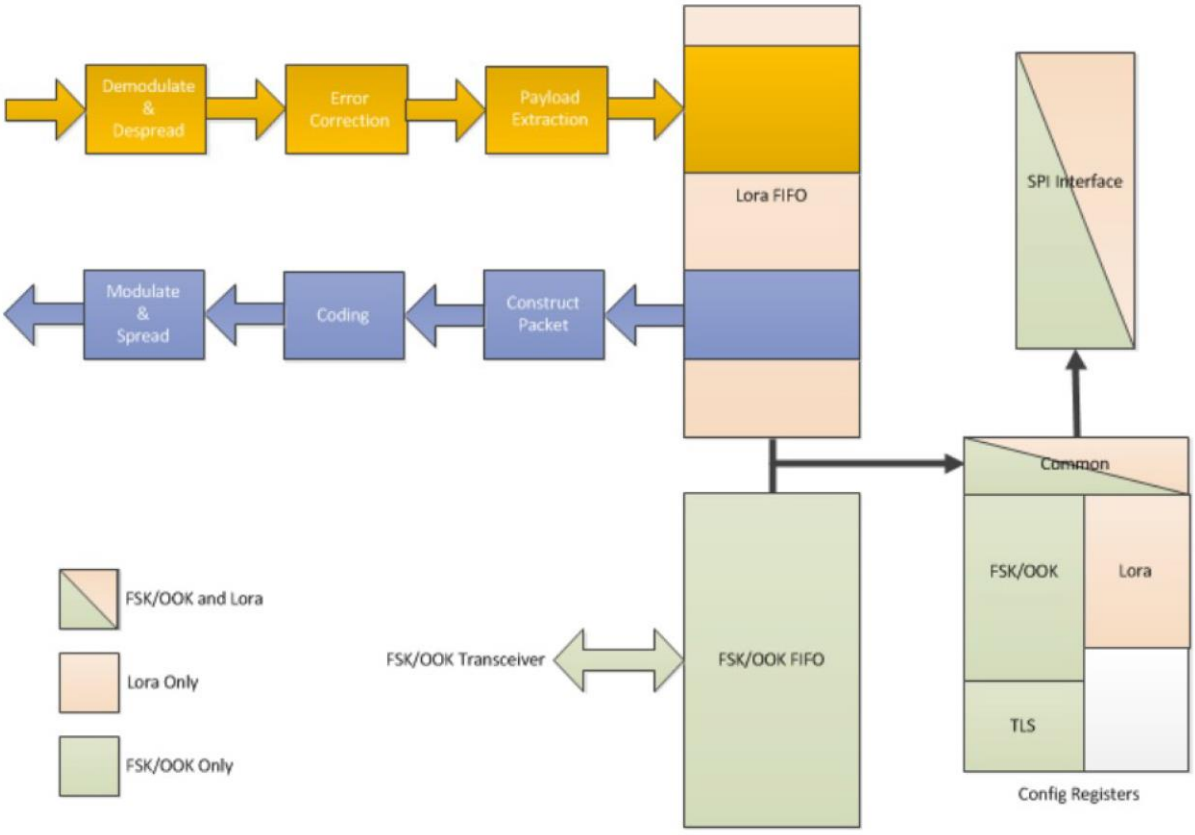


Рисунок 2.6 – Логічна схема модему LoRa

LoRa-модем використовує два типи формату пакетів: явний та неявний. Явний пакет містить короткий заголовок з інформацією про кількість байтів, швидкість кодування та наявність CRC в пакеті. Формат пакету можна побачити на рисунку 2.7.

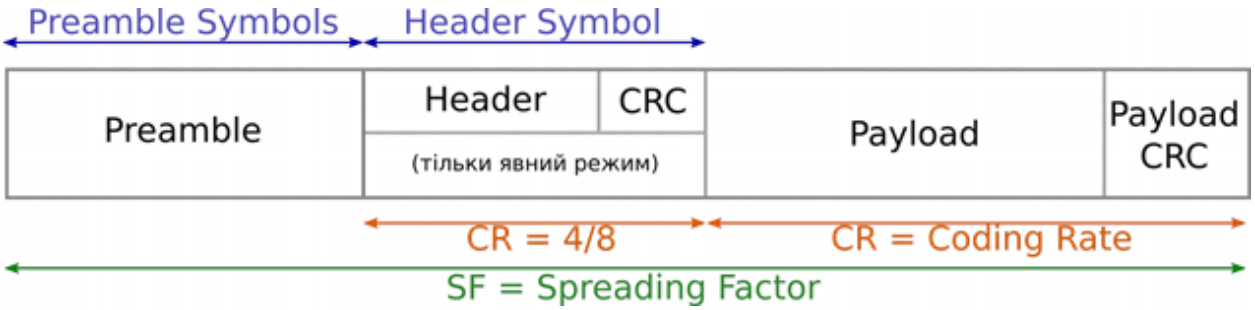


Рисунок 2.7 – Вміст пакету LoRa

Преамбула – це символьний рядок, що використовується для синхронізації приймача з потоком вхідних даних. Розмір преамбули можна змінювати, щоб скоротити робочий цикл приймача під час інтенсивної передачі даних. Довжина преамбули повинна бути налаштована ідентичною довжині передавача. Залежно від режиму роботи доступні два типи заголовків.

Оскільки у LoRa модемі є два режими передачі пакетів: явний та неявний. В явному режимі заголовок пакету містить інформацію про довжину корисного навантаження, швидкість корекції помилок та наявність додаткового CRC для корисного навантаження. Заголовок має свій власний CRC для перевірки його правильності. У неявному режимі заголовок відсутній, що скорочує час передачі пакету, але довжина корисного навантаження, швидкість кодування помилок та наявність CRC повинні бути налаштовані заздалегідь.

Радіопередавач RFM95 має три типи цифрового інтерфейсу: статичні регістри конфігурації, регістри стану та буфер даних FIFO. Інтерфейс між трансивером та хост-мікроконтролером виконується за допомогою послідовного периферійного інтерфейсу SPI. Доступ до всіх регістрів можливий через командний код та параметри, які передаються через піни MOSI та SCK. Для запису в регістр пін CS має бути в низькому стані, а для передачі 16-бітних слів, пін CS повинен бути підтягнутий до низького рівня. Таким чином, інтерфейс дозволяє керувати та слідкувати за станом RFM95 через хост-мікроконтролер.

Модуль RFM95 підтримує спілкування з мікроконтролерами за допомогою послідовного інтерфейсу SPI. Для цього він має три типи цифрових інтерфейсів: статичні регістри конфігурації, регістри стану та буфер даних FIFO. Доступ до них можна отримати через SPI-інтерфейс. Максимальна тактова частота на SPI-шині – 20000000 Гц = 20 МГц, при цьому трансивер підтримує SPI-режим 0.0. Для здійснення комунікації з модулем, пін CS повинен утримуватись в низькому стані, а дані отримуються через MOSI і тактуються по зростаючому фронту SCK. RFM95 посилає дані через MISO і тактується по спадаючому фронту SCK. Пін MISO встановлюється в нижній логічний рівень по замовчуванню, коли пін CS у високому рівні, і має три-становий буфер. Крім цього, модуль має внутрішню POR-схему, яка встановлює значення по замовчуванню у всіх регістрах, які він має.

У модемі RFM95 доступні різні регістри конфігурації, які можна читати та записувати через послідовний периферійний інтерфейс. Зокрема, є регістри конфігурації LoRa та регістри спільні для режимів FSK/OOK та LoRa. Регістри можна читати в будь-який режим, включаючи режим сну, проте їх слід писати тільки в режимі сну чи очікуванні. Також є регістри стану, які надають інформацію про стан трансивера. У режимі LoRa автоматичний секвенсор верхнього рівня недоступний. Крім того, модем має певні обмеження на швидкість передачі даних через SPI-інтерфейс, так як максимальна тактова частота на SPI-шині – 20000000 Гц = 20 МГц, і пін CS повинен бути у низькому стані для комунікації між мікроконтролером та RFM95.

RF96 має 256-байтний буфер оперативної пам'яті, який використовується як буфер даних FIFO в режимі LoRa. Цей буфер доступний для читання через інтерфейс SPI і може бути використаний для збереження переданих та отриманих даних. За замовчуванням, при увімкненні живлення, половина пам'яті виділяється під передачу, а інша половина – під прийом.

Регістри FifoTxBaseAd та FifoRxBaseAd вказують на адресу в пам'яті, де зберігаються передані та отримані дані. Однак, режим сну та зберігання даних не дозволяють доступ до буфера FIFO, оскільки буфер автоматично очищається від старого вмісту при переході до режиму прийому. Зміна базових адрес для прийому та передачі може бути налаштована в області 256-байтної оперативної пам'яті.

Трансивер RF96 має 256-байтний буфер оперативної пам'яті, який використовується як буфер даних FIFO в режимі LoRa. Цей буфер можна прочитати через інтерфейс SPI в будь-якому режимі, крім режиму сну та зберігання даних. Буфер може використовуватися для зберігання переданої та отриманої інформації завдяки подвійній конфігурації портів. Регістри FifoTxBaseAd та FifoRxBaseAd вказують на адресу в пам'яті, де зберігаються передані та отримані дані, відповідно. Для використання максимального розміру буфера, можна встановити базові адреси регістрів в нижній області пам'яті. Регістр FifoRxBytesNb визначає розмір комірки пам'яті для запису в разі успішного прийому, а змінна FifoRxCurrentPtr вказує на розташування останнього отриманого пакета у FIFO. Всі отримані дані, включаючи ті з невалідним CRC, записуються у буфер даних FIFO, що дозволяє їх обробляти та вирішувати проблеми зі зв’язком. При отриманні пакета розміром більше буфера, частину його буде записано в передавальну частину буфера даних.

На рис. 2.8 представлена блок-схема алгоритму передачі даних з використанням LoRa модема. Для зменшення витрат енергії можна активувати блоки RF, PLL та PA лише в тому випадку, якщо необхідно передавати пакети даних, що дозволяє зберігати енергію.

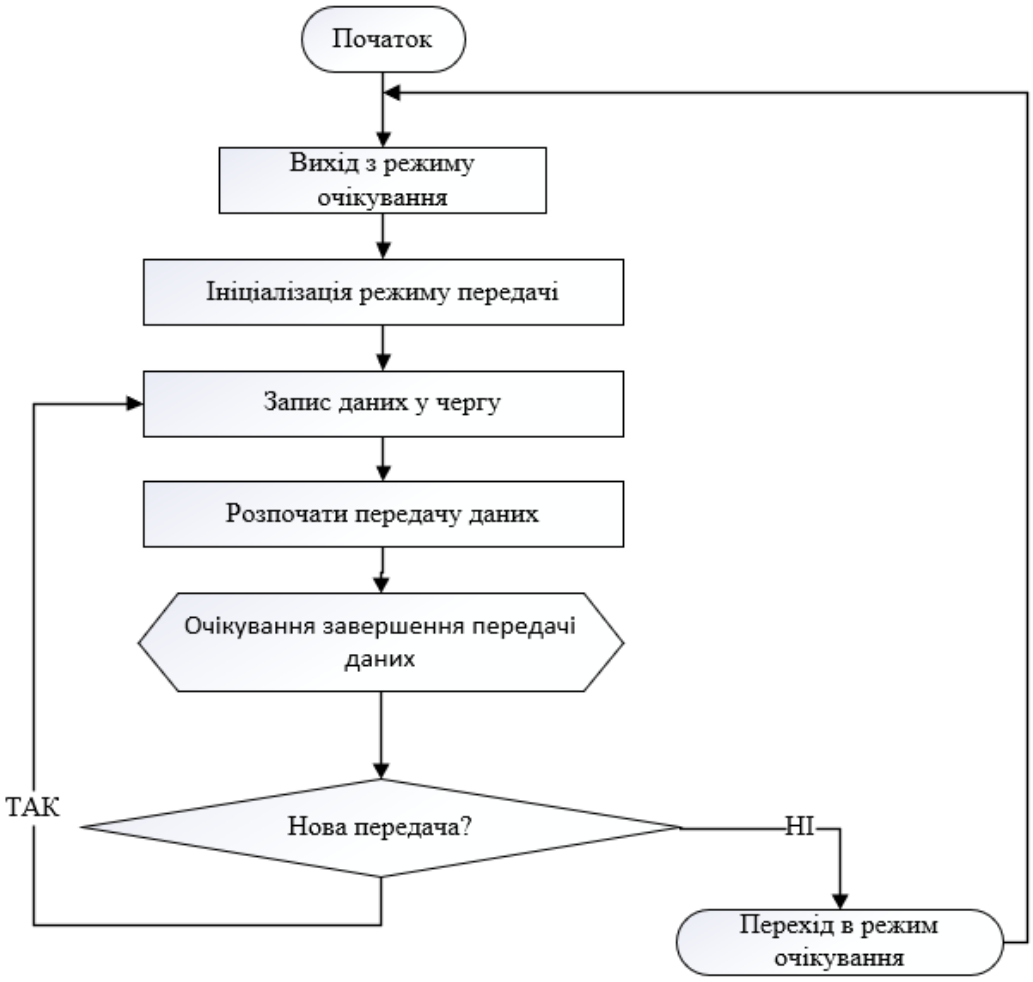


Рисунок 2.8 – Блок-схема алгоритму передачі пакету в LoRa

Для запису даних у FIFO буфер необхідно спочатку вивести трансивер з режиму очікування і ініціалізувати режим передачі. Статичні регістри конфігурації доступні лише у режимі сну або після ініціалізації режиму передачі. Після ініціалізації режиму передачі можна записати дані у буфер, і модуль автоматично розпочне передачу, яка завершиться сигналом переривання TxDone. Для прийому пакета використовується схожий спосіб – необхідно вийти з режиму очікування, ініціалізувати режим прийому та очікувати на сигнал прийому RxDone. Після цього можна прочитати дані з буферу. Подібним чином відбувається прийом пакету (рис. 2.7).

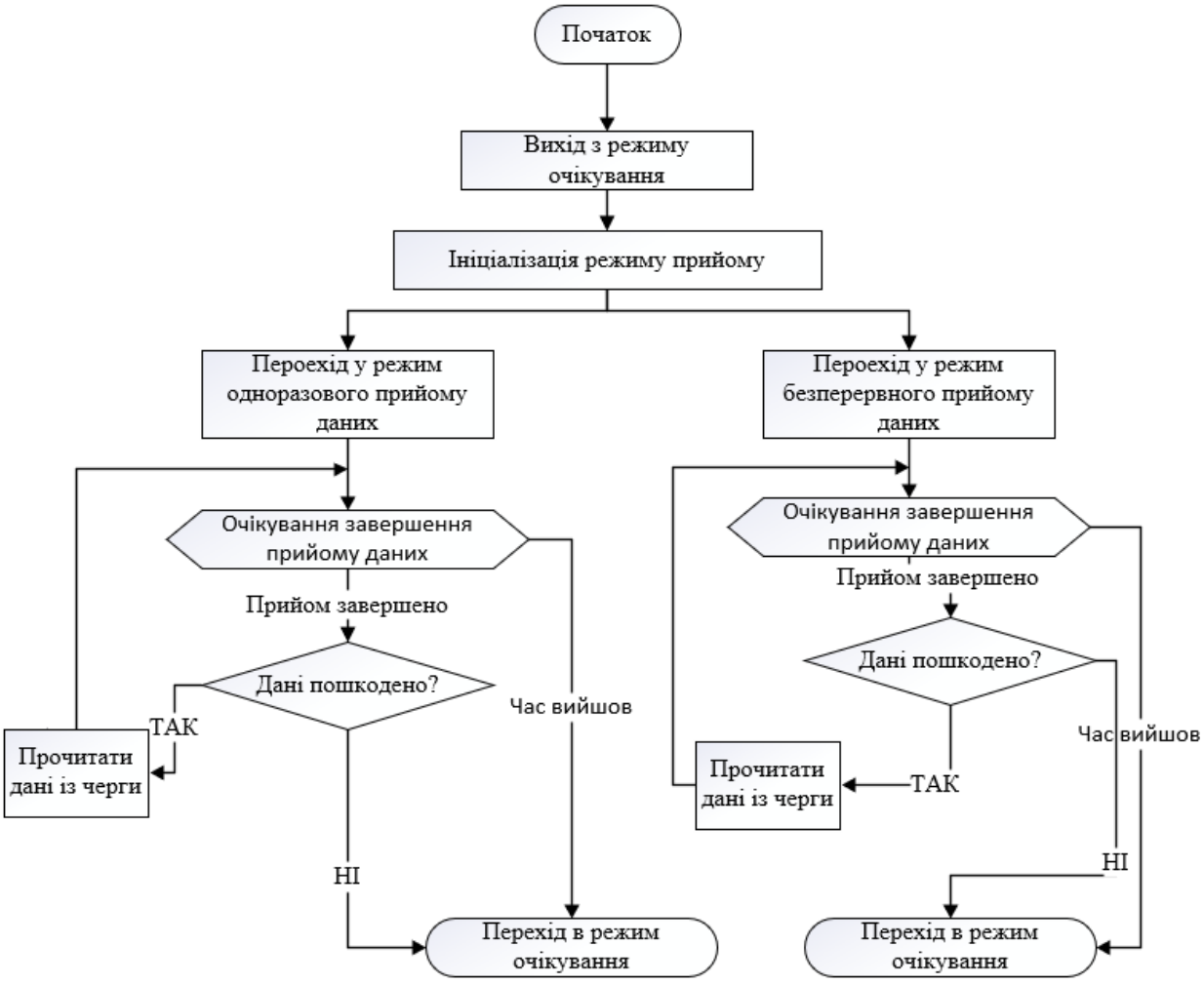


Рисунок 2.9 – Блок-схема алгоритму прийому пакету в LoRa

У безперервному режимі прийому даних, трансивер постійно шукає преамбулу та сигнали синхронізації, не зупиняючись. Цей режим використовується, коли потрібно постійно моніторити канал для отримання пакетів, наприклад, у випадку радіостанцій, які працюють у пожежній охороні або медичному обладнанні. У цьому режимі пакети приймаються без переривання, тому корисне навантаження потрібно витягувати з FIFO буфера засобами програми. При цьому, реєстри конфігурації не можна змінювати, поки трансивер знаходиться в цьому режимі, і переривання викликаються лише у випадку отримання пакету з правильним CRC. Цей режим потребує більшої кількості енергії, оскільки модем постійно перебуває в режимі прийому.

У режимі безперервного прийому даних модем постійно сканує канал на преамбулу і зберігає корисне навантаження у буфер після кожного виявленого пакету. У цьому режимі важливо враховувати, що байти записуються в пам'ять буфера даних в отриманому порядку, тому необхідно правильно обробляти адресний вказівник, щоб запобігти переповненню буфера. Модем автоматично фільтрує отримані пакети на основі адреси, а також можна налаштувати фільтрування на основі вмісту перших кількох байтів корисного навантаження, що дозволяє відкидати непотрібні пакети і підвищувати енергоефективність.

2.4 Опис Wi-Fi модуля ESP8266

Платформу, що базується на модулі ESP8266, було обрано NodeMcu, так як вона має декілька переваг, таких як автономне живлення від акумулятора стандарту 18650, вбудований перетворювач, який дозволяє ефективно використовувати ємність акумулятора, та автономний модуль зарядки від Micro-USB. Платформа розроблена для створення пристроїв IoT та має можливість передавати та отримувати інформацію в локальну мережу або в інтернет за допомогою Wi-Fi. На платі є також E32 LoRa модуль з SMA конектором для підключення зовнішньої антени та восьмипіновий роз'єм для радіомодуля NRF24L01. Платформа містить 1 аналоговий вхід, 10 цифрових входів/виходів, шини SPI та I²C, а також дві кнопки для перезавантаження та переведення у режим програмування. На рис. 2.10 зображено зовнішній вигляд платформи , що базується на модулі ESP8266.

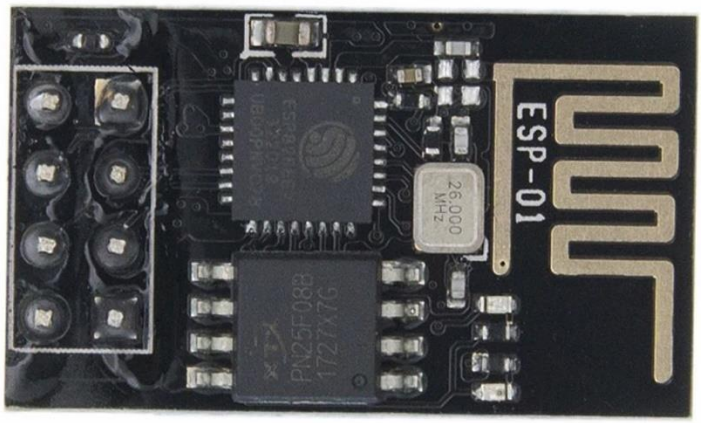


Рисунок 2.10 – Зовнішній вигляд платформи на базі ESP8266

ESP8266 має 32-бітний процесор, вбудовану пам'ять із програмним забезпеченням та RAM, і може бути використаний як самостійний мікроконтролер або як модуль вбудованої системи [19]. Крім того, цей мікроконтролер має дуже низьку вартість, що робить його привабливим для використання в багатьох проектах Інтернету речей (IoT).

Таблиця 2.3 – Основні технічні характеристики ESP8266

Характеристика	**Значення**
Частотний діапазон	24 000 000 000 Гц (2.4 ГГц)
Модуляція	DSSS/CCK
Потужність передачі	18 dBm
Чутливість приймача	-98 dBm
Швидкість передачі даних	до 72 200 000 біт/с (72.2 Мбіт/с)
Підтримка стандартів	802.11 b/g/n
Інтерфейси	UART, SPI, I²C, GPIO
Живлення	3.0-3.6 В
Споживання енергії	до 170 мА в режимі передачі, до 20 мА в режимі очікування
Розміри	0.0016 х 0.0024 м (16 х 24 мм)

ESP8266 можна програмувати за допомогою різноманітних мов програмування, включаючи C++, Python і Lua, та з використанням різноманітних інтерфейсів програмування, таких як Arduino IDE, MicroPython, NodeMCU та ін.

Мікроконтролер ESP8266 можна використовувати у типових проектах з використанням готових вбудованих програм, які можуть бути розподілені на кілька груп. Перша група включає програми, які працюють під керуванням зовнішнього контролера через UART. Друга група – програми з вбудованими інтерпретаторами різних мов високого рівня, які дозволяють виконувати скрипти розробника пристрою. Третя група – програми для Інтернету речей, які дозволяють підключити ESP8266 до набору датчиків і виконавчих пристроїв і надають необхідну мережеву функціональність для роботи в інфраструктурі Інтернету речей. У цій групі є програми, які реалізують віртуальний перехідник UART-WiFi.

На рис. 2.11 показано повний перелік всіх виходів та їх призначення для Wi-Fi модуля на базі мікроконтролера ESP8266. При встановленні високого рівня на виході EXT_RSTB, джерело виконуваної програми модуля визначається станом портів GPIO2 і GPIO0. Найважливіші режими – це виконання коду з UART (GPIO2=1, GPIO1=0) та з зовнішньої флеш-пам'яті (GPIO2=1, GPIO0=1). Перший режим використовується для запису нової програми в флеш-пам'ять, а другий режим є штатним робочим.

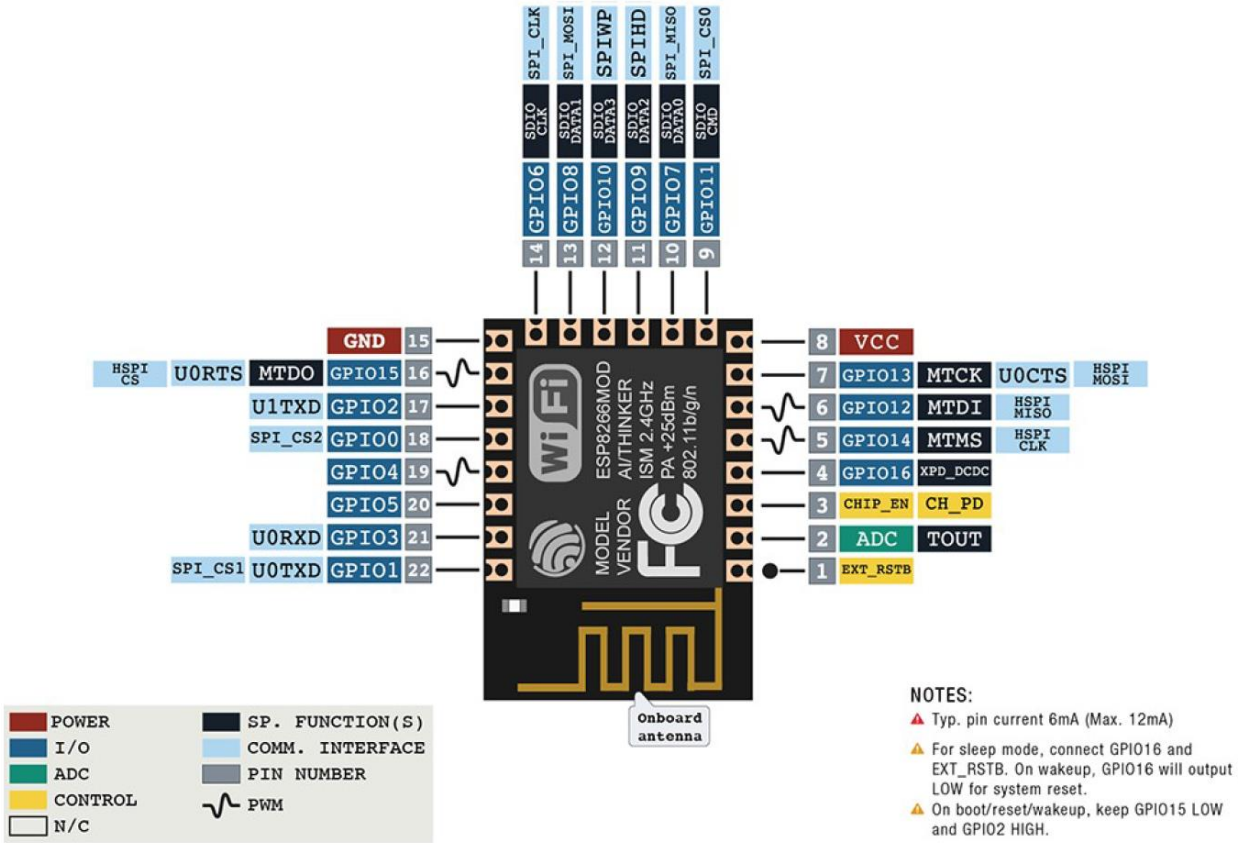


Рисунок 2.11 – Позначення виходів модуля ESP8266

У пристрої передбачена можливість оновлення вбудованої програми через Wi-Fi. Флеш-пам'ять розділена на кілька частин, одна відведена менеджеру вбудованих програм, а дві інші – під програми користувача. При оновленні програми, новий образ завантажується у вільну ділянку флеш-пам'яті, після чого після перевірки цілісності менеджер перемикає маркер, звільнюючи ділянку пам'яті зі старою вбудованою програмою і починаючи виконання коду з нової ділянки.

ESP8266 – це пристрій, який може працювати в режимі точки доступу або клієнта. У режимі клієнта ESP8266 працює в локальній мережі і для цього необхідно задати SSID Wi-Fi мережі та пароль. Для початкового налаштування параметрів зручно використовувати режим точки доступу, в якому пристрій видно при стандартному пошуку мереж в планшетах і комп'ютерах. Для цього потрібно підключитися до пристрою і відкрити HTML сторінку конфігурації веб-браузера та задати ім'я локальної Wi-Fi

мережі та її пароль. Після цього пристрій штатно підключиться до мережі в режимі клієнта.

Після налаштування параметрів Wi-Fi, платформа отримує IP-адрес локальної мережі, який можна використовувати для доступу до неї. Доступ можна отримати через мережу Інтернет, наприклад, з мобільного телефону. В цьому випадку пристрій працює в режимі сервера, до якого звертається зовнішній клієнт. Крім того, доступ можна отримати за допомогою мережевого імені або сервісу, якщо вони налаштовані.

Платформа на основі ESP8266 зазвичай працює в локальній мережі офісу або будинку, з’єднаній з мережею Інтернет через маршрутизатор. Маршрутизатор отримує свою IP-адресу від провайдера інтернету і забезпечує трансляцію локальних адрес в мережу провайдера. Така конфігурація не дозволяє звертатися до локальних адрес з боку мережі Інтернет, але забезпечує вільну видимість інтернет-адрес у локальній мережі.

Більшість маршрутизаторів надають можливість налаштування трансляції мережевих адрес між локальною та глобальною мережами за допомогою технології DMZ. Це дозволяє звертатися до сервера в локальній мережі з глобальної мережі, знаючи тільки IP-адресу, видану маршрутизатором провайдером. Однак, такий підхід може бути не зручним, оскільки потребує налаштування маршрутизатора та вручного визначення його IP-адреси, яка може змінюватися.

Для розв'язання проблеми доступу до сервера в локальній мережі з глобальної мережі можна скористатися технологією DDNS. Це реалізовано через спеціальні інтернет-сервіси, які надають користувачам можливість створити свій обліковий запис з унікальним ім'ям та параметрами, що прописуються в налаштуваннях модуля. Далі модуль періодично звертається до DDNS-сервера, повідомляючи йому ім'я свого облікового запису та свою поточну IP-адресу. Кінцевий користувач звертається до того ж DDNS-сервісу, отримуючи поточну IP-адресу. Основна проблема полягає в

гарантіях існування конкретного DDNS-сервісу, які зазвичай гарантуються лише при використанні комерційних сервісів, за які потрібно платити.

2.5 Використання LCD-дисплею

Для моніторингу та відображення інформації про стан LoRa Mesh-мережі використовується рідкокристалічний LCD-дисплей з можливістю відображення двох рядків по шістнадцять символів. Дисплей заснований на контролері HD44780.

Для зменшення кількості провідників, що підключають дисплей до плати, використовують модуль 2PH84388A та шину I2C [20]. Підключення до плати здійснюється за допомогою 2 провідників для передачі даних та 2 провідників для живлення. До цифрових портів вводу-виводу D1 і D2 під'єднуються виводи SCL і SDA відповідно (рис. 2.12). Для живлення модуля використовується напруга +3.3 вольт. Результатом такого з'єднання є можливість виводити на дисплей інформацію щодо моніторингу та налагодження LoRa Mesh-мережі.

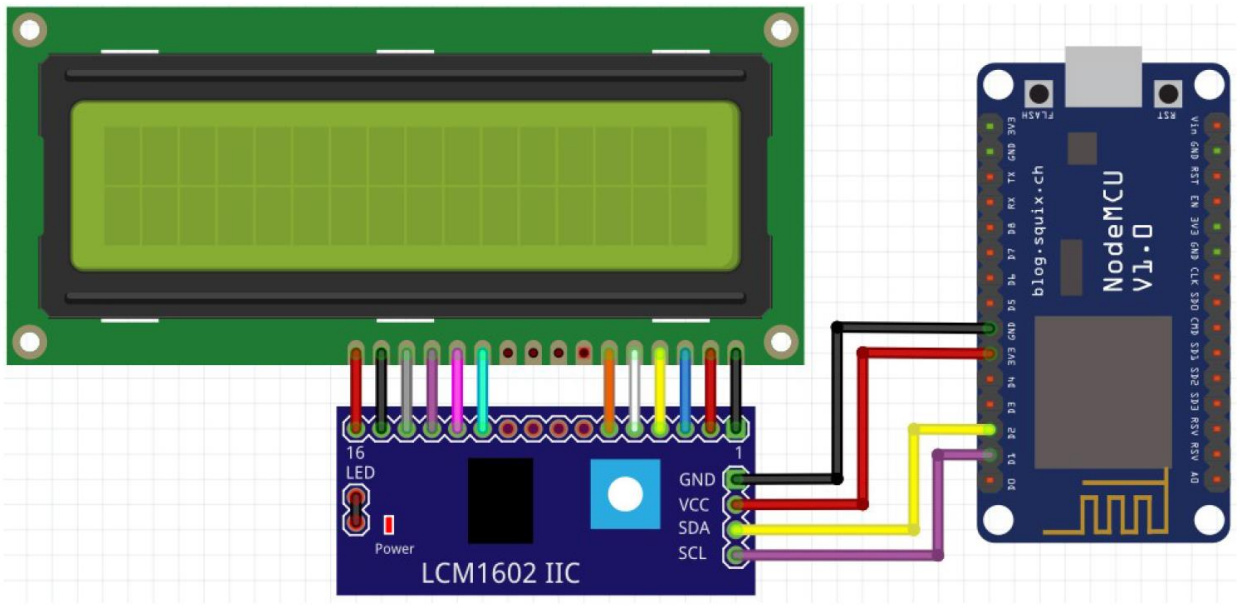


Рисунок 2.12 – Загальна схема підключення LCD дисплею

На платі модуля для керування LCD-дисплеєм є резистор для регулювання контрастності та перемичка для включення підсвітки дисплею.

Виходи 1–16 призначені для підключення модуля до виводів LCD-дисплею, а перемичками A0–A2 можна змінювати адресу пристрою.

Для підключення рідкокристалічного дисплею до модуля 2PH84388A використовується схема з 4-бітною шиною даних. Контролер відправляє команди на дисплей, які керують режимами його роботи і ASCII-кодами символів, які виводяться на екран. Дисплей також може передавати контролеру інформацію про свій внутрішній стан і дані зі своїх внутрішніх блоків пам'яті за запитом, але в даній схемі така можливість не передбачена, оскільки вивід R/W замкнутий на землю.

I²C – це послідовна шина, яка передає дані і адреси по лінії SDA порозрядно. Тактовий сигнал на лінії SCL використовується для передачі бітів інформації. Під час дії сигналу SCL (SCL = 1) стан лінії SDA не повинен змінюватися, а зміна даних на лінії SDA відбувається при відсутності тактового сигналу на лінії SCL (SCL = 0). За винятком бітів, що визначають початок і кінець процесу обміну інформацією, формування стартового біту полягає у зміні рівня сигналу на лінії SDA з логічної 1 на логічну 0 при SCL = 1, а формування стопового біту полягає у зміні рівня сигналу на лінії SDA з логічної 0 на логічну 1 також при SCL = 1 (рис. 2.13).

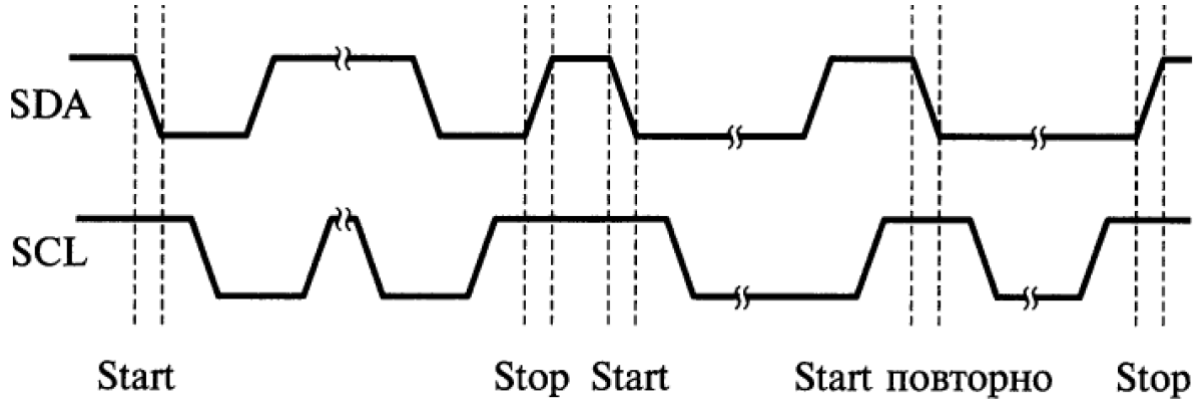


Рисунок 2.13 – Часова діаграма сигналів інтерфейсів

Для зміни адреси пристрою LCD дисплею використовується модуль 2PH84388A, на якому запаюються перемички для встановлення потрібної адреси. Для передачі команд або даних на дисплей необхідно виставити потрібний логічний рівень на виводі RS та використовувати керуючий сигнал

на лінії E. Для запису даних в дисплей використовується спадний фронт сигналу. Після прийому байту даних або команди контролеру HD44780 потрібен час на обробку інформації, тому модуль 2PH84388A не повинен передавати дані в цей момент. Якщо передаються ASCII-коди символів, то вони записуються в буфер даних, який містить вісімдесят комірок.

Для передачі даних на цьому LCD-дисплеї використовуються сигнали RS, RW, E, D4, D5, D6 і D7. Сигнал RS використовується для вибору типу передачі даних: логічний «1» вказує на передачу символу, а логічний «0» вказує на передачу команди. Сигнал RW використовується для вказівки напрямку передачі даних: логічний «1» означає читання даних з дисплею, а логічний «0» означає запис даних на дисплей. Сигнал E використовується для синхронізації передачі даних: при переході сигналу E зі стану «0» до стану «1» відбувається запис/читання даних.

Для передачі даних використовуються сигнали D4-D7, які відповідають за передачу байту даних по 4 біта на півцикла E. При передачі даних байт поділяється на дві групи по 4 біта, які передаються послідовно через лінії D4-D7. Спочатку передається старший біт, а потім менший. Після передачі кожного з 4-бітних пів-байтів, сигнал E змінює свій стан зі «1» на «0» і знову на «1». Таким чином, за один цикл сигналу E передається 8 біт даних (2 пів-байти) за 2 пів-цикли E.

Всі ці сигнали контролюються контролером HD44780, який керує роботою LCD-дисплея. Цей контролер зберігає символи у своєму внутрішньому буфері і контролює їх виведення на дисплей. Він також відповідає за керування рівнем контрастності і підсвічування дисплея.

Висновки до розділу 2

Даний розділ присвячений детальному аналізу апаратних компонентів програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі.

Для досягнення високої надійності системи голосового зв'язку було вирішено використовувати дві платформи: Arduino Nano та ESP8266. Arduino Nano обрано для реалізації основних модулів, що забезпечують маршрутизацію в Mesh-топології, тоді як ESP8266 використовується для створення мережевого шлюзу, який надає доступ до Інтернету та керування кінцевими пристроями через смартфон.

У розділі було розглянуто основні характеристики платформ Arduino Nano та ESP8266, визначено їх підходящість для використання у контексті досліджуваної системи. Трансивер RFM95 відіграє важливу роль у побудові mesh-мережі, забезпечуючи стабільне радіозв'язку між вузлами.

Застосування Wi-Fi модуля ESP8266 дозволяє створити зв'язок між mesh-мережею та Інтернетом, що робить систему доступною для керування та моніторингу за допомогою смартфонів. Окрім того, використання LCD-дисплею на основі контролера HD44780 дозволяє відображати інформацію про статус та параметри мережі, спрощуючи процес відлагодження та налаштування системи.

Загалом, реалізація апаратної частини Mesh-мережі забезпечує надійність та гнучкість програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі, враховуючи потреби користувачів та технічні можливості обраних платформ.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЧАСТИНИ MESH-МЕРЕЖІ

3.1 Розробка програмної частин для мережевого шлюзу

LoRa Mesh-технологія безпровідної радіозв'язку дозволяє передавати невеликі обсяги даних на великі відстані з використанням трансивера малої потужності. Для використання цієї технології можна використовувати дешеві модулі RFM95 та платформу Arduino Nano, до якої можна під'єднати трансивер за допомогою шини SPI. Використання такої платформи з мікроконтролером ATmega328p є одним з найпростіших способів розробки на базі технології LoRa Mesh.

3.2 Налаштування послідовного периферійного інтерфейсу

RFM95 є залежним пристроєм, який комунікує з хост-мікроконтролером (ATmega328p) за допомогою 4-дротового послідовного периферійного інтерфейсу (SPI). Інтерфейс SPI дозволяє керувати та слідкувати за станом RFM95 через мікроконтролер. Для з'єднання трансивера з мікроконтролером через інтерфейс SPI, необхідно використовувати провідники мінімальної довжини, оскільки частота шини є високою.

Протокол SPI – це протокол передачі даних між керуючим (Master) та підлеглим (Slave) пристроями, що використовує трьохпровідникову шину даних (MISO, MOSI, SCK). Керуючий пристрій ініціює передачу даних, генеруючи тактові імпульси на лінії SCK, підлеглий пристрій відправляє дані на лінії MISO, а керуючий пристрій відправляє дані на лінії MOSI. Один і той самий провідник може використовуватися для передачі даних в обидві сторони, залежно від того, хто ініціює передачу. Роль керуючого пристрою зазвичай виконує мікроконтролер, який контролює роботу всіх підлеглих пристроїв.

SPI-інтерфейс передачі даних включає в себе три провідники – MISO, MOSI та SCK, а також окрему лінію SS (Slave Select) для активації підлеглого

пристрою. Керуючий пристрій, зазвичай мікроконтролер, контролює передачу даних між собою та підлеглими пристроями. Дані передаються від керуючого до підлеглих по MOSI, а від підлеглих до керуючого по MISO, за допомогою тактового сигналу SCK. Передача даних починається лише після активації підлеглого пристрою з лінії SS зі значенням «0», а в іншому випадку дані будуть проігноровані.

Для передачі даних за допомогою SPI-інтерфейсу до трансивера RFM95, спочатку потрібно налаштувати шину. У бібліотеці RadioHead доступна програмна емуляція SPI, що дозволяє підключати трансивер до будь-якого з 14 цифрових портів вводу-виводу на платформі Arduino Nano, крім RX та TX, які зайняті шиною UART. Також у трансивера є лінія DIO0, яку використовують для надсилання переривань мікроконтролеру. При отриманні переривання, мікроконтролер призупиняє виконання програми та обробляє його. Для переналаштування ліній DIO0 та SS, можна використати конструктор на початку програми.

```
// Ініціалізуємо трансивер та передаємо йому обрану частоту
if(rf95.init()) {
    Serial.println("RFM95W Module Init OK!");
} else {
    Serial.println("RFM95W Module Init Failed!");
}
rf95.setFrequency(915.0);

// Перевизначаємо піни для ліній DIO0 та SS
rf95.setPins(RFM95_CS, RFM95_INT);
```

Рисунок 3.1 – Перевизначення ліній DIO0 та SS

При використанні трансивера RFM95 потрібно дотримуватися певних правил передачі даних. Зокрема, пін SS повинен бути підтягнутим до нижнього рівня під час передачі 16-бітового слова, а біти даних на піні MISO передаються до пристрою до підйому в такті на піні SCK, за умови, що пін SS має низький логічний рівень.

3.3 Робота з EEPROM пам'яттю

Кожен пристрій в мережі LoRa Mesh має свій унікальний ідентифікатор, який дозволяє пристроям взаємодіяти та правильно формувати таблицю маршрутизації. Цей ідентифікатор можна зберігати у внутрішній енергонезалежній пам'яті мікроконтролера, такій як EEPROM. Це дозволяє зберігати ідентифікатор після вимкнення пристрою, а також змінювати програмне забезпечення мікроконтролера без втрати ідентифікатора.

ATmega328P має вбудовану EEPROM пам'ять, яка забезпечує зберігання даних, що не втрачаються при вимкненні живлення. Розмір пам'яті складає 1 Кбайт, її можна зчитувати та записувати окремими байтами. Однак, ресурс EEPROM пам'яті обмежений і становить приблизно 100000 циклів стирання/запису. Щоб виконати запис даних в EEPROM, потрібно враховувати особливості роботи платформи Arduino Nano, а саме, уникати запису в першу секунду роботи платформи через можливу недостатню напругу на лінії живлення. Процес зчитування або запису в EEPROM пам'ять призводить до зупинки виконання інструкцій на 4 або 2 тактові цикли відповідно.

EEPROM пам'ять може бути пошкоджена при нестабільній напрузі живлення, що може призвести до некоректного виконання інструкцій центральним процесором і роботи внутрішньої пам’яті. Це може статися як з внутрішньою, так і з зовнішньою EEPROM пам'яттю. Є два випадки, коли дані у пам'яті можуть пошкодитися: коли дані записуються в одні й ті самі комірки при низькій напрузі живлення, та коли процесор виконує некоректну процедуру запису при занадто низькій напрузі.

Уникнення пошкодження даних в EEPROM можна досягти завдяки активному утриманню внутрішнього інверсійного сигналу AVR RESET в низькому рівні, коли напруга живлення знижується нижче необхідного рівня. Це можна зробити, використовуючи внутрішню систему контролю живлення

(СКЖ). Якщо СКЖ виявляє зниження напруги під час запису даних, операція буде призупинена до нормалізації живлення. Це забезпечує збереження цілісності даних і попереджує їхнє пошкодження.

Для запису байту даних в комірку EEPROM пам'яті на мові С потрібно використати функцію *EEPROM.write()*. Під час виконання цієї функції, необхідно заборонити глобальні переривання командою *cli()*, оскільки запис до EEPROM пам'яті є досить часоємним і може спричинити неочікувану поведінку системи у випадку виникнення переривань. Для підтвердження запису даних в EEPROM використовується функцію *commit()*. На рис. 3.2 наведено код на мові С++ для запису байту даних у комірку EEPROM пам'яті

```
#include <EEPROM.h>

void writeDataToEEPROM(int address, byte data) {
    cli(); // Заборонити глобальні переривання
    EEPROM.write(address, data); // Записати байт даних в EEPROM пам'ять
    EEPROM.commit(); // Підтвердити запис данис в EEPROM
    sei(); // Розрішити глобальні переривання
}
```

Рисунок 3.2 – Виконання запису в EEPROM пам’ять

Щоб запобігти можливому конфлікту при доступі до пам'яті, перед читанням даних з EEPROM потрібно переконатися, що операції пов'язані з перезаписом флеш-пам'яті завершено. Для читання байту з EEPROM пам'яті необхідно використати функцію *EEPROM.read()*, яка поверне значення з вказаної комірки пам'яті. Після завершення роботи з EEPROM пам'яттю потрібно дозволити глобальні переривання командою *sei()* (рис. 3.3).

```
#include <EEPROM.h>

byte address = 0; // Адреса комірки пам'яті, яку потрібно прочитати
byte data; // Змінна для зберігання прочитаних даних

cli(); // Забороняємо глобальні переривання

data = EEPROM.read(address); // Читаємо байт даних з комірки пам'яті

sei(); // Дозволяємо глобальні переривання

// Потім можна використовувати прочитані дані
```

Рисунок 3.3 – Зчитування EEPROM пам'яті

3.4 Керування трансивером RFM95

Було обрано пакет бібліотек RadioHead для роботи з LoRa трансивером RFM95. Цей пакет складається з двох наборів класів: драйвера та менеджера. Драйвер забезпечує низькорівневий доступ до внутрішніх регістрів трансивера, що дозволяє ефективно керувати модулем за допомогою простих викликів. Менеджер надає високорівневий доступ для управління процесом надсилання та отримання пакетів даних. Бібліотеки підтримують більшість платформ Arduino, включаючи Arduino Nano та Raspberry Pi, і дозволяють надсилати та отримувати повідомлення у формі пакетів даних.

Перед створенням об'єкту менеджера RHMesh потрібно прочитати унікальний ідентифікатор пристрою з EEPROM пам'яті та записати його у змінну «nodesId». Далі потрібно ініціалізувати драйвер RFM95 та менеджера мережі RHMesh. У функції *loop()* можна отримувати та обробляти повідомлення від інших пристроїв мережі. Для цього використовуються методи бібліотеки RHMesh, такі як *recvFrom()* та *sendtoWait()*. Метод *recvFrom()* отримує повідомлення та повертає адресу пристрою-відправника, з якого прийшло повідомлення, а також довжину повідомлення. Метод *sendtoWait()* дозволяє надіслати повідомлення до конкретного пристрою з мережі та очікувати на відповідь від цього пристрою.

Крім того, бібліотека RHMesh підтримує автоматичну маршрутизацію повідомлень, тобто вона сама визначає найкоротший шлях для передачі повідомлення до призначення, що значно спрощує розробку та налагодження мережі. Також можливе налаштування параметрів мережі, таких як максимальна кількість спроб надіслати повідомлення, таймаути для очікування відповіді та інші.

Фрагмент коду використання менеджера RHMesh зображено на рис. 3.4.

```
// Отримання повідомлення від іншого пристрою
if (manager.available()) {
    // Оголошення змінної для збереження отриманого повідомлення
    uint8_t buf[RH_MESH_MAX_MESSAGE_LEN];

    // Отримання повідомлення
    uint8_t len = sizeof(buf);
    if (manager.recvfromAck(buf, &len)) {
        // Обробка отриманого повідомлення
    }
}

// Відправлення повідомлення іншому пристрою
if (/* умова для відправлення повідомлення */) {
    // Оголошення змінної для збереження відправленого повідомлення
    uint8_t data[] = "Hello World";

    // Відправлення повідомлення
    if (manager.sendtoWait(data, sizeof(data), /* адрес отримувача */)) {
        // Обробка відправленого повідомлення
    }
}
```

Рисунок 3.4 – Використання бібліотек RadioHead

RadioHead пакет бібліотек підтримує чотири основні менеджери для забезпечення надійності передачі повідомлень та використання фіксованої довжини пакетів: RHDatagram, RHReliableDatagram, RHRouter та RHMesh. Крім того, усі драйвери мають однаковий API, що дозволяє їх використовувати самостійно, але такий режим роботи є менш надійним. У деяких випадках пакет бібліотек дозволяє ініціалізувати більше одного драйвера та менеджера і використовувати їх для різноманітних завдань. Кожен менеджер має свої особливості, але в цілому дозволяє надсилати та

отримувати повідомлення з заданою довжиною та забезпечити їх надійну доставку.

В контексті тестування Mesh-мережі, слід зазначити, що складність розміщення вузлів таким чином, щоб вони могли взаємодіяти лише між собою, є досить високою. Однак, бібліотека RadioHead надає можливість моделювання такої ситуації, використовуючи кілька тестових мереж, де деякі вузли не мають можливості взаємодіяти між собою. При визначенні такої мережі, бібліотека RadioHead просто ігнорує повідомлення, отримані від вузлів, які не повинні знаходитися в радіусі дії конкретного модуля. Це значно спрощує моделювання роботи мережі. Щоб бібліотека RadioHead ігнорувала повідомлення від вузлів, які не повинні знаходитися в радіусі дії конкретного модуля, необхідно визначити список вузлів, з якими може взаємодіяти поточний вузол. Наприклад, якщо потрібно заборонити взаємодію з вузлами з ID 2 та 3, то код може бути таким як на рис. 3.5.

```
// Оголошення списку вузлів з якими може взаємодіяти поточний вузол
uint_t allowedNodes[] = {1, 4, 5};

void loop() {
    // Отримання повідомлення
    uint8_t buf[RH_MESH_MAX_MESSAGE_LEN];
    uint8_t len = sizeof(buf);
    uint8_t from;
    if (manager.recvfromAck(buf, &len, &from)) {
        //Перевірка, чи належить відправник списку дозволених вузлів
        bool allowed = false;
        for (int i = 0; i < sizeof(allowedNodes); i++) {
            if (from == allowedNodes[i]) {
                allowed = true;
                break;
            }
        }
        if (allowed) {
            // Обробка отриманого повідомлення
            // ...
        } else {
            // Повідомлення відхилено
        }
    }
}
```

Рисунок 3.5 – Визначення обмежень для вузла мережі

Отже, Mesh-мережа є складною системою, в якій кожен вузол намагається з'єднатися з будь-яким іншим вузлом у мережі, в процесі він відстежує таблицю маршрутизації та рівень сигналу від інших вузлів. Кожен вузол може взаємодіяти тільки з тими вузлами, які знаходяться в межах його радіусу дії, ігноруючи повідомлення від інших вузлів. Кожен вузол має свою структуру даних, яка містить цю інформацію.

3.5 ESP8266, як основа Mesh-шлюзу

Шлюз, який розроблено на базі мікроконтролера ESP8266, дозволяє забезпечити доступ до мережі Інтернет та керувати нею зі звичайного смартфона або комп'ютера. Шлюз розподілено на два блоки пам'яті – 1 та 3 мегабайти. У першому блоку знаходиться виконавчий код програми, а в другому – файлова система SPIFFS, куди можна завантажити різні файли, такі як HTML сторінки, CSS-файли та конфігураційні файли для правильної роботи трансивера, MQTT та веб-серверів, що забезпечує коректну роботу серверів. Всі ці можливості дозволяють забезпечити ефективну роботу мережі та зручний контроль за нею.

Увімкнення трансивера спочатку ініціалізує файлову систему та завантажує необхідні конфігураційні файли, які містять HTML сторінки, CSS файли, конфігураційні файли для роботи трансивера, MQTT та веб-серверів. Після ініціалізації файлової системи та завантаження конфігураційних файлів, трансивер переводить Wi-Fi модуль у режим клієнта та намагається підключитися до точки доступу, вказаної у конфігураційних файлах. Якщо спроба підключення була невдалою, трансивер переводить Wi-Fi модуль у режим точки доступу та запускає MQTT-клієнт, який намагається підписатися на MQTT-брокер, використовуючи адресу, порт, логін та пароль з конфігураційних файлів. Наступним кроком є ініціалізація веб-сервера та всіх його API, після чого всі необхідні дані виводяться на рідкокристалічний дисплей.

Для правильної роботи та взаємодії з мережею необхідно, щоб клієнтські пристрої підтримували сучасні веб-браузери та мали мінімальні апаратні вимоги. Такі вимоги забезпечують коректну роботу веб-інтерфейсу та доступ до всіх функцій шлюзу.

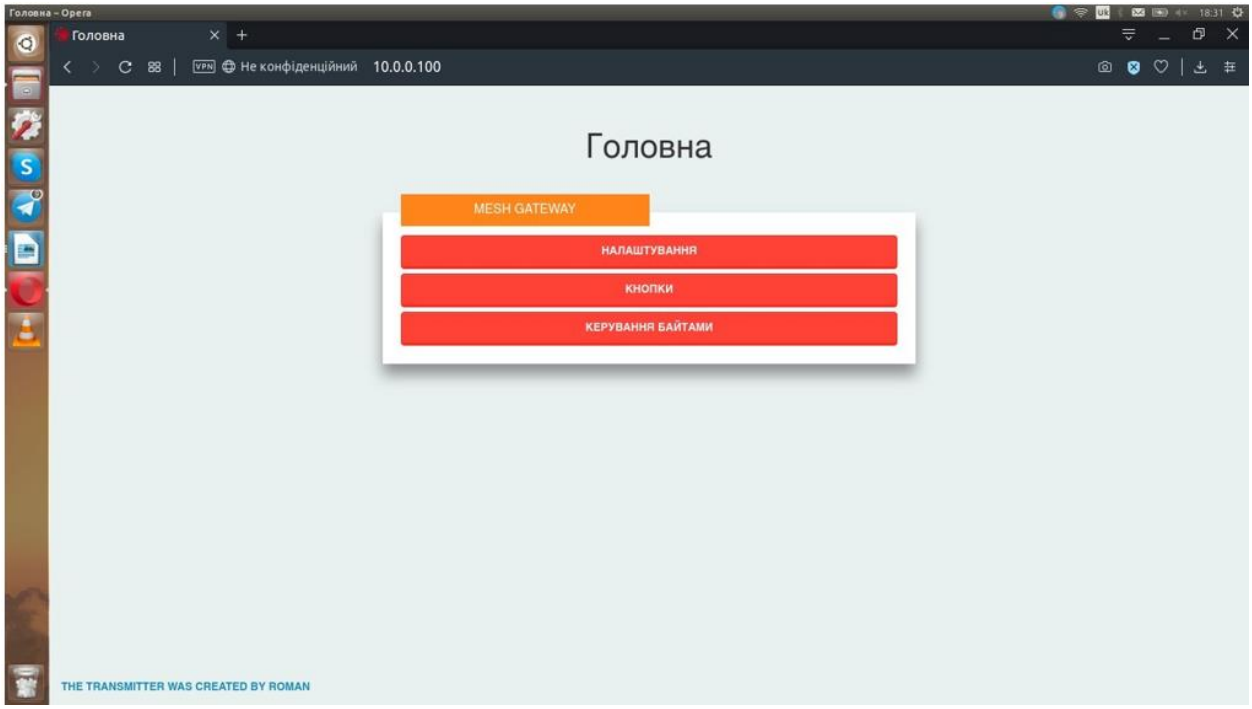


Рисунок 3.6 – Стартова сторінка LoRa Mesh-шлюзу

Шлюз забезпечує доступ до керування мережею через веб-інтерфейс, який можна відкрити за допомогою будь-якого сучасного браузера. Для входу на сторінку адміністрування потрібно перейти за статичною IP-адресою пристрою 192.168.0.1, якщо шлюз перейшов у режим точки доступу. Якщо ні, то він буде доступний у переліку пристроїв у локальній мережі або за допомогою команди iscover -i wlo1 --timeout=3 в терміналі. Режим користувача дозволяє зручно та швидко керувати мережею за допомогою інтуїтивно зрозумілих елементів управління. Рідкокристалічний дисплей виводить інформацію про динамічну адресу, яку можна використати для входу на сторінку адміністрування мережі.

3.6 Робота з протоколом MQTT

Протокол MQTT (Message Queue Telemetry Transport) є протоколом прикладного рівня, що дозволяє обмінюватися повідомленнями між пристроями, використовуючи принцип видавець-підписник. У проекті використовується для передачі таблиці маршрутизації у реальному часі з мережі Mesh, щоб візуалізувати структуру мережі та показати, які вузли можуть безпосередньо взаємодіяти один з одним, а які вузли потребують проміжних вузлів для передачі повідомлень. MQTT протокол працює на стеку протоколів TCP/IP та забезпечує надійну та ефективну передачу даних між пристроями.

MQTT працює на стеку протоколів TCP/IP і дозволяє створювати ієрархію каналів зв'язку, які можуть мати різний рівень підписників. Кожен раз, коли у видавця з'являються нові дані для поширення серед підписників, повідомлення супроводжується приміткою контролю доставки. Канали можуть мати будь-який рівень відгалуження від нижньої частини ієрархії, і це полегшує обмін інформацією різного обсягу. У проекті протокол використовується для передачі таблиці маршрутизації у реальному часі з метою візуалізації структури Mesh-мережі.

Бібліотека PubSubClient дозволяє легко налаштувати MQTT-клієнта на LoRa Mesh-шлюзі для підключення до MQTT-брокера. Це забезпечує можливість публікувати або підписуватися на топіки MQTT за допомогою функцій *publish()* та *subscribe()*. На рис. 3.5 наведений приклад коду мовою C для налаштування клієнта та підключення до MQTT-брокера.

```
void callback(cahr* topic, byte* payload, unsigned int length) {
    Serial.print("Message received [");
    Serial.print(topic);
    Serial.print("]: ");
    for (int i=0;i<length;i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        if(client.connect("ESP32Client", mqtt_user, mqtt_password)) {
            Serial.println("connected");
            client.subscribe(mqtt_topic);
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}
```

Рисунок 3.7 – Функції підключення шлюзу до MQTT-брокера

Спочатку задаються налаштування WiFi та MQTT брокера. Потім викликаються функції *setup_wifi()* та *setup()*, де встановлюється підключення до WiFi і MQTT брокера відповідно. Функція *loop()* перевіряє стан підключення до брокера і виконує функцію *reconnect()*, якщо підключення було втрачено.

Функція *callback()* викликається, коли клієнт отримує повідомлення з топіка, на який він підписаний. Даний код демонструє спосіб підключення до MQTT-брокера та обробки отриманих повідомлень, але цю бібліотеку можна використовувати для більш складних задач, таких як керування IoT-пристроями або збір телеметрії з датчиків.

Щоб встановити з'єднання потрібно визначити IP-адресу та порт сервера, до якого потрібно підключитися. Для цього було використано структуру sockaddr_in. Код для встановлення з'єднання зображено на рис. 3.8.

```
struct sockaddr_in servaddr;

// встановлюємо IP-адресу та порт сервера
char* ip = "127.0.0.1";
int port = 8080;

// створюємо сокет
sockfd = socket(AF_INET, SOCK_STREAM, 0);

// заповнюємо структуру sockaddr_in
memset(&servaddr, 0, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr(ip);
servaddr.sin_port = htons(port);

// встановлюємо з'єднання
if (connect(sockfd, (struct sockaddr*)&servaddr, sizeof(servaddr)) < 0) {
    perror("Помилка з'єднання");
    exit(1);
} else {
    printf("З'єднання встановлено успішно.\n");
}

// закриваємо сокет
close(sockfd);

return 0;
```

Рисунок 3.8 – Встановлення з'єднання

У наведеному коді встановлюється з'єднання з сервером, який запущений на локальній машині за адресом 127.0.0.1 на порті 8080. Якщо з'єднання успішно встановлено, програма виведе повідомлення «З'єднання встановлено успішно». У кінці програма закриває сокет.

Коли дані публікуються або отримуються через MQTT-брокер, вони закодовані в двійковому форматі, оскільки протокол є бінарним. Щоб отримати справжній зміст повідомлення, необхідно його інтерпретувати. Іноді брокери MQTT можуть накопичувати повідомлення на каналах, де немає поточних підписників, які вони будуть зберігати або відкидати, в залежності від налаштувань керуючого повідомлення. Це корисно, коли нові підписники можуть потребувати останніх повідомлень.

В якості сервера було обрано MQTT-брокер Mosquitto. Mosquitto – це відкритий і безкоштовний MQTT-брокер, який може бути встановлений як на малопотужних одноплатних комп'ютерах, так і на повноцінних серверах. Для

встановлення Mosquitto потрібно відкрити термінал та ввести послідовність команд, які дозволяють додати репозиторій, оновити всі пакети і встановити брокер (рис. 3.9).

```
sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
sudo apt-get update
sudo apt-get install mosquitto
```

Рисунок 3.9 – Встановлення Mosquitto на Ubuntu

Для того, щоб задати логін та пароль для Mosquitto, потрібно створити файл конфігурації та внести в нього налаштування безпеки. Для цього можна скористатися командою sudo mosquitto_passwd -c /etc/mosquitto/passwd <username>, де <username> – це бажаний логін користувача. Після введення команди система запропонує ввести пароль для користувача. Після цього потрібно внести зміни в файл конфігурації Mosquitto, додавши рядки allow_anonymous false та password_file /etc/mosquitto/passwd у секцію security. Ці зміни дозволять заборонити доступ анонімним користувачам та використовувати файл з логінами та паролями для авторизації.

Команда service mosquitto status дозволяє перевірити стан Mosquitto-брокера після перезавантаження. Якщо брокер успішно запустився, відповідь буде містити інформацію про цей процес, а якщо щось пішло не так, буде показано повідомлення про помилку. Отже, перезавантажуємо брокер Mosquitto та переконуємось, що все функціонує (рис. 3.10).

```
 mosquitto.service - Mosquitto MQTT v3.1/v3.1.1 Broker
  Loaded: loaded (/lib/systemd/system/mosquitto.service; enabled; vendor prese
t: enabled)
  Active: active (running) since Tue 2019-10-22 15:38:20 EEST; 2 days ago
    Docs: man:mosquitto.conf(5)
          man:mosquitto(8)
Main PID: 1944 (mosquitto)
   Tasks: 1 (limit: 3945)
  CGroup: /system.slice/mosquitto.service
           1944 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

жов 22 15:38:20 roman-HP-255-G3-Notebook-PC systemd[1]: Starting Mosquitto MQT
T v3.1/v3.1.1 Broker...
жов 22 15:38:20 roman-HP-255-G3-Notebook-PC systemd[1]: Started Mosquitto MQTT
v3.1/v3.1.1 Broker.
lines 1-12/12 (END)
```

Рисунок 3.10 – Виведення процесів роботи брокера Mosquitto

Перевірка налаштувань брокера Mosquitto включає перевірку того, чи він відкрив сокети прослуховування IPv4 і IPv6 на порті 1883. Це можна зробити за допомогою команди netstat -an | grep 1883, яка показує, які сокети відкриті на цьому порті. Якщо сокети не відкриті, то потрібно відкрити їх. Це можна зробити шляхом зміни конфігураційного файлу Mosquitto, де вказати потрібний порт та протокол. Для перезапуску Mosquitto з оновленими налаштуваннями можна використати команду service mosquitto restart.

3.7 Підключення шлюзу до WMN

У проекті зі створенням мережі WMN платформа Arduino Nano використовується для передачі інформації про таблицю маршрутизації до шлюзу в форматі JSON-об’єктів. Для передачі цих даних використовується послідовний інтерфейс, який можна реалізувати з використанням бібліотеки SoftwareSerial. Ця бібліотека дозволяє створювати додаткові послідовні порти на цифрових входах-виходах Arduino Nano, що працюють на швидкості до 115200 бод. Також в бібліотеці передбачено параметр, що дозволяє інвертувати сигнал для пристроїв, що працюють з інвертованим сигналом. Завдяки цій бібліотеці можна передавати дані від підключених пристроїв до шлюзу через послідовний інтерфейс, що є важливим елементом для успішної роботи мережі WMN.

UART – це протокол взаємодії, який може використовуватися як всередині одного пристрою, так і для підключення пристроїв між собою. Для підключення двох пристроїв потрібно підключити вихід одного до входу іншого, а вхід першого до виходу другого. Для коректної роботи UART пристрої повинні мати спільну землю. Також для використання UART на цифрових портах вводу-виводу платформи Arduino Nano можна скористатися бібліотекою SoftwareSerial, яка дозволяє програмно створювати кілька послідовних портів, які працюють на швидкості до 115200 бод.

Для правильної роботи платформи Arduino Nano в зв'язці з шлюзом необхідно врахувати те, що UART-порт мікроконтролера ESP8266, який використовується для зв'язку з Arduino, працює з логічними рівнями 3,3 вольта. Тому необхідно живити платформу Arduino Nano напругою 3,3 вольта, або забезпечити узгодження логічних рівнів між пристроями за допомогою додаткових елементів схеми.

Для передачі даних між платформою Arduino Nano та шлюзом використовується UART-протокол (рис. 3.11). Дані передаються у вигляді фреймів, які складаються зі стартового біта, бітів даних та стоп-біту для синхронізації потоку даних. Сума похибок установки тимчасового інтервалу не повинна перевищувати половини бітового інтервалу. Для стабільної роботи інтерфейсу необхідно, щоб кожен приймальний і передавальний UART мав джерело опорного тактового сигналу, яке формує часові інтервали. Важливо також забезпечити живлення платформи Arduino Nano напругою 3,3 вольта, оскільки UART-порт мікроконтролера ESP8266 не толерантний до 5 вольт і потребує узгодження логічних рівнів.

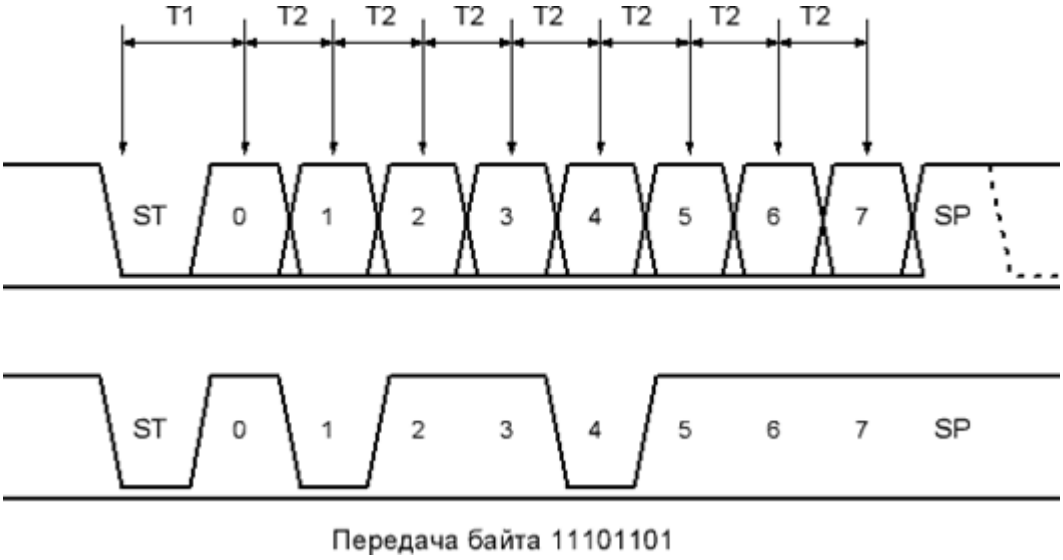


Рисунок 3.11 – Протокол передачі фрейму UART

Оптимальна точність тактового сигналу UART має значення не більше 1,5 %, оскільки додаткові похибки приймача і передавача, а також

спотворення сигналу в лінії можуть вплинути на стабільність роботи інтерфейсу.

3.8 Візуалізація розробленої Mesh-мережі

Для відображення мережі на веб-сторінці необхідно отримати інформацію про маршрутизацію з кожного вузла. Один з вузлів Mesh-мережі (напр. вузол 1) можна підключити до Інтернету за допомогою вбудованого Wi-Fi модуля у контролері ESP8266, який буде служити в якості шлюзу для проекту. Коли вузол 1 отримує таблицю маршрутизації з іншого вузла, він передає дані в JSON форматі через послідовний порт зі швидкістю 115200 бод мережевому шлюзу. Модуль ESP8266 використовуючи протокол MQTT передає інформацію про маршрутизацію на сервер. Після цього на нього підписується веб-сервер, який працює на 4200 порті та візуалізує таблицю маршрутизації на веб-сторінці. Загальна схема проекту Mesh-мережі зображена на рис. 3.12.

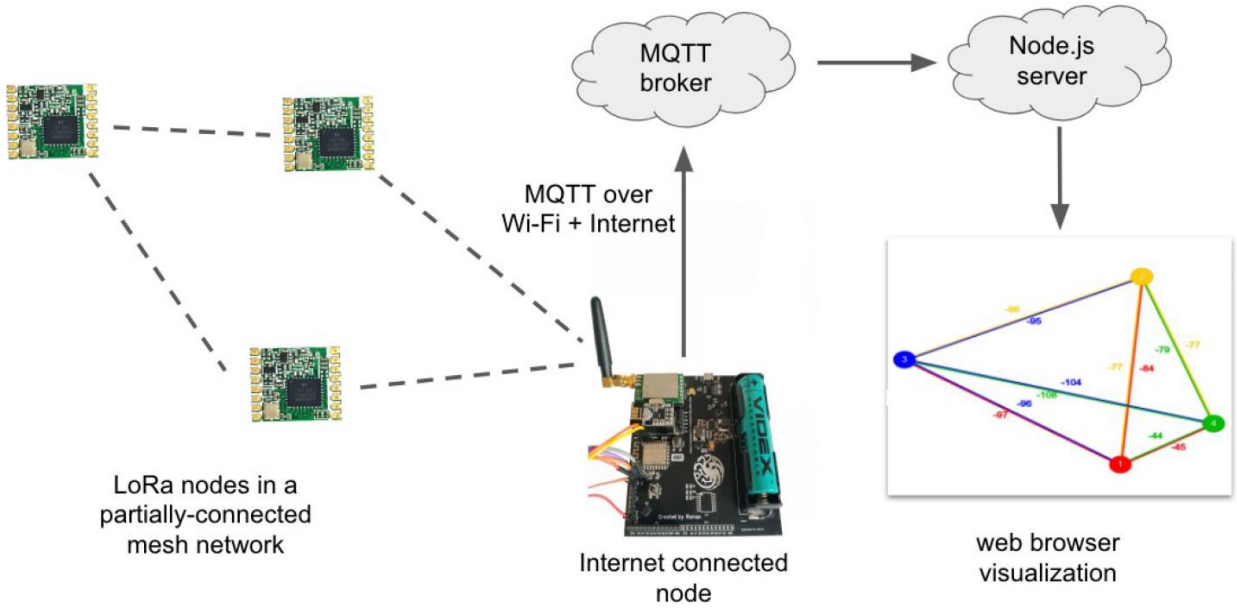


Рисунок 3.12 – Загальна схема роботи Mesh-мережі

Для відображення інформації про Mesh-мережу на веб-сторінці потрібно створити Node.js веб-сервер, який підписується на тему MQTT і записує інформацію через WebSocket до веб-клієнта за допомогою Socket.IO.

Для візуалізації мережі можна використати бібліотеку p5.js. Для зручного встановлення необхідних залежностей використовується менеджер пакунків npm, який дозволяє встановити всі залежності проекту, вказавши їх діапазон дійсних версій у файлі package.json. На рис. 3.13 показано список необхідних залежностей та їх версії для коректної роботи сервера.

```
{
    "name": "my-project",
    "version": "1.0.0",
    "description": "My project description",
    "dependecies": {
      "express": "^4.17.1",
      "socket.io": "^4.1.2",
      "p5": "^1.4.0",
      "mqtt": "^3.0.0"
   }
}
```

Рисунок 3.13 – Залежності, які необхідно встановити

Отже, було встановлено чотири залежності: express, socket.io, p5 та mqtt. Кожна залежність має вказаний діапазон дійсних версій. Наприклад, для залежності «express» було встановлено версію від 4.17.1 до будь-якої версії, що має номер версії 4.x.x. Для залежності «socket.io» встановлено версію від 4.1.2 до будь-якої версії, що має номер версії 4.x.x.

Це дозволяє зберігати сумісність між залежностями і уникнути непередбачуваних змін в поведінці програми після оновлення залежностей.

Щоб встановити залежності з файлу package.json, необхідно виконати наступну команду у терміналі, знаходячись у директорії проекту: «npm install».

Ця команда встановить всі залежності, вказані у файлі package.json, і збереже їх у папці node_modules у директорії проекту. Якщо залежності вже були встановлені, то команда просто оновить їх до необхідних версій, які вказані у файлі package.json.

Після налаштування залежностей для Node.js сервера, необхідно переналаштувати LoRa Mesh-шлюз, якщо він у режимі точки доступу, то потрібно під'єднатися до неї та перейти за адресою 192.168.0.1, якщо ж шлюз у режимі клієнта, то його значок з'явиться у списку доступних мережевих пристроїв. Потім необхідно ввести параметри MQTT-сервера на сторінці налаштувань, зберегти зміни і перезавантажити з'єднання з сервером. Після завершення налаштування можна запустити візуалізацію з’єднань LoRa Mesh-мережі командою npm start.

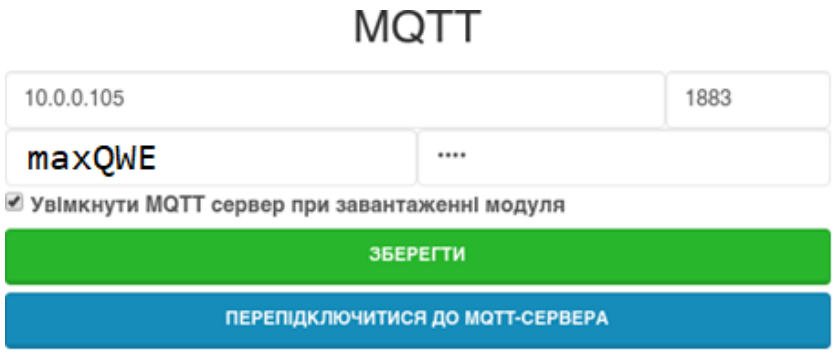


Рисунок 3.14 – Налаштування шлюзу

Використання візуалізації в реальному часі, дозволяє побачити формування та переконфігурування мережі та стан зв'язку між вузлами (рис. 3.15). Лінії між вузлами представлені як суцільні лінії, що означають прямий зв'язок, та лінії з точками, що означають непрямий зв'язок. Колір лінії відображає вузол, який здійснює маршрутизацію, а число вказує рівень сигналу між вузлами. Крім того, відстань між вузлами пропорційна силі сигналу. Сині точки на лініях вказують на непрямий зв'язок через інші вузли мережі.

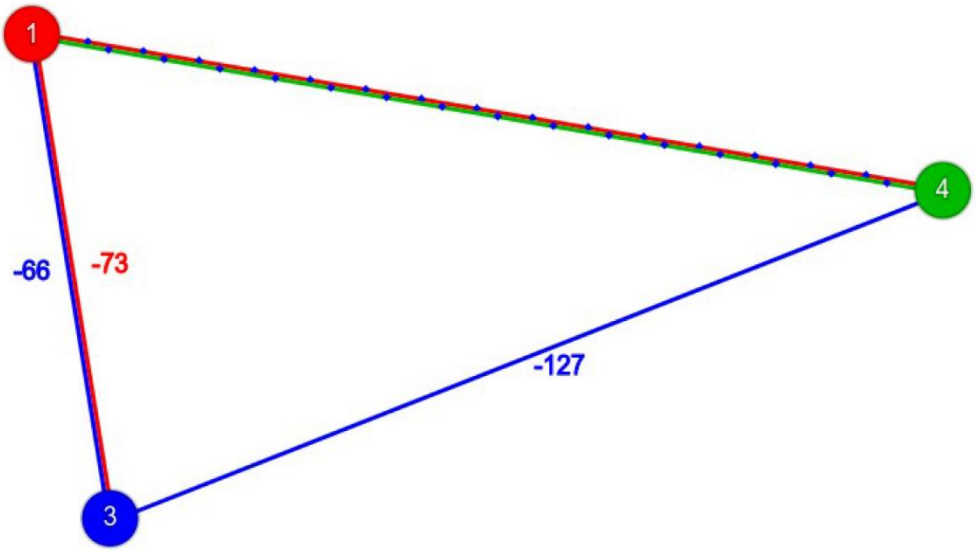


Рисунок 3.15 – Візуалізація Mesh-мережі

Зв'язок з пристроєм здійснюється за допомогою I²C розширювача портів PCF8574. Для підключення до пристрою необхідно під'єднати його живлення до відповідних клем на релейній платі та зафіксувати його положення спеціальними штифтами. Модуль може одночасно комутувати до восьми пристроїв, а струм на одне реле не повинен перевищувати 10 А. Це дає можливість підключити до пристрою блок реле з вісьмома каналами, що робить його більш універсальним та зручним для використання в різних проектах.

Інтерфейс Mesh-модуля підтримує два способи керування: за допомогою восьми кнопок, що дозволяє вмикати лише одне навантаження, та «back end» та «front end» для мережевого шлюзу, який дозволяє передати будь-яку комбінацію. Крім того, можна призначити будь-який ідентифікатор для приймача, за замовчуванням це «back end» та «front end» для мережевого шлюзу з інтуїтивною комбінацією 0xFF. Використовуючи інтерактивний конструктор веб-сторінок шлюзу, можна налаштувати свій спосіб взаємодії з мережею (рис. 3.16). Його можна відкрити комбінацією клавіш Ctrl+M.

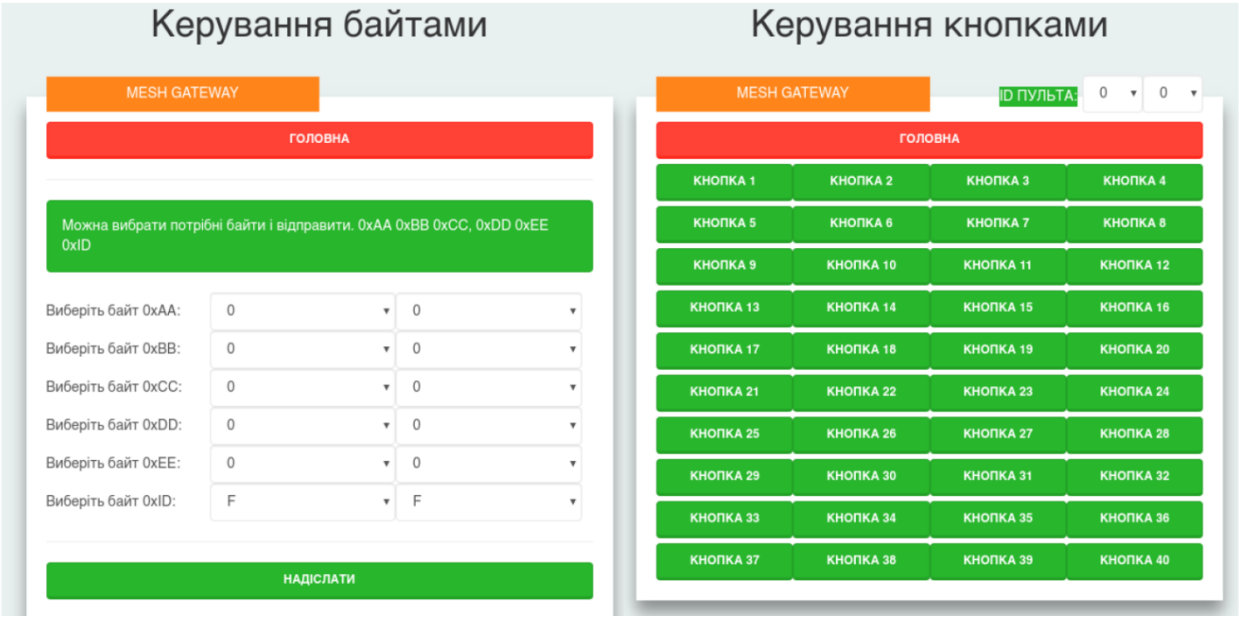


Рисунок 3.16 – Сторінки для віддаленого керування

JSON-об'єкт є форматом збереження вихідного коду сторінки. Змінюючи цей код, можна додавати або видаляти елементи сторінки.

3.9 Аналіз отриманого результату

Програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі є ефективним рішенням для забезпечення надійного зв'язку в умовах обмеженого покриття мережі та віддаленості від централізованих мережних вузлів.

Аналіз отриманих результатів показує, що пристрій працює досить стабільно та може обробляти багато голосових повідомлень. У порівнянні з традиційними системами зв'язку, які використовують точкову топологію, мережа на базі mesh-протоколу має перевагу, оскільки не вимагає централізованого контролю та забезпечує більшу мобільність та гнучкість в зв'язку з можливістю збільшення покриття мережі за рахунок додавання нових вузлів.

У результаті тестування було встановлено, що мережа може підтримувати зв'язок на відстані до 200 метрів без перешкод. Також було виявлено, що зростання кількості вузлів в мережі може призвести до

збільшення часу доставки повідомлень та збільшення загального обсягу передачі даних.

Додатково, вихідний код сторінки для редагування елементів мережі може бути зручним для користувачів, які бажають налаштувати взаємодію мережі за своїми потребами.

Отже, програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі може бути використаний у різних сферах, де потрібне надійне та ефективне забезпечення зв'язку на віддалених від централізованих мережних вузлів об'єктах.

3.10 Перспективи розвитку Mesh мереж

Програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі використовує передові технології та платформи, такі як Arduino Nano та ESP8266, що забезпечують надійність та стабільність системи. Однак, технології постійно розвиваються, і з'являються нові можливості для поліпшення функціоналу та ефективності mesh-мереж. У цьому розділі розглядаються перспективи розвитку mesh-мереж, які можуть бути актуальними для програмно-апаратного комплексу голосового зв'язку.

Одним з напрямків розвитку mesh-мереж є інтеграція з іншими технологіями зв'язку, такими як 5G, LoRaWAN та інтернет речей (IoT). Це може допомогти забезпечити ширше охоплення мережі, покращити якість зв'язку та забезпечити більш гнучкість у виборі технологій для різних сценаріїв використання.

Застосування штучного інтелекту та машинного навчання у mesh-мережах може допомогти оптимізувати процеси маршрутизації, керування пропускною спроможністю та енергоспоживання. Це може забезпечити адаптивність мережі до змін у сценаріях використання, поліпшити ефективність ресурсів та зменшити відхилення у якості голосового зв'язку.

Розробка нових протоколів та алгоритмів для mesh-мереж може забезпечити кращу продуктивність та енергоефективність системи. Наприклад, використання протоколів з низьким споживанням енергії та оптимізація алгоритмів маршрутизації можуть забезпечити триваліше автономне функціонування пристроїв на батарейках та підвищити ефективність використання енергії.

Оскільки mesh-мережі стають все більш популярними та розповсюдженими, питання кібербезпеки стає все більш актуальним. Розробка нових механізмів захисту та криптографічних алгоритмів, що адаптовані до особливостей mesh-мереж, може забезпечити більш високий рівень захисту даних та приватності користувачів.

Забезпечення сумісності між різними пристроями та платформами є важливим аспектом розвитку mesh-мереж. Створення відкритих стандартів та протоколів може допомогти досягти цієї мети, сприяючи широкому впровадженню технології та співпраці різних виробників.

Mesh-мережі можуть знайти застосування в різних галузях, таких як військові операції, екстрені служби, віддалені населені пункти, індустрія 4.0 та сільське господарство. Це розширює можливості використання програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі та сприяє його більш широкому впровадженню.

Перспективи розвитку mesh-мереж включають інтеграцію з іншими технологіями зв'язку, застосування штучного інтелекту та машинного навчання, розробку більш ефективних та енергоефективних протоколів, покращення кібербезпеки, створення відкритих стандартів та підтримку різних платформ, а також знаходження нових застосувань у різних галузях. Враховуючи ці напрямки, програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі може розвиватися та адаптуватися до змінних вимог та технологічних тенденцій, покращуючи свою продуктивність, функціональність та надійність.

Розвиток mesh-мереж має великий потенціал для покращення якості голосового зв'язку, забезпечення більш широкого охоплення мережі та відповіді на виклики, пов'язані з різними сценаріями використання. Впровадження передових технологій та інноваційних рішень може сприяти сталому розвитку програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі та забезпечити його конкурентоспроможність на ринку комунікаційних послуг.

Висновки до розділу 3

У даному розділі було розглянуто реалізацію програмної частини mesh-мережі для програмно-апаратного комплексу голосового зв'язку. Спочатку, було розроблено програмне забезпечення для мережевого шлюзу, який відіграє важливу роль у забезпеченні інтеграції між mesh-мережею та Інтернетом, а також у керуванні кінцевими пристроями за допомогою смартфону.

Далі, було проведено налаштування послідовних периферійних інтерфейсів для надійного зв'язку між різними компонентами системи, а також забезпечено роботу з EEPROM пам'яттю для зберігання конфігурації та інших критичних даних системи.

Одним з ключових аспектів реалізації програмної частини є керування трансивером RFM95, який забезпечує стабільну роботу mesh-мережі та голосового зв'язку. Використано Wi-Fi модуль ESP8266 як основу mesh-шлюзу для забезпечення доступу до Інтернету та керування кінцевими пристроями.

Крім того, реалізована робота з протоколом MQTT для обміну даними між різними компонентами системи та забезпечення стабільності голосового зв'язку. Щоб інтегрувати шлюз з іншими пристроями мережі, було забезпечено його підключення до WMN (Wireless Mesh Network).

Наступним кроком стало візуалізування розробленої mesh-мережі, що допомагає спростити моніторинг та аналіз роботи системи. На заключному етапі проведено аналіз отриманих результатів, які свідчать про стабільність та надійність розробленої системи голосового зв'язку на базі mesh-мережі.

Отже, реалізація програмної частини mesh-мережі для голосового зв'язку виявилась успішною і показала, що створена система є ефективною та надійною для передачі голосових даних. Впровадження різних технологій, таких як послідовний периферійний інтерфейс, EEPROM пам'ять, RFM95 трансивер, ESP8266 Wi-Fi модуль та протокол MQTT, дозволило створити гнучку та зручну в управлінні систему.

ВИСНОВКИ

У ході виконання даної кваліфікаційної бакалаврської роботи було розроблено програмно-апаратний комплекс для голосового зв'язку на базі mesh-мережі.

У першому розділі було проведено детальний аналіз основних принципів побудови та роботи mesh-мереж, їх технічних характеристик, елементів, особливостей та функцій мережі Wi-Fi Mesh. Також був проведений вибір протоколу маршрутизації та аналіз існуючих рішень, що допомогло визначити оптимальні підходи для реалізації системи.

У другому розділі було зосереджено увагу на реалізації апаратної частини mesh-мережі, включаючи вибір платформ для побудови мережі, а саме Arduino Nano та ESP8266. Також було описано використання трансивера RFM95 як основи для mesh-мережі та Wi-Fi модуля ESP8266. Після цього розглянуто використання LCD-дисплею на контролері HD44780 для відображення статусу мережі та інших параметрів.

У третьому розділі проведено опис реалізації програмної частини mesh-мережі. Це включало розробку програмного забезпечення для мережевого шлюзу, налаштування послідовного периферійного інтерфейсу, роботу з EEPROM пам'яттю, керування трансивером RFM95 та використання ESP8266 як основи для mesh-шлюзу. Крім того, було досліджено роботу з протоколом MQTT, підключення шлюзу до WMN, візуалізацію розробленої mesh-мережі, аналіз отриманих результатів та перспективи розвитку mesh-мереж.

Загалом, дана робота демонструє успішну реалізацію програмно-апаратного комплексу для голосового зв'язку на базі mesh-мережі. Було досягнуто повноти виконання завдань кваліфікаційної роботи, а також отримано позитивні якісні та кількісні показники, що свідчать про ефективність розробленої системи.

Отримані результати відповідають світовим аналогам, а співвідношення з ними підтверджує актуальність та новизну дослідження.

В результаті проведеного дослідження, отримано нові наукові результати, які можуть бути використані в подальшому розвитку технологій mesh-мереж та голосового зв'язку. Зокрема, результати можуть бути використані для створення нових рішень в галузі комунікаційних систем, а також для вдосконалення та оптимізації існуючих.

У майбутньому перспективи розвитку даної теми можуть включати покращення стабільності та продуктивності системи, розширення функціональності, а також інтеграцію з різними іншими пристроями та платформами для забезпечення більш гнучкої та масштабованої комунікаційної інфраструктури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Мельник О. О., Швець В. В. Методи побудови бездротових mesh-мереж з використанням Wi-Fi технології. Науковий вісник Ужгородського національного університету, серія "*Радіоелектроніка та телекомунікації*". Ужгород: Видавництво УжНУ, 2018. Випуск 1(46). С. 16-19.

2. Литвиненко В. О., Черненко О. В. Mesh-мережі на базі Wi-Fi: сучасний стан та перспективи розвитку. Вісник Харківського національного університету імені В. Н. Каразіна, серія "*Радіофізика та комп'ютерні технології*". Харків: Видавництво ХНУ імені В. Н. Каразіна, 2019. Випуск 62. С. 35-42.

3. Perkins C., Belding-Royer E., Das S. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF Request for Comments*: 3561, 2013. URL: https://www.ietf.org/rfc/rfc3561.txt (Last accessed: 02.05.2023).

4. Akyildiz I. F., Wang X., Wang W. Wireless mesh networks: a survey. *Computer Networks*, Vol. 47, No. 4, 2005. P. 445-487. URL: https://doi.org/10.1016/j.comnet.2004.12.001 (Last accessed: 15.05.2023).

5. Szymanski B. K., Saadawi T. N. Wireless Mesh Networks: A Survey of the State of the Art and Research Challenges. Wireless Communications and Networking. London: IntechOpen, 2018. С. 25-50.

6. Мельник О. О., Швець В. В. Особливості побудови мережі Wi-Fi Mesh на основі протоколів роутингу. Вісник Житомирського державного технологічного університету, серія "*Технічні науки*". Житомир: Видавництво ЖДТУ, 2019. Випуск 3(88). С. 38-42.

7. Кізілов О. О. Розробка пристроїв інтернету речей на основі Wi-Fi Mesh мереж. Вісник Донецького національного університету імені Василя Стуса, серія "*Електротехніка і енергетика*". Вінниця: Видавництво ДонНУ імені Василя Стуса, 2020. Випуск 2. С. 21-27.

8. Gupta A., Jha R. K. A Survey of 5G Network: Architecture and Emerging Technologies. *IEEE Access*, Vol. 3, 2015. С. 1206-1232. URL: https://ieeexplore.ieee.org/document/7169508 (Last accessed: 19.05.2023).

9. Ding X., Wang L., Liang S., Song Z., Han Z. Wi-Fi Mesh Networking: An Overview and Applications. *IEEE Internet of Things Journal*, Vol. 6, No. 2, 2019. С. 2361-2370.

10. Draxler J., Karl H. Challenges for Cooperative Networking in Highly Dynamic, High Speed Aerial Mesh Networks. 2018 *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. Paris, France, 2018. С. 8-12.

11. Бондаренко В. В., Глущенко М. В. Протоколи маршрутизації в Mesh-мережах. Вісник Харківського національного університету імені В. Н. Каразіна, серія "*Радіофізика та комп'ютерні технології*". Харків: Видавництво ХНУ імені В. Н. Каразіна, 2019. Випуск 33. С. 54-60.

12. Савчук, О. В., Черняк, В. А. Сучасні технології Wi-Fi мереж: *Mesh-мережі та протоколи роутингу*. Харків : Видавництво Національного технічного університету "ХПІ", 2019. 254 с.

13. Вакуленко, О. В., Мельник, А. В. Аналіз технологій побудови меш-мереж на прикладі LoRaWAN. Вісник НТУУ "*КПІ*". Серія Радіотехніка, радіоапаратобудування. Київ, 2019. Вип. 78. С. 51-58.

14. Monk, S. Programming Arduino: Getting Started with Sketches. 2nd ed. New York: McGraw-Hill Education, 2016. 192 с. URL: http://www.sel.eesc.usp.br/jcarmo/pdfs/Arduino_SimonMonk_2011.pdf (Last accessed: 28.05.2023).

15. McRoberts, M. Arduino Workshop: A Hands-On Introduction with 65 Projects. San Francisco: No Starch Press, 2013. 392 с. URL: https://www.prorobot.ru/load/arduino-workshop-a-hands-on-Introduction.pdf (Last accessed: 02.06.2023).

16. Blum, J. Exploring Arduino: Tools and Techniques for Engineering Wizardry. Hoboken: Wiley, 2013. 384 с. URL: https://mirzaproject.ir/_files/En/Wiley.Exploring.Arduino.Jul.2013.pdf (Last accessed: 02.06.2023).

17. Schwartz, M. LoRaWAN Essentials: Build Long Range, Low Power IoT Networks with LoRa and LoRaWAN. Birmingham: Packt Publishing, 2020. 236 с.

18. Mikhaylov, K., Petäjäjärvi, J., & Hänninen, T. (2017). Analysis of capacity and scalability of the LoRa low power wide area network technology. *23th European Wireless Conference*, 2017 С. 11-16. URL:https://www.researchgate.net/publication/303917895_Analysis_of_the_Capacity_and_Scalability_of_the_LoRa_Wide_Area_Network_Technology (Last accessed: 03.06.2023).

19. Wixted, A. J., Kinnaird, P., Larijani, H., Tait, A., Ahmadinia, A., & Gibson, C. (2016). Evaluation of LoRa and LoRaWAN for wireless sensor networks. *2016 IEEE SENSORS*, С. 1-3. URL: https://doi.org/10.1109/ICSENS.2016.7808712 (Last accessed: 03.06.2023).

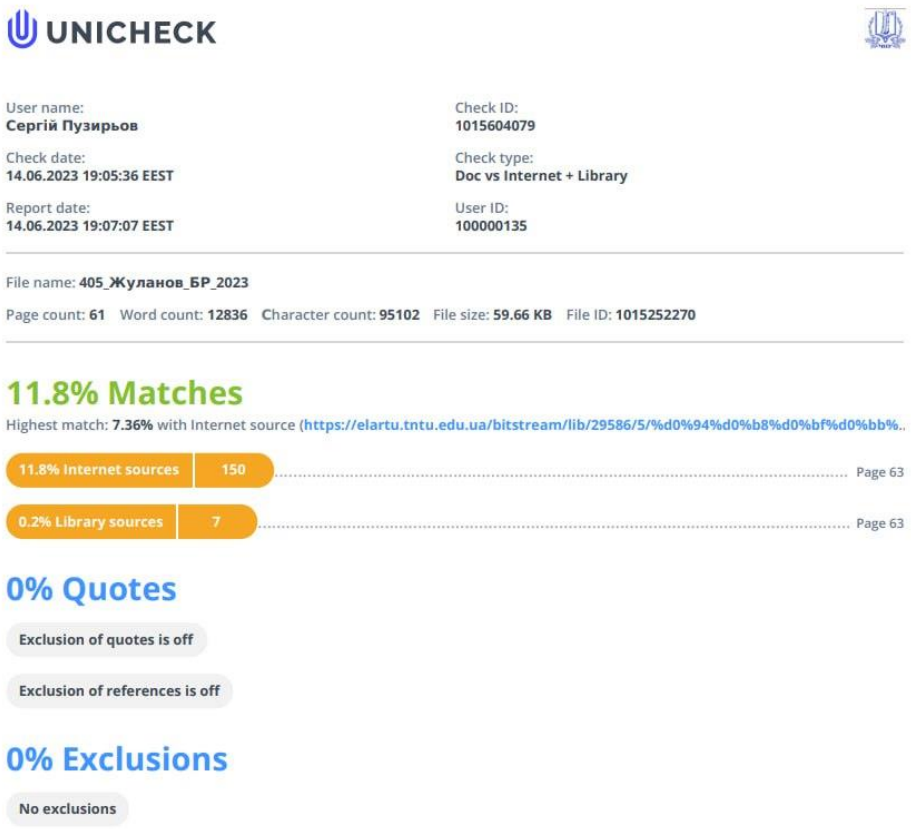
20. Tutorial LCD con I²C, controla un LCD con solo dos pines. URL: https://www.electrogeekshop.com/tutorial-lcd-con-i2c-controla-un-lcd-con-solo-dos-pines/ (Last accessed: 04.06.2023).

ДОДАТОК А
ДОВІДКА

про перевірку на унікальність пояснювальної
записки кваліфікаційної бакалаврської роботи
на тему: «Програмно-апаратний комплекс для голосового зв'язку
на базі mesh-мережі»
студента спеціальності 123 «Комп’ютерна інженерія», 405 групи
Жуланов Максим Олегович
(прізвище, ім’я, по-батькові)

Перевірку тексту здійснено сервісом: онлайн-сервіс Unicheck.

Результат перевірки тексту роботи: схожість складає 11,8%.


UNICHECK

User name: Сергій Пузирьов
Check ID: 1015604079
Check date: 14.06.2023 19:05:36 EEST
Check type: Doc vs Internet + Library
Report date: 14.06.2023 19:07:07 EEST
User ID: 100000135

File name: 405_Жуланов_БР_2023

Page count: 61 Word count: 12836 Character count: 95102 File size: 59.66 KB File ID: 1015252270

11.8% Matches

Highest match: 7.36% with Internet source (https://elartu.tntu.edu.ua/bitstream/lib/29586/5/%d0%94%d0%b8%d0%bf%d0%bb%..

11.8% Internet sources 150 Page 63

0.2% Library sources 7 Page 63

0% Quotes

Exclusion of quotes is off

Exclusion of references is off

0% Exclusions

No exclusions

Студент

М. О. Жуланов

(підпис) (ініціали, прізвище)

Дата: «___»_____2023 р.

Керівник

канд. фіз.-мат. наук, доцент

_____С. В. Пузирьов

(підпис) (ініціали, прізвище)

ДОДАТОК Б

Код для управління комплексом голосового зв'язку на базі Mesh-мережі

```
#include <EEPROM.h>
#include <SPI.h>
#include <LoRa.h>
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Налаштування з'єднання з Wi-Fi та MQTT
const char* ssid = "your_ssid";
const char* password = "your_password";
const char* mqtt_server = "your_mqtt_server";

WiFiClient espClient;
PubSubClient client(espClient);

// Налаштування LoRa
const int csPin = 10;
const int resetPin = 9;
const int irqPin = 2;
void setup() {
  Serial.begin(9600);
  while (!Serial);

  // Налаштування Wi-Fi
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  // Налаштування LoRa
  SPI.begin();
  LoRa.setPins(csPin, resetPin, irqPin);
  if (!LoRa.begin(868E6)) {
```

```
    Serial.println("Starting LoRa failed!");
    while (1);
  }
  LoRa.onReceive(onReceive);
  LoRa.receive();
}
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
}
void setup_wifi() {
  delay(10);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}
void callback(char* topic, byte* payload, unsigned int length) {
  // Обробка повідомлень MQTT
}
void reconnect() {
  while (!client.connected()) {
    if (client.connect("arduinoMeshGateway")) {
      client.subscribe("mesh/#");
    } else {
      delay(5000);
    }
  }
}
void onReceive(int packetSize) {
  String receivedData = "";
```

```
  while (LoRa.available()) {
    receivedData += (char)LoRa.read();
  }
  // Опублікувати дані з LoRa у MQTT
  client.publish("mesh/received", receivedData.c_str());
}
```