Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломної роботи

магістра
(освітній ступінь)

на тему: Система контролю розумним будинком з використанням LoRa-Mesh-технологій

Виконав: студент (ка) 6 курсу, групи СІм-61
спеціальності 123
„Комп'ютерна інженерія"
(шифр і назва спеціальності)

(підпис) Бунтій Р.А.
(прізвище та ініціали)

Керівник (підпис) Кравець О.Й.
(прізвище та ініціали)

Нормоконтроль (підпис) Бунтій Р.А.
(прізвище та ініціали)

Рецензент (підпис) Чайковський І.В.
(прізвище та ініціали)

м. Тернопіль – 2019

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)

Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра комп'ютерних систем та мереж
Освітній ступінь магістр
Напрям підготовки
(шифр і назва)
Спеціальність 123 Комп'ютерна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КС
Осухівська Р.П.
«30» 09 2019 р.

ЗАВДАННЯ
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Бузню Роману Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система контролю розумних будинків з використанням LoRa-Mesh-технологій

Керівник роботи Ярошук Олександр Васильович к.ф.-м.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від «27» 09 2019 року №411-85ч

2. Термін подання студентом роботи 23 грудня 2019 р.

3. Вихідні дані до роботи спосіб об'єднати технології LoRa та Mesh для підвищення відмовостійкості, архітектура та визначення забезпечення системи контролю розумних будинків, макети системи контролю розумних будинків та програмне забезпечення для нього

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Основні відомості та принципи побудови LoRa Mesh-мережі
2. Загальна характеристика технології LoRa Mesh-мережі
3. Програмна програмна частина LoRa Mesh-мережі
4. Обґрунтування економічної ефективності проекту
5. Охорона праці та безпеки в надзвичайних ситуаціях
6. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
актуальність теми, мета і завдання дослідження, об'єкт, предмет, методи дослідження, наукова новизна і практична цінність одержаних результатів, основні етапи роботи, порівняльна характеристика існуючих і пропонованої технологій, будова мережі, архітектура LoRa Mesh-мережі, принцип відправки роботи LoRa Mesh-мережі, організація таблиці маршрутизації в JSON-форматі, приклади створення LoRa Mesh-мережі

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економіка	Лесюта О.М., доц.		
Обґрунт. екон. ефект.	Кирєєв Н.Б.		
Безпека в НС	Сергучов В.С., ст. викл. каф. ОХ		
Охорона праці	Осуйська Г.М.	24.11.19	

7. Дата видачі завдання 30.09.2019

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд публікацій авторів, які досліджували проблематику системи контролю розумних будинків	1.10.2019 - 6.10.2019	виконано
2.	Аналіз переваг та недоліків існуючих систем	7.10.2019 - 8.10.2019	виконано
3.	Аналіз існуючих протоколів маршрутизації для Систем розумного будинку	9.10.2019 - 13.10.2019	виконано
4.	Вибір оптимального протоколу маршрутизації для системи розумного будинку	20.10.2019 - 25.10.2019	виконано
5.	Вибір комплектної бази	26.10.2019 - 5.11.2019	виконано
6.	Реалізація апаратної частини LoRa мезх-мережі	6.11.2019 - 10.11.2019	виконано
7.	Розробка архітектури та алгоритмічного забезпечення для системи контролю розумних будинків	11.11.2019 - 14.11.2019	виконано
8.	Створення проекту системи контролю розумних будинків з використанням LoRa радіомодулів	15.11.2019 - 19.11.2019	виконано
9.	Написання програмного забезпечення для контролера та Wi-Fi модуля	20.11.2019 - 23.11.2019	виконано
10.	Тестування розробленої системи	24.11.2019 - 1.12.2019	виконано
11.	Аналіз економічної ефективності внесених розробок	2.12.2019	виконано
12.	Оформлення розділів «Охорона праці та безпека в надзвичайних ситуаціях» і «Економіка»	3.12.2019 - 8.12.2019	виконано
13.	Перевірка на плагіат	9.12.2019	виконано
14.	Оформлення графічної частини	10.12.2019 - 17.12.2019	виконано
15.	Рецензування роботи	18.12.2019	виконано
16.	Захист роботи	23.12.2019	виконано

Студент _____ (підпис) Бузій Т.А. (прізвище та ініціали)

Керівник роботи _____ (підпис) Крамар О.Й. (прізвище та ініціали)

АНОТАЦІЯ

Система контролю розумним будинком з використанням LoRa–Mesh–технологій // Дипломна робота // Буцій Роман Андрійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, група СІм-61 // Тернопіль, 2019 // с. – 143, рис. – 51, табл. – 7, аркушів А1 – 10, додат. – 2, бібліогр. – 39.

Ключові слова: РОЗУМНИЙ БУДИНОК, БЕЗДРОТОВА MESH-ТЕХНОЛОГІЯ.

Метою роботи є розробка апаратно-програмної системи контролю розумним будинком з використанням LoRa Mesh-технологій.

У дипломній роботі магістра проаналізовано сучасні системи контролю розумним будинком. Проведено аналіз протоколів маршрутизації для бездротового зв'язку. Для підвищення завадостійкості та масштабованості системи, запропоновано спосіб поєднання Mesh і LoRa технологій.

Проведено аналіз представлених технологій та розроблено архітектуру системи контролю розумним будинком з їх використанням. На її базі створено та протестовано програмне забезпечення для функціонування системи. Представлено спосіб візуалізації формування таблиць маршрутизації безпровідної LoRa Mesh-мережі з використанням протоколу MQTT.

Розроблено "back end" та "front end" для мережевого шлюзу з інтуїтивно зрозумілим веб-інтерфейсом.

ANNOTATION

Smart house control system using LoRa-Mesh technologies // Master diploma thesis // Roman Butsiy // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information System and Software Engineering, group SIm-61 // Ternopil, 2019 // p. – 143, fig. – 51, tab. – 7, sheets A1. – 10 , addit. – 2, bibliography. – 39.

Key words: SMART HOME, WIRELESS MESH TECHNOLOGY.

The purpose of the work is to develop a hardware and software control system for a smart home using LoRa Mesh technologies.

In the master's thesis the modern control systems of the smart house are analyzed. The analysis of routing protocols for wireless communication is conducted. To improve system resilience and scalability, a combination of Mesh and LoRa technologies is proposed.

The presented technologies were analyzed and the architecture of the smart home control system was developed with their use. The software required by the aforementioned system to function is written and tested on its basis. A method of visualizing the formation of routing tables of a wireless LoRa-Mesh network using the MQTT protocol is presented.

Back-end and front-end for network gateway with intuitive web interface is developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІМ	Широтно-імпульсна модуляція
AODV	Ad hoc On-Demand Distance Vector
ASCII	American Standard Code for Information Interchange
CMOS	Complementary Metal-Oxide-Semiconductor
CRC	Cyclic redundancy check
DSR	Dynamic Source Routing
EEPROM	Electrically Erasable Programmable Read-Only Memory
FHSS	Frequency Hopping Spectrum Spreading
FIFO	First In, First Out
FSK	Frequency Shift Keying
GCC	GNU Compiler Collection
GFSK	Gaussian Frequency-Shift Keying
HWMP	Hybrid Wireless Mesh Protocol
HWMP	Hybrid Wireless Mesh Protocol
I^2C	Inter-Integrated Circuit
ICSP	In-Circuit Serial Programming
IIP_3	Third-order intercept point
JSON	JavaScript Object Notation
LoRa	Long Range
MANET	Mobile Ad hoc Network
MIPS	Million Instructions Per Second
MQTT	Message Queue Telemetry Transport
MSK	Minimum-Shift Keying
npm	Node Package Manager

OLSR	Optimized Link State Routing Protocol
OOK	On-Off Keying
RISC	Reduced Instruction Set Computer
RSSI	Received Signal Strength Indication
SDK	Software Development Kit
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WMN	Wireless Mesh Network
ZRP	Zone Routing Protocol

ВСТУП

Актуальність теми. В наш час інформаційні технології інтенсивно інтегруються у наше повсякденне життя. На сьогоднішній день у багатьох будинках вже можна зустріти так звані “розумні” системи, до яких висувається ряд вимог, зокрема: стабільність, зручність у користуванні, надійність, та низька ціна. Найважливішою проблемою при функціонуванні таких систем є низька завадостійкість та невеликий радіус дії.

Існуючі системи розумних будинків проектують з використанням стандарту IEEE 802.11a/b/g/n (Wi-Fi). Пристрої, які створені з використанням цього стандарту, мають ряд недоліків, зокрема, високу споживчу потужність та необхідність великих обчислювальних ресурсів для функціонування стеку протоколів 802.11.

У зв’язку з необхідністю усунення подібних проблем все більшого попиту починають набирати Mesh-технології. У Mesh-мережі всі вузли рівноправні, що дозволяє відмовитися від центрального вузла, і при відмові будь-якого кінцевого пристрою, мережа може самостійно переконфігуруватися. Саме тому застосування Mesh-топології у системах розумного будинку дозволить відмовитися від маршрутизаторів, а технологія LoRa дасть змогу зменшити енергоспоживання та підвищити відмовостійкість таких системи загалом.

Принципи Mesh-технологій запропоновано і досліджено у працях значної кількості зарубіжних вчених, серед яких Yu Liu, Wang X., Kin-Fai Tong та багато інших.

Мета і завдання дослідження

Метою роботи є розробка апаратно-програмної системи контролю розумним будинком з використанням LoRa Mesh-технологій.

Для досягнення поставленої мети необхідно виконати такі задачі:

- провести огляд публікацій авторів, які займались дослідженням проблематики систем контролю розумним будинком та проаналізувати їх;

- сформулювати базові вимоги до системи контролю розумним будинком з використанням LoRa Mesh-технологій;

- обґрунтувати методи використання LoRa Mesh-технологій в системі контролю розумним будинком;

- розробити архітектуру та алгоритмічне забезпечення системи контролю розумним будинком з використанням запропонованих методів;

- розробити макет системи контролю розумним будинком з використання LoRa радіо-модуля;

- розробити програмне забезпечення для контролера та WI-FI модуля запропонованої системи контролю розумним будинком;

- протестувати та оцінити завадостійкість модуля безпровідного зв'язку для системи контролю розумним будинком.

Об'єкт дослідження: процес контролю розумним будинком.

Предметом дослідження: система контролю розумним будинком з використанням LoRa Mesh - технологій.

Методи дослідження. Методи дослідження контролю розумним будинком з використанням LoRa Mesh - технологій базуються на основі використання методів системного аналізу, імітаційного моделювання, теорії проектування обчислювальних систем та комп'ютерних мереж, а також математичного та комп'ютерного моделювання.

Наукова новизна одержаних результатів

- Запропоновано модифікований спосіб поєднання технологій LoRa і Mesh для підвищення відмовостійкості та завадостійкості бездротової мережі.

- Розроблено архітектуру та власний прототип системи контролю розумним будинком на основі поєднання цих технологій.

Практичне значення одержаних результатів

Отримані результати дозволять розробити систему контролю розумним будинком, яку буде легко розгорнути, масштабувати і, яка не буде втрачати своєї працездатності при відмові окремих вузлів системи.

Публікації. Результати роботи апробовано на II Міжнародній студентській науково-технічній конференція “Природничі та гуманітарні науки. актуальні питання” м. Тернопіль 25-26 квітня 2019 року, VIII Міжнародній науково-технічної конференції молодих учених та студентів “Актуальні задачі сучасних технологій” м. Тернопіль 27-28 листопада 2019 року та VII науково-технічній конференції Тернопільського національного технічного університету імені Івана Пулюя “Інформаційні моделі, системи та технології”. м. Тернопіль 11-12 грудня 2019 року.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, шести розділів, висновків, список використаних джерел та додатків. Обсяг роботи: пояснювальна записка – 143 аркуша формату А4, графічна частина – 10 аркушів формату А1.

РОЗДІЛ 1
ОСНОВНІ ВІДОМОСТІ ТА ПРИНЦИПИ ПОБУДОВИ LoRa MESH-МЕРЕЖ

1.1. Загальні відомості про Mesh-мережі

Бездротові Mesh-мережі (WMNs) - це нова технологія, яка впроваджується як рішення для бездротового зв’язку в системи розумних будинків. Самоорганізованість є одним з ключових принципів побудови таких мереж. При відмові окремих компонентів такої мережі вона здатна переконфігуруватися та відновити свою працездатність. Відомо, що WMN здатні самостійно переорганізуватися при підключені нових вузлів, тому розгортання такої мережі може коштувати набагато дешевше, оскільки вони не вимагають дорогої інфраструктури. Також то переваг даної технології можна віднести високу надійність, оскільки при виході з ладу одного з вузлів навантаження розподіляється на сусідні.

Топологія WMN (рис. 1.1) базується на децентралізованій організації мережі, на відміну від типових мереж 802.11a/b/g, які створюються за централізованим принципом [1].

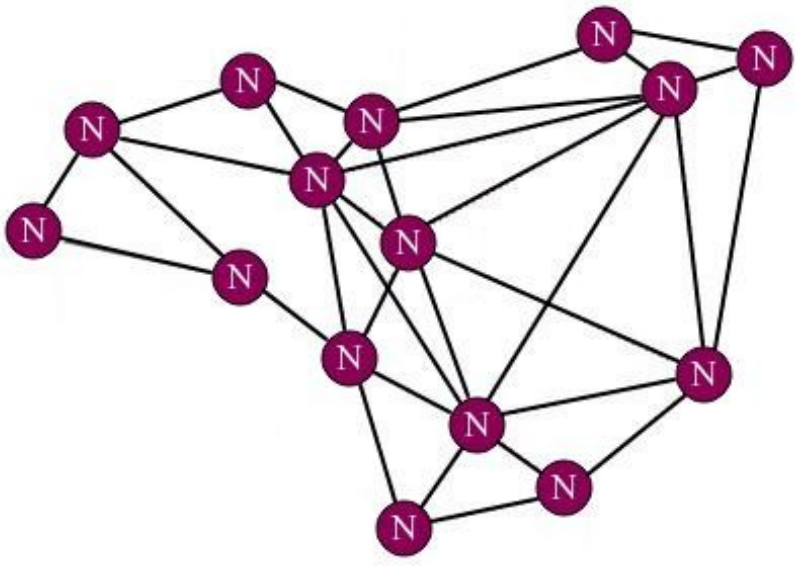

Рис. 1.1. Приклад топології Mesh-мережі

WMN умовно можна розділити на стаціонарні та мобільні. Останні можуть часто змінювати місце свого розташування. Насамперед Mesh-мережі - це багаторівневі мережі, кожен вузол якої має функції маршрутизатора, тобто здатний використати різні шляхи для передачі пакетів. Ця особливість відкриває можливість створювати самовстановлювальні та самовідновлювальні сегменти мережі. Зручно WMN будувати, як сукупність кластерних зон, число яких теоретично може бути не обмежене.

Більшість таких систем засновані на технології IEEE 802.11a/b/g/n. Недоліком пристроїв побудованих з використанням даного стандарту є висока споживана потужність, невеликий радіус дії, низька завадостійкість та громіздкий стек протоколів 802.11, який потребує значних обчислювальних ресурсів для його підтримки. Рішенням цієї проблеми є використання стандарту IEEE802.15.4g. Це стандарт для бездротового малопотужного зв’язку, який спочатку був розроблена для розумних комунальних мереж, тобто, для підключення розумних лічильників. Пізніше на його базі було засновано технологію LoRa.

1.2. Особливості технології LoRa

LoRa - це технологія яка дозволяє створювати мережі з низькою споживаною потужністю та розширеним спектром частот. Вона призначена для забезпечення стабільного зв'язку на великих відстанях при невисоких швидкостях передачі даних, що дозволяє збільшити час роботи радіомодуля від батареї та забезпечує його автономність.

LoRa націлена на ключові вимоги Інтернету речей - безпечність, безвідмовність, двосторонню комунікацію та мобільність [2]. Технології LoRa поділяються на дві субтехнології, які тісно пов'язані: LoRa-технологія фізичного рівня та LoRaWAN, що стосується решти верхніх рівнів.

Технологія розроблена Cycleo з Гренобля (Франція) та придбана корпорацією Semtech у 2012 році, яка є членом-засновником альянсу LoRa [3].

LoRa використовує радіочастотні діапазони, що не потребують ліцензування, зокрема 433 МГц, 868 МГц (Європа) та 915 МГц (Північна Америка). LoRa забезпечує передачу даних на великій відстані (більше 10 км в сільській місцевості) з низьким споживанням енергії [4].

У січні 2018 року було оголошено про нові набори мікросхем LoRa зі зменшеним споживанням електроенергії, збільшенням потужності передачі та зменшеним розміром порівняно зі старшим поколінням. Оскільки технологія LoRa працює лише на нижньому фізичному рівні, то для підтримки мережевого рівня створено протокол LoRaWAN. Модулі LoRa мають геолокаційні можливості, які використовуються для тріангуляції положень пристроїв за допомогою часових позначок від шлюзів [5].

Існує велика кількість різних реалізацій протоколів маршрутизації багатоінтервальних бездротових мереж, зокрема для мереж з низьким енергоспоживанням та динамічною топологією, як у нашому випадку. У системах розумного будинку кінцеві вузли не мобільні, тому їхня топологія не змінюється швидко, однак через електронні завади, які створюють електроприлади, зв'язок може бути ненадійним, а тому вузли можуть відключитися від мережі через низьку потужність.

Маршрутизація для таких мереж, як правило, складаються з п’яти основних етапів - виявлення маршруту, вибір маршруту, обслуговування маршрутів, пересилання даних, представлення маршруту та метрика - у сукупності з кількома допоміжними компонентами. Мобільні однорангові протоколи маршрутизації на основі інформації про топологію мережі, що використовується для виявлення маршруту, поділяються на три категорії: активні (або табличні), реактивні (або працюючі) та гібридні. Існує доволі багато протоколів маршрутизації, які дозволяють реалізувати кожну з цих трьох категорій. Нижче розглянемо реалізацію деяких з них.

1.2.1. Оптимізований Link State протокол маршрутизації. Оптимізований Link State протокол маршрутизації (OLSR) [6] - це активний протокол, який

адаптує маршрутизацію стану зв'язку для використання в мобільних однорангових мережах (MANET) для отримання інформації про топологію мережі [7]. Принцип оновлення топології полягає у механізмі широкомовного розсилання. Особливістю протоколу є те, що кожен вузол володіє інформацією про топологію мережі. Вузли відправляють один-одному так звані HELLO-повідомлення. Таблиця маршрутизації формується з інформації прийнятих таких повідомлень від сусідніх вузлів. Таке повідомлення складається з адреси вузла, що відправив його, а також з переліку всіх доступних йому сусідів, їх унікальні адреси та типи з’єднань, які можуть бути як симетричні так і асиметричні. Таким чином вузол може повідомити своїм сусідам про доступні зв'язки. Кожен вузол зберігає у своїй таблиці маршрутизації інформацію про своїх одно- і двокрокових сусідів. Надсилання HELLO-повідомлень здійснюється через конкретно зазначені проміжки часу. Зв’язок з сусіднім вузлом буде вважатися розірваним, якщо протягом певного інтервалу часу вузол так і не прийняв HELLO-повідомлення від нього.

OLSR добре підходить для великих та щільних мереж із випадковим та хаотичним трафіком. Однак мережі LoRa в розумних будинках, як правило, будуть відносно невеликими. Для покриття такої території достатньо лише одного або двох шлюзів для зв’язку із зовнішнім світом. Таким чином, додаткові витрати на вибір маршруту та оновлення інформації про топологію в нашому випадку не потрібні.

1.2.2. Маршрутизація впорядкованого за призначенням вектору відстані. Маршрутизація впорядкованого за призначенням вектору відстані (DSDV) протокол маршрутизації, який базується на протоколі маршрутної інформації (RIP) для маршрутизації динамічних мереж. Особливість його полягає в тому, що додається новий атрибут, тобто порядковий номер, до кожного запису таблиці маршруту звичайного RIP. Використовуючи щойно доданий порядковий номер, вузли можуть відрізнити застарілу інформацію маршруту від нової і, таким чином, запобігти зацикленню маршрутизації.

У DSDV кожен вузол самоорганізованої мережі формує та оновлює таблицю маршрутизації, в якій перераховані всі доступні шляхи передачі пакета, метрика маршруту та порядковий номер, згенерований вузлом призначення. Використовуючи таку таблицю маршрутизації, що зберігається у кожному вузлі, можна легко визначити оптимальний маршрут для передачі пакета між вузлами.

Хоча DSDV, протокол дає значно менше навантаження на мережеві вузли у порівняні з OLSR, однак постійні оновлення таблиці маршрутизації доволі сильно завантажують повільний канал зв’язку радіомодулів LoRa і є непотрібні для мереж зі статичними вузлами.

1.2.3. Протокол спеціального вектора відстані на вимогу. Реактивні протоколи вимагають обміну інформацією про топологію мережі лише тоді, коли деякі маршрути стають недоступними і потрібно встановити новий маршрут до конкретного вузла. Цей вид протоколів дозволяє зменшити додаткові навантаження на мережу, а отже, і енергоспоживання, порівняно з активними протоколами.

Мабуть, найвідомішим протоколом реактивної маршрутизації для MANETs мереж є спеціальний вектор відстані на вимогу (AODV). Порівняно з іншими схожими протоколами маршрутизації, він є превентивними, тобто знаходить нові шляхи маршрутизації незалежно від використання маршрутів. Як можна побачити з назви, для обчислення маршрутів використовується дистанційно-векторний алгоритм маршрутизації.

У AODV вузол ініціює пошук маршруту, розсилаючи мережею Route Request (RREQ) пакетами. Кожен вузол, який отримує RREQ пакет, зберігає адрес відправника, отримувача та вузла, від якого цей пакет було отримано. Потім ця інформація використовується для створення зворотного шляху, використовуючи Route Reply (RREP) пакет, надісланий адресатом назад адресанту. При виникненні збою у маршрутизації, наприклад, якщо вузол перемістився занадто далеко від свого попереднього положення, для

сповіщення всіх вузлів використовується Route Error пакет (RERR), який може ініціювати пошук нового оптимального маршруту.

1.3. Вибір протоколу маршрутизації

Хоча реактивні протоколи забезпечують менше навантаження на мережу порівняно з активними, але тоді доводиться жертвувати швидкістю доставки пакетів в момент пошуку нового маршруту. Це може бути неприпустимо у випадку, коли протягом певного відрізку часу потрібно швидко передати важливу інформацію. Проблема полягає в тому, що, згідно з протоколом LoRa, кінцеві пристрої (залежно від класу пристроїв) можуть передавати та приймати інформацію у строго визначений час, тому якщо мережа занадто велика, пошук маршруту може зайняти багато часу, що призведе до того, що пакет занадто пізно надійде до кінцевого пристрою.

Одним з рішень цієї проблеми є гібридні протоколи маршрутизації, які можуть балансувати між навантаженістю на мережу та затримками між пошуком нових маршрутів. Протокол маршрутизації зон (Zone Routing Protocol) розбиває мережу на окремі ділянки (зони) і використовує активний протокол маршрутизації в середині зон та реактивний для маршрутизації між зонами. Такий підхід в першу чергу корисний у великих мережах, однак в межах одного будинку розділяти мережу на зони необов’язково.

З іншого боку, для систем розумного будинку, які базуються на мережі LoRa, можна використати гібридний протокол для безпроводових Mesh-мереж (HWMP), який базується на стандарті IEEE 802.11s. HWMP заснований на поєднанні протоколу AODV та на деревовидній маршрутизації і може працювати як і в активному так і в реактивному режимі.

Активний режим вимагає, щоб кореневий вузол виступав у ролі шлюзу і мав підключення до Інтернету. У цьому режимі кореневий вузол періодично надсилає інформацію про маршрутизацію та метрику вниз по мережевому дереву, що дозволяючи кожному вузлу дізнатися маршрут до кореневого вузла.

Оскільки цей обмін інформацією обмежений лише дотриманням структури дерева, він вимагає значно менших обчислюваних ресурсів, ніж інші активні протоколи, які одночасно розсилають таблиці маршрутизації по мережі.

Незважаючи на всі переваги даного підходу, окрім затрати на обчислювальні ресурси для підтримки протоколу та затримки на виявлення маршруту, також слід враховувати продуктивність при надсиланні трафіку мережею. Для порівняння протоколів можна використати мережевий симулятор з відкритим кодом NS-3, який розповсюджується під ліцензією GNU gplv2. При порівнянні протоколів на середню затримку у симуляторі було виявлено, що DSR показав кращі результати, ніж AODV і ZRP, через кешування маршрутів, що використовується в DSR. Незважаючи на це, AODV досяг найвищої пропускної здатності незалежно від кількості вузлів у мережі. Однак, HWMP показав трохи кращі результати, ніж AODV з точки зору доставки пакетів, пропускної здатності та затримки. Але мережі на протоколі AODV набагато легше розгортати та конфігурувати в порівнянні з HWMP, оскільки він вимагає додаткових кореневих вузлів, і деревовидна топологія - не найкраще рішення для даного типу мережі. Тому протокол AODV є оптимальним рішенням для побудови WMN в системі розумного будинку.

1.4. Висновки до розділу

У даному розділі проаналізовано переваги і недоліки технологій та протоколів бездротової передачі даних, які використовуються в сучасних системах контролю розумним будинком. У результаті аналізу літературних джерел встановлено, що розробка такої системи є актуальною прикладною задачею, саме тому для розробки системи було обрано технології LoRa та Mesh.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ АПАРАТНОЇ ЧАСТИНИ LoRa MESH-МЕРЕЖІ

2.1. Вибір платформ для побудови LoRa Mesh-мережі

З метою забезпечення належного рівня надійності систему контролю розумним будинком з використанням LoRa Mesh-технологій, було вирішено розробити на двох різних платформах. Основні модулі, які будуть реалізувати підтримку маршрутизації Mesh-топології, працюватимуть на платформі Arduino Nano. Шлюз, який надаватиме доступ WMN до мережі Інтернет, та дозволить керувати кінцевими пристроями за допомогою смартфону, доцільно реалізувати на платформі ESP8266. Зважаючи на часткову готовність, у роботі використано відлагоджувальний модуль на цій платформі, який був створений раніше для інших проектів. Вказані платформи було обрано через гнучкість, простоту в програмуванні та відлагоджені готового проекту, а також наявність великої кількості бібліотек та розширень.

В якості радіомодуля було обрано поширений LoRa трансивер RFM95, який є простий у користуванні, має відкриту документацію та підтримується бібліотекою RadioHead розробленою компанією Adafruit Industries. Для виводу інформацію про стан роботи основного шлюзу обрано LCD-дисплей, що може відображати шістнадцять символів у два рядки та модуль 2PH84388A для його підключення.

2.2. Опис платформи Arduino Nano

Arduino Nano - це пристрій, основою якого є мікроконтролер ATmega328p. Його розробила і випускає компанія Gravitech. Ширина і довжина друкованої плати становить 1,85 см і 4,3 см відповідно. Платформа містить: 8 аналогових входів, 14 цифрових входів/виходів, шину SPI, кварцовий резонатор на 16 МГц Кнопка, яка міститься на платі, під'єднана до інверсного виводу

RESET мікроконтролера ATmega328p, і підтягує його до землі, тобто при її натиснені він перезавантажиться. Arduino Nano можна живити за допомогою USB кабелю стандарту Mini-B, використовуючи зовнішнє джерело живлення з діапазоном напруги 7,2-15 В (вивід VIN) або напругою 2,7-5 В (вивід +5V). Напруга на USB-UART перетворювач на базі мікросхеми FTDI FT232RL подається у випадку, якщо платформа живиться від USB. При живленні Arduino Nano від зовнішнього джерела на виході 3V3, напруга на якому формується за допомогою мікросхем FTDI, потенціалу не буде, тому світлодіоди RX і TX можуть світитися лише при наявності високого потенціалу на вбудованому послідовному порті. На платі Arduino Nano є вбудований лінійний стабілізатор на 5 вольт, тому її можна живити напругою до 12 вольт. В залежності від виробника стабілізатора напруги живлення можна піднімати аж до 35 вольт, проте на більшості дешевих плат встановлено стабілізатори сумнівної якості, тому не рекомендовано перевищувати поріг в 15 вольт, оскільки модуль може вийти з ладу.

Виводи живлення, розташовані на платі:

— VIN – для живлення плати від нестабілізованого джерела напруги;

— +5V – стабілізованої напруги 2,7-5В;

— GND – виводи землі.

Плата спеціально поставляється без впаяних роз'ємів, що дозволяє користувачеві приєднувати провідники або використовувати необхідні типи роз'ємів на власний розсуд.

Мікроконтролер ATmega328p має такі характеристики:

— напруга живлення 2,7-5,5 вольт;

— підтримує розширений набір RISC-інструкцій;

— має 32x8 робочих регістрів;

— при тактуванні зовнішньою тактовою частотою у 16 МГц, обчислювальна потужність може сягати понад 16 MIPS;

— внутрішня флеш-пам'ять, має об’єм 32 Кбайти;

— 1 Кбайт EEPROM;

— 2 Кбайта внутрішньої SRAM;

— цикли запису/стирання: флеш-пам'ять 10000, EEPROM 100000;

— вбудовані SPI, I^2C та UART інтерфейси;

— програмований сторожовий таймер з вбудованим генератором;

— шість каналів ШІМ та шість режимів очікування;

— лічильник реального часу з окремим генератором;

— один 16-бітний та два 8-бітні таймери/лічильники з окремими дільниками та пристроями порівняння.

ATmega328p - це 8-бітний CMOS мікроконтролер з низьким рівнем енергоспоживання на основі розширеної архітектури RISC. Виконуючи інструкції за один тактовий цикл, ATmega328p досягає пропускної здатності понад 1MIPS на МГц, що дозволяє оптимізувати енергоспоживання [8] в залежності від швидкості виконання програми.

Ядро мікроконтролера поєднує розширений набір інструкцій з 32 робочими регістрами загального призначення [8]. Всі регістри безпосередньо підключені до арифметично логічного пристрою, що дозволяє одній інструкції отримати доступ до двох незалежних регістрів, за один тактовий цикл [9]. З 32 Кбайт вбудованої флеш-пам'яті, 2 Кбайта виділено під завантажувач, який дозволяє переписувати внутрішню пам’ять програм, не використовуючи спеціалізований SPI програматор. Як правило внутрішні FUSE-біти контролера налаштовані так щоб, спочатку відбувався запуск завантажувача і користувач, мав доступ до всіх видів пам’яті, внутрішніх таймерів та регістрів. Слід зазначити, що неправильна зміна одного з цих бітів, може призвести до виходу з ладу мікроконтролера. Відновити його роботу можна буде лише за допомогою паралельного програматора. Також контролер має, 2 Кбайта EEPROM, 1 Кбайт SRAM, 23 лінії вводу/виводу, 32 робочих регістри загального призначення, три гнучких таймера/лічильники з режимами порівняння, внутрішніми та зовнішніми перериваннями, універсальний синхронно-асинхронний приймач, біт-орієнтований послідовний інтерфейс, 6-канальний 10-бітний АЦП,

програмований сторожовий таймер [8] із внутрішнім генератором та п’ятьма режимами програмного енергозбереження.

Вбудований мікросхему драйвер ISP дозволяє перепрограмувати пам'ять програми в через послідовний інтерфейс SPI. Завантажувач може використовувати будь-який інтерфейс для завантаження програми у флеш-пам’яті [8]. ATmega328p підтримується повним набором програмних та системних засобів, включаючи: компілятори C, макроскладачі, програмні відлагоджувачі та вбудовані емулятори [9].

Основна функція ядра ATmega328p - забезпечення правильного виконання програми. Тому мікроконтролер повинен мати доступ до пам'яті, виконувати обчислення, контролювати периферійні пристрої та обробляти переривання (рис. 2.1 [9]).

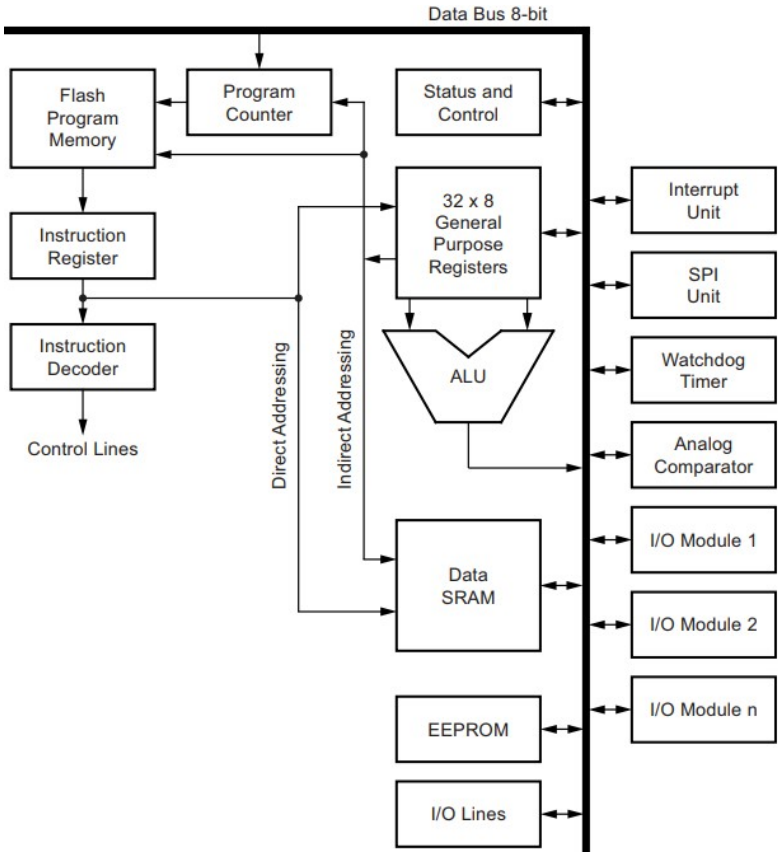

Рис. 2.1. Блок-схема мікроконтролера

Для роботи з чотирнадцятьма цифровими вводами/виводами, можна використовувати стандартні функції digitalWrite(), digitalRead(), pinMode() або

працювати напряму з стандартними регістрами PORTD, PORTB, DDRD та DDRB. Напруга цифрового нуля або одиниці визначається напругою живлення. Струм навантаження, підключеного до цифрового виводу, не повинен перевищувати 40 мА. Також на цифрових входах можна увімкнути внутрішні підтягуючі резистори номіналом 20-47 кОм. Цифрові виводи можуть виконувати і додаткові функції. Нульовий і перший виводи (на платі вони підписані RX і TX) можуть працювати в режимі універсального асинхронного приймача/передавача. Зазвичай вони з'єднані з відповідними виводами мікросхеми FT232RL і через них відбувається програмування мікроконтролера. Також їх зручно використовувати для моніторингу виконання програми, під час відлагодження коду.

На виводи 3, 5, 6, 9, 10, 11, використовуючи функцію analogWrite(), можна подавати 8-бітовий аналоговий ШІМ сигнал.

Вивід 8 пов'язаний з внутрішнім таймером мікроконтролера T1. Якщо налаштувати дільник частоти, та перевести таймер у режим захоплення, то використовуючи регістри ICR1 і ICP1, можна заміряти характеристики (коефіцієнт заповнення, частоту, період) будь-якого прямокутного сигналу.

На виводах 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK), реалізовано апаратну підтримку інтерфейсу SPI. Використовуючи стандартні функції, які входять в середовище розробки Arduino IDE, можна задати наступні параметри роботи інтерфейсу: частота, послідовність даних та тип тактового імпульсу.

До тринадцятого цифрового виводу під'єднано вбудований світлодіод. При наявності на виводі високого рівня сигналу він горить. На платі Arduino Nano також є 8 аналогових входи. Кожен з них з'єднано з 10-бітним аналого-цифровим перетворювачем, тому сигнал з них може бути представлений у вигляді 1024 різних значень. На платі вони пронумеровані значеннями A0-A7. Напругу можна вимірювати в діапазоні від нуля до напруги живлення мікроконтролера. Як і цифрові виводи, деякі з них теж мають додаткові функції.

Виводи A4 і A5 можуть працювати, як послідовна асиметрична шина I^2C. Для цього можна використовувати стандартну бібліотеку Wire.

На платі є ще вивід RESET. Зазвичай він підтягнутий до плюсової шини живлення через резистор 10 кОм. Він використовується для функціонування додаткових кнопок перезавантаження на платах розширення. Оскільки як він з’єднаний з відповідним інверсним виводом мікроконтролера, то при підтягнені його до землі, плата перезавантажиться.

На рис. 2.2 [10] показано повний перелік всіх виводів та їх призначення для платформи Arduino Nano на базі мікроконтролера ATmega328p.

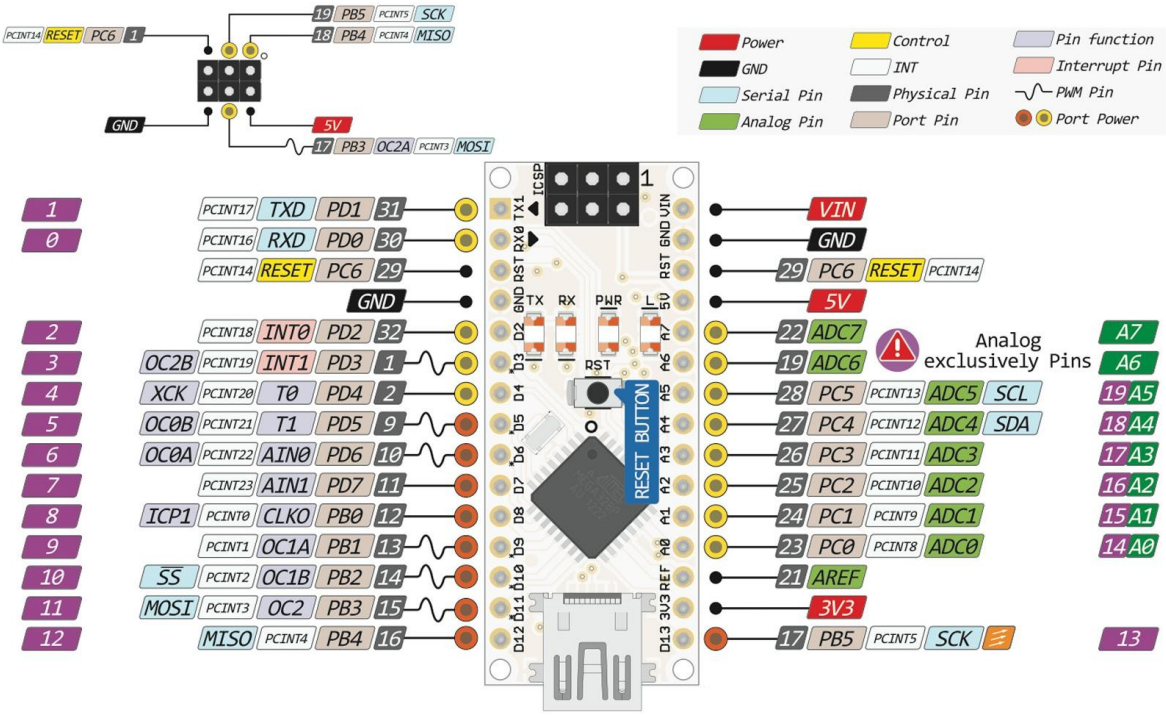

Рис. 2.2. Призначення виводів платформи Arduino Nano

На платі передбачена можливість для обміну інформацією ще з однією платою Arduino, комп'ютером або іншими мікроконтролерами. Тобто, ATmega328p за допомогою цифрових виводів RX і TX, може здійснювати зв’язок завдяки вбудованому UART-інтерфейсу. В додаток до цього, використовуючи стандартну бібліотеку SoftwareSerial, можна будь-які цифрові виводи платформи, перетворити у віртуальні послідовні інтерфейси. В середовище Arduino IDE вбудовано монітор послідовного порту, який дозволяє відправляти або приймати текст у вигляді ASCII символів, через USB-інтерфейс.

У мікроконтролер можна завантажувати нові програми без використання зовнішнього програматора, оскільки плата випускається вже із вшитим завантажувачем. Програмування здійснюється за протоколом STK500. Якщо мікроконтролер без завантажувача або завантажувач пошкоджено, можна скористатися внутрішньо-схемним програмуванням ICSP. Зручно використовувати програматор USBAsp, підключивши його до відповідного роз'єму. За допомогою утиліти AVRDude спочатку потрібно переконатися у правильній конфігурації FUSE-бітів і за необхідності виставити їх ключем avrdude -c usbasp -p m328p -P usb -v -U lock:w:0xFD:m -U lfuse:w:0xFF:m -U hfuse:w:0xDA:m. Після цього оновити завантажувач стандартними інструментами середовища Arduino.

Платформою передбачено перезавантаження мікроконтролера за допомогою комп'ютера, щоб кожен раз при програмуванні не потрібно було натискати кнопку на платі. Для цього вивід RESET, через керамічний конденсатор з номінальною ємністю 100-200 нФ, під'єднаний до виводу управління потоку даних (DTR) мікросхеми FT232RL. Таку ємність конденсатора підібрано відповідним чином: якщо на лінії DTR з'являється логічний нуль, то вивід RESET також переходить в низький рівень на час, достатній для перезавантаження мікроконтролера. Оскільки процес прошивання мікроконтролера напряму зв'язний зі спадом сигналу на лінії DTR, то це дозволяє завантажити код, лише натисканням однієї кнопки в середовищі розробки.

Разом з тим дана система має один недолік - при під'єднані платформи Arduino Nano до комп'ютера під управлінням операційної системи на базі ядра Linux або Mac OS X, мікроконтролер буде перезавантажуватися, при кожному з'єднані програмного забезпечення з віртуальним послідовним портом. Якщо при першому запуску платі потрібно отримати від комп'ютера налаштування або будь-яку іншу інформацію, то завантажувач під час свого запуску може перехопити ці дані і програма може працювати некоректно. Для цього слід передбачити перевірку на готовність Arduino Nano, отримувати дані через

послідовний порт, або відправляти їх через секунду після встановлення з'єднання.

2.3. Трансивер RFM95, як основа LoRa Mesh-мережі

Радіопередавач RFM95 оснащений модемом LoRa на базі мікросхеми RF96 (його архітектурна діаграма зображена на рис. 2.3 [11]). Він забезпечує зв’язок у широкому спектрі частот та високий захист від завад, при мінімальній споживаній потужності.

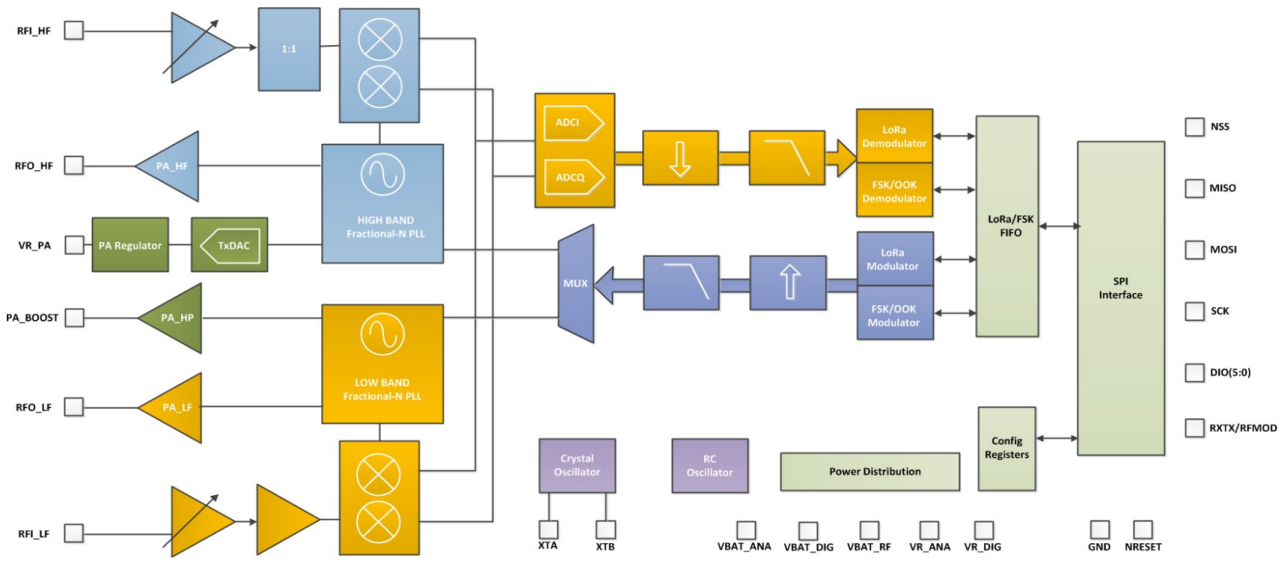

Рис. 2.3. Архітектура RF96

Використовуючи запатентовану технологію модуляції LoRa Hope RF [12], чутливість трансивера може досягати понад 148 дБм. Дана технологія не потребує використання специфічних матеріалів, для виготовлення кристалу мікросхеми RF96, що дозволяє суттєво знизити ціну модуля. LoRa має значну перевагу як у блокуванні, так і у селективності порівняно із звичайними методами модуляції, що дозволяє досягти компромісу між завадостійкістю та низьким споживанням електроенергії [12]. Висока чутливість у поєднанні з вбудованим підсилювачем потужності +20 дБм [12], забезпечує високу

надійність зв’язку, що робить модуль оптимальним для використання в системах розумного будинку.

Трансивер підтримує високоефективні режим частотної маніпуляції (FSK) включаючи WMBus та IEEE802.15.4g. Конструкція RF96 забезпечує низький фазовий шум, що дозволяє збільшити селективність каналів зв’язку, лінійність приймача та IIP_3.

Радіомодуль RFM95 має такі характеристики:

— максимальний енергетичний баланс лінії 168 дБ;

— швидкість передачі даних до 300 кбіт/с;

— висока чутливість: до -148 дБм;

— споживання струму: в режимі прийому 10,8 мА, в режимі передачі 120 мА, в режимі очікування 200 нА;

— інтегрований синтезатор частоти з роздільною здатністю 61 Гц [11];

— підтримує GFSK, FSK, MSK, LoRa та OOK [11] модуляції;

— вбудований бітовий синхронізатор для забезпечення тактової синхронізації через послідовний інтерфейс;

— динамічний діапазон RSSI 127 дБ;

— вбудований датчик температури та індикатор низького рівня заряду акумулятора [11].

RFM95 включає в себе модем розширеного спектру LoRa, який здатний досягти значно більшого діапазону, ніж існуючі системи, засновані на модуляції FSK або OOK. За допомогою цієї модуляції можна досягти на 8 дБ вищої чутливості ніж при FSK. Це дозволяє розширити радіус та надійність зв’язку без використання додаткових підсилювачів потужності.

Для максимальної гнучкості, модуль дозволяє налаштувати пропускну здатність, спектр поширення, коефіцієнт поширення та швидкістю виправлення помилок. Ще одна перевага розширеної модуляції полягає в тому, що кожен коефіцієнт розширення є ортогональним, тобто одночасно можна приймати дані з різних передавачів одним каналом зв’язку. Це також дозволяє усунути конфлікт вже з існуючими безпровідними мережами на базі FSK. Блок-схему

модему LoRa показано на рис. 2.4 [12]. Як бачимо, він має незалежний буфер даних FIFO з подвійним портом, до якого можна отримати доступ через інтерфейс SPI. Дана конфігурація дозволяє швидко переключатися між FSK та LoRa модемами, використовуючи RegOpMode регістр. Переключення можна здійснювати в режимі сну, це дозволяє використовувати FSK і OOK модуляції в поєднані з LoRa, завдяки чому радіус дії трансивера можна значно розширити. Спрощений представлення контур передачі та прийому також показано на рисунку. Процес модуляції та демодуляції є комерційною таємницею, відомо що він використовує форму модуляції з розширеним спектром, в поєднану з кодуванням циклічного виправлення помилок.

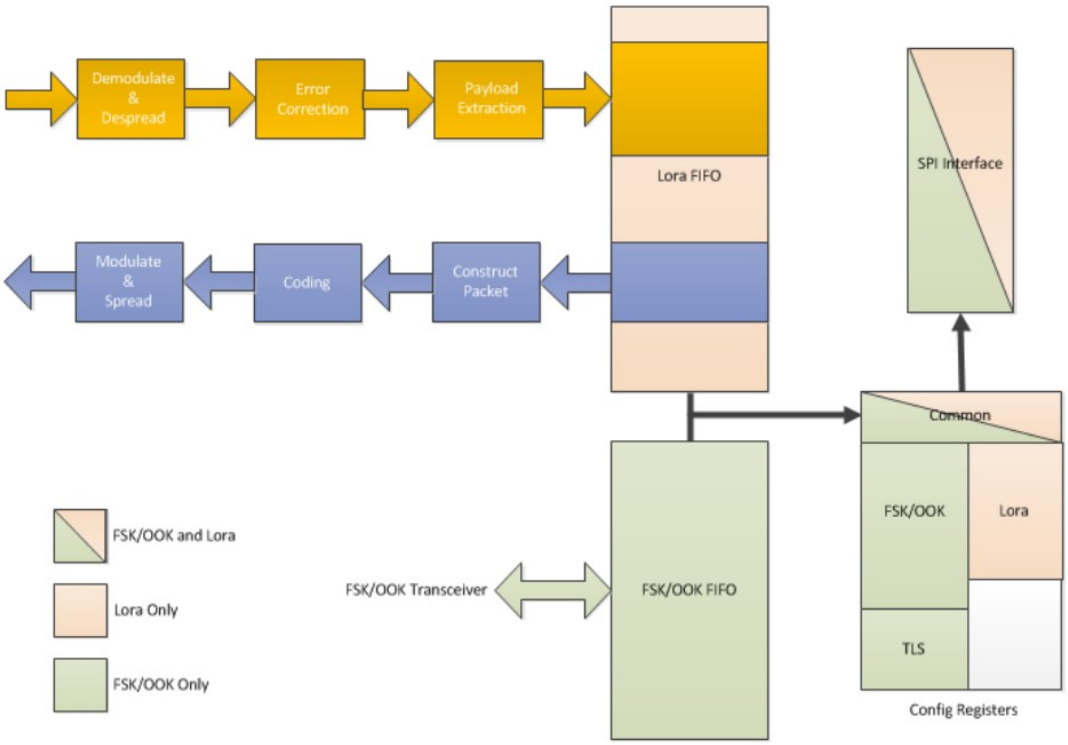

Рис. 2.4. Блок-схему модему LoRa

Модуляція LoRa з розширеним спектром, передбачає передачу кожного біта корисного навантаження, паралельно декількома каналами. Швидкість, з якою надсилається інформація корисного навантаження, називається швидкістю модуляції (R_M), співвідношення між номінальною швидкістю модуляції та реальною швидкістю трансивера визначається коефіцієнтом спектру поширення

і являє собою кількістю символів, надісланих на біт інформації. Діапазон значень, які доступні модему LoRa, наведено у табл. 2.1 [12].

Таблиця 2.1

Доступні коефіцієнти спектру поширення

Значення, яке потрібно занести в регістр конфігурації	Коефіцієнт спектру поширення	Співвідношення сигнал/ шум, при демодуляції
6	64	-5 дБ
7	128	-7,5 дБ
8	256	-10 дБ
9	512	-12,5 дБ
10	1024	-15 дБ
11	2048	-17,5 дБ
12	4096	-20 дБ

Для правильного функціонування мережі коефіцієнт поширення повинен бути однаковим на всіх трансиверах, оскільки різні коефіцієнти є ортогональними один до одного. Щоб збільшити чутливість приймача, необхідно обрати коефіцієнт з найнижчим співвідношення сигнал/шум. Для подальшого покращення надійності зв'язку LoRa-модем використовує попередню корекцію помилок для виявлення та виправлення помилок в отриманих пакетах даних. Цей метод передбачає додаткову службову інформацію в структурі пакета, тому може призвести до значного зниження швидкості мережі. Найоптимальнішим значенням коефіцієнта попередньої корекції помилок є 4/5, тобто надлишковість даних сягатиме лише 25%.

Визначивши всі ключові параметри модема, можна легко знайти частоту модуляції каналу за формулою [11]:

$$\nu_s = \frac{B_W}{2^{S_F}}, \tag{2.1}$$

де B_W - запрограмована пропускна здатність,

S_F - коефіцієнт поширення.

Сигнал, що передається, є обвідною коливального сигналу, тобто кривою, яка огинає всі його екстремуми. Розмірність такої величини - кількість біт, надісланих одним каналом за секунду, на герц пропускної здатності.

Модем LoRa використовує два типи формату пакетів - явний та неявний. Явний пакет включає короткий заголовок, який містить інформацію про кількість байтів, швидкість кодування та про те, чи використовується CRC в пакеті. Формат пакету наведено на рис. 2.5 [12].

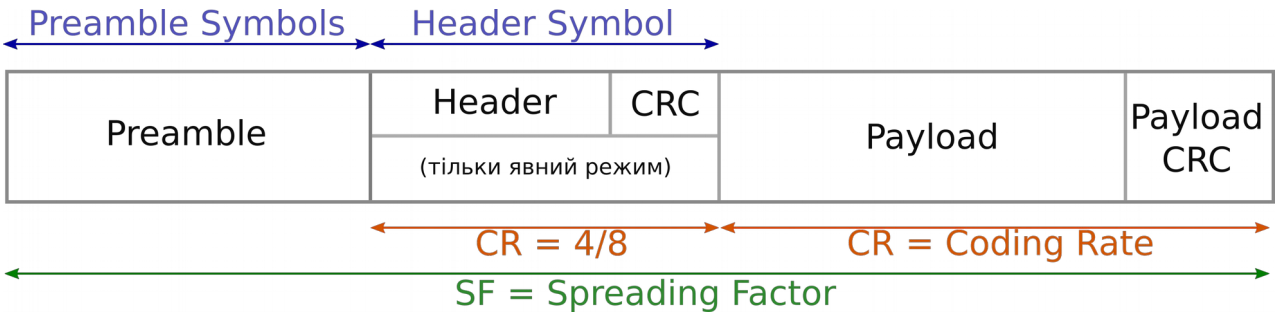

Рис. 2.5. Структура пакету LoRa

Преамбула використовується для синхронізації приймача з потоком вхідних даних (за замовчуванням її розмір становить 12 символів). Розмір преамбули можна змінити, записавши відповідне значення в регістр PreambleLength. Наприклад, її розмір можна збільшити для скорочення робочого циклу приймача, при інтенсивній передачі даних. Розмір можна змінювати в діапазоні від 6 до 65535, задаючи загальну довжину преамбули від 6 + 4 до 65535 + 4 символів. Довжина преамбули на приймачі повинна бути налаштована ідентичною довжині преамбули передавача. Якщо довжина преамбули не відома або може змінюватися, потрібно налаштувати максимальну довжину преамбули на боці приймача.

Залежно від режиму роботи доступні два типи заголовків. Тип заголовка можна встановити бітом ImplictHeaderMode, який знаходиться в регістрі RegSymbTimeoutMsb.

У явному режимі, який увімкнено за замовчуванням, заголовок містить інформацію про корисне навантаження, а саме:

— довжину корисного навантаження в байтах;

— швидкість попередньої корекції помилок;

— наявність додаткового 16-бітного CRC для корисного навантаження.

Заголовок має власну CRC, яка дозволяє приймачу відкидати недійсні заголовки. Він передається з максимальним кодом виправлення помилок (4/8).

Неявний режим використовується у випадку, якщо корисне навантаження, швидкість кодування та наявність CRC відомі заздалегідь. В цьому режимі заголовок відсутній взагалі, що дозволяє вигідно скоротити час передачі пакета. У цьому випадку довжина корисного навантаження, швидкість кодування помилок та наявність CRC, повинні бути налаштовані заздалегідь з обох боків радіозв'язку. Слід зазначити таку особливість - якщо вибрати режим спектру поширення 6 (див. таблиця 2.1), то неявний режим буде єдиним можливим режимом роботи.

Корисне навантаження - це поле змінної довжини, яке містить фактичні дані, кодовані з коефіцієнтом помилок, який вказано в заголовку в явному режимі, або налаштованим в регістрі в неявному режимі. До нього може бути додано необов'язкове поле CRC.

Для заданої комбінації коефіцієнта поширення, швидкості кодування та пропускної здатності сигналу, можна обчислити загальний час передачі пакету LoRa. З визначеної частоти модуляції каналу (див. формула 2.1), зручно визначити час модуляції каналу:

$$T_s = \frac{1}{\nu_s}, \tag{2.2}$$

Час за який можна передати пакет LoRa, рівний сумі часу, який необхідно для передачі корисного навантаження та преамбули. Тоді, час за який можна передати преамбулу, можна знайти за формулою [11]:

$$T_{preamble} = T_{sym} \cdot (n_{preamble} + 4{,}25), \tag{2.3}$$

де T_{sym} - час, необхідний на передачу одного символу,

$n_{preamble}$ - запрограмована довжина преамбули за допомогою регістра PreambleLength.

З наступної формули можна знайти час передачі корисного навантаження в неявному (без заголовка) та явному (із заголовком) режимах [11]:

$$T_{payload} = \begin{cases} T_{sym}\left[8 + ceil\left(\dfrac{8L_P - 4S_F + 24}{4S_F}\right)\right] \cdot (C_R + 4), при\ L_P > 0, без\ заголовка, \\ T_{sym}\left[8 + ceil\left(\dfrac{8L_P - 4S_F + 44}{4S_F}\right)\right] \cdot (C_R + 4), при\ L_P > 0, з\ заголовком, \end{cases} \tag{2.4}$$

де C_R - швидкість кодування,

L_P - кількість символів корисного навантаження,

ceil() - математична функція, яка повертає найменше ціле число, що є більшим або дорівнює даному числу.

Тоді час, який необхідно для передачі пакету LoRa, можна знайти за формулою [11]:

$$T_{packet} = T_{preamble} + T_{payload} \tag{2.5}$$

Якщо, час необхідний для передачі одного пакета перевищує нормативні вимоги, що стосуються максимально допустимого часу використання каналу зв'язку, можна скористатися функцією псевдовипадкового переналаштування робочої частоти (FHSS). Ця функція корисна в нашому випадку так як обраний трансивер працює в діапазоні частот 902-928 МГц. Для увімкнення режиму FHSS, необхідно встановити конфігураційні біти в регістрі RegTxCfg1.

Суть методу псевдовипадкового переналаштування робочої частоти полягає у періодичній стрибкоподібній зміні частоти модему за певним алгоритмом, відомим для приймача та передавача.

Модем LoRa містить три типи цифрового інтерфейсу, статичні регістри конфігурації, регістри стану та буфер даних FIFO. Доступ до них можна отримати через послідовний периферійний інтерфейс. Трансивер спілкується з хост-мікроконтролером через 4-дротовий інтерфейс SPI, як залежний пристрій. SPI-сумісний послідовний інтерфейс дозволяє обирати, керувати і слідкувати за станом RFM95 через хост-мікроконтролер. Всі регістри складаються з командного коду, за яким слідує змінне число параметрів чи бітів даних. Оскільки пристрій використовує запис словом, то пін CS повинен бути підтягнутим до нижнього рівня для 16 бітів. Біти даних на піні MOSI зсуваються в пристрій до підйому в такті на піні SCK, допоки пін CS має низький логічний рівень.

Максимальна тактова частота на SPI-шині – 20 МГц. Трансивер підтримує SPI-режим 0,0 що вимагає від SCK залишаться в режимі очікування на низькому логічному рівні. Пін CS повинен утримуватись в низькому стані задля комунікації між мікроконтролером і RFM95. Дані отримуються трансивером через MOSI і тактуються по зростаючому фронту SCK. RFM95 посилає дані через MISO і тактується по спадаючому фронту SCK. Найбільший значущий біт відправляється першим (наприклад 15-ий біт для 16-бітної команди) в будь-яких даних. POR-схема встановлює значення по замовчуванню у всіх командних регістрах і регістрах керування. Пін MISO встановлюється в нижній логічний рівень по замовчуванню, коли пін CS у високому рівні. Цей пін має три-становий буфер і використовує логіку утримання шини.

Регістри читаються в усіх режимах RFM95, включаючи режим сну. Однак їх слід писати лише в режимі сну та очікуванні. Слід зауважити, що автоматичний секвенсор верхнього рівня (режим TLS) недоступний у режимі LoRa. Вміст регістрів конфігурації LoRa зберігається в режимі FSK/OOK. Для функціональності регістри режимів, спільні як для режиму FSK/OOK, так і для

LoRa. Під час роботи приймача, інформацію про стан трансивера можна отримати через регістри статусу.

Трансивер RF96 оснащений 256-байтним буфером оперативної пам'яті, який доступний в режимі LoRa. Ця область оперативної пам’яті, використовується як буфер даних FIFO. Повний доступ до буфера даних LoRa FIFO можна отримати через інтерфейс SPI. Цей буфер даних FIFO можна прочитати у всіх режимах роботи, за винятком режиму сну та зберігання даних, тому що даний режим пов'язаний з операцією прийому, оскільки буфер автоматично очищається від старого вмісту при кожному новому переході до режиму прийому.

Завдяки подвійній конфігурації портів можна одночасно зберігати як передану, так і отриману інформації у FIFO-буфер даних. Регістр FifoTxBaseAddr вказує на адресу в пам'яті, де зберігаються передані дані. Аналогічно регістр FifoRxBaseAddr вказує на адресу, куди будуть записані дані під час операції прийому. За замовчуванням трансивер налаштовано так, що при увімкнені живлення половина наявної пам'яті виділяється під передачу (FifoRxBaseAddr ініціалізовано за адресою 0x00), а інша половина виділена під прийом (FifoTxBaseAddr ініціалізовано за адресою 0x80). Зрозуміло, що базові адреси для прийому/передачі, повинні налаштовуватися в області 256-байтної оперативної пам'яті.

Для використання максимального розміру буфера даних FIFO в режимі передачі або прийому, весь буфер можна використовувати в кожному режимі, встановивши базові адреси регістрів FifoTxBaseAddr і FifoRxBaseAddr в нижній області пам'яті (0x00). Буфер FIFO очищається, коли трансивер перебуває в режимі сну, отже, в цьому режимі неможливий доступ до нього. Буфер не має функції самоочищення (якщо тільки трансивер не перебуває у сплячому режимі), тому нові дані будуть записуватися в область зайнятої пам'яті.

Регістр FifoRxBytesNb визначає розмір комірки пам'яті для запису в разі успішної операції прийому. З іншого боку, в явному режимі у заголовку вказано розмір пам'яті, який потрібно зарезервувати для корисного навантаження. У

неявному режимі, регістр FifoRxBytesNb не використовується, оскільки кількість байтів корисного навантаження відома заздалегідь. Змінна FifoRxCurrentAddr вказує розташування останнього пакета, отриманого у FIFO, тому останній отриманий пакет можна легко прочитати, записавши в регістр FifoAddrPtr вищезгадану змінну. Слід зауважити, що всі отримані дані будуть записані в буфер даних FIFO, навіть якщо CRC недійсний. Це дозволяє, наприклад, обробляти отримані дані, для подальшого усунення помилок і налагодження зв’язку. Також важливо відзначити, що при отриманні, пакета розмір якого перевищує буферну пам'ять, виділену для прийому, його частину буде записано в передавальну частину буфера даних.

На рис. 2.6 [12] наведено блок-схему алгоритму передачі даних LoRa

Рис. 2.6. Алгоритм передачі пакета LoRa

модемом. Для зменшення енергоспоживання, можна активізувати блоки RF, PLL та PA, якщо необхідно передавати лише пакети даних.

Статичні регістри конфігурації, для запису даних у FIFO буфер, можуть бути доступні лише у режимі сну або після ініціалізації режиму передачі. Саме тому для відправлення пакету LoRa, спочатку необхідно вивести трансивер з режиму очікування (у режим сну його переводити не слід, бо при надходженні вхідних даних модуль прийому не зможе обробити необхідні переривання в цей момент), ініціалізувати режим передачі та записати дані в буфер. Тоді модуль автоматично розпочне передачу, і по її завершенні надішле сигнал переривання TxDone. Після цього в буфер можна записати нову порцію даних, тоді модуль знову розпочне їх передачу, або перевести його вручну в режим очікування. Схожим способом відбуваєте прийом пакета (рис. 2.7 [12]).

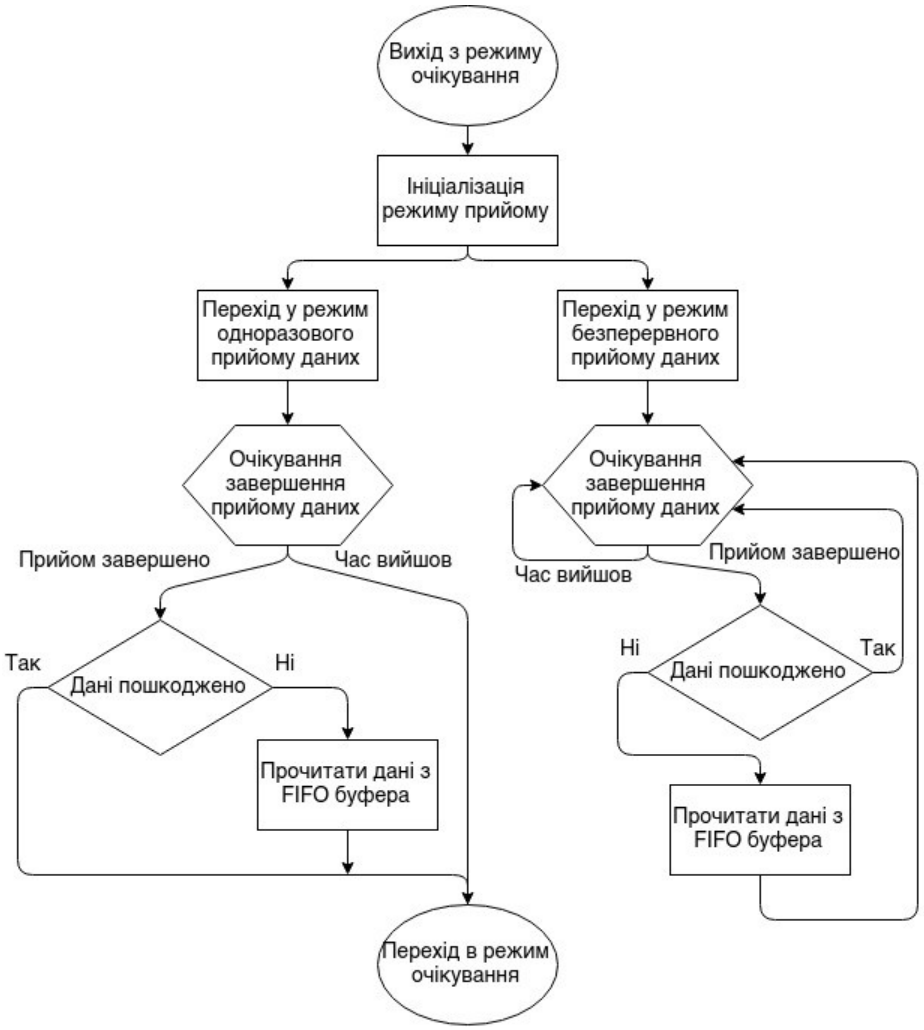

Рис. 2.7. Алгоритм прийому пакета LoRa

Блок-схема демонструє, що приймач може працювати в одноразовому та безперервному режимах прийому даних. Ці два режими використовуються у різним випадках, тому важливо зрозуміти відмінності між ними.

У режимі одноразового прийому даних, модем шукає преамбулу протягом заданого відрізку часу. Довжина часового вікна (у символах) попередньо задається регістром RegSymbTimeout, який повинен знаходитися в діапазоні від 4 до 1023 символів. Якщо за цей час преамбулу не знайдена, то трансивер генерує переривання RxTimeout і повертається в режим очікування.

Після отримання пакета, разом із переривання RxDone, якщо CRC корисного навантаження було недійсний, генерується переривання PayloadCrcError. У будь-якому випадку пакет однаково буде записано у FIFO буфер для подальшого опрацювання. Отже, режим одноразового прийому даних слід використовувати, коли заздалегідь відомо часове вікно прийому даних, оскільки LoRa-модем автоматично повертається в режим очікування, після генерування переривань RxDone або RxTimeout.

У режимі безперервного прийому даних, модем постійно сканує канал на преамбулу. Кожен раз, коли виявляється преамбула, модем зберігає корисне навантаження у буфер і продовжує сканувати канал зв’язку. Слід зазначити, що байти записуються в пам'ять буфера даних в отриманому порядку. Значить, перший байт нового пакета пишеться відразу після останнього байта попереднього пакету. Тому, доки режим безперервного прийому даних увімкнено, вказівник FifoRxCurrentAddr ніколи не скидається Для цього платформі Arduino Nano необхідно обробляти адресний вказівник, щоб запобігти переповнення буфер даних FIFO.

Модем LoRa автоматично фільтрує отримані пакети на основі будь-якої адреси. Однак RFM95 дозволяє налаштувати фільтрування отриманих пакетів на основі вмісту перших кількох байтів корисного навантаження. Процес фільтрації пакетів дозволяє відкидати непотрібні пакети, не приймаючи повністю їхнє корисне навантаження і не записуючи його у буфер. Після

операції відкидання модем переходить в режим очікування, для підвищення енергоефективності.

2.4 Опис Wi-Fi модуля ESP8266

Платформу власної конфігурації з використанням модуля ESP8266 було обрано на основі NodeMcu. Її перевагою є: автономне живлення від акумулятора стандарту 18650, вбудований stand-up і stand-down dc-dc перетворювач, який дозволяє ефективно використовувати ємність акумулятора, та автономний модуль зарядки від Micro-USB. Ширина і довжина друкованої плати становлять 10 см і 10 см відповідно.

Платформа спеціально розроблена для створення різних пристроїв інтернету речей (IoT). Вбудований ESP8266 модуль вміє відправляти (і отримувати) інформацію в локальну мережу або в інтернет за допомогою Wi-Fi. Крім цього, на платі розмічено 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 I^2C. На платі, також розміщено дві функціональні кнопки FLASH і RESET, які використовуються для її перезавантаження та переведення у режим програмування.

Існує декілька поколінь плат, які відрізняються схемою dc-dc перетворювача та комплектацією мікросхем-розширювачів. Перша версія плати містить SPI to I^2C міст CP2120 та розширювач цифрових портів PCF8574. Для вибору адресу I^2C шини, на зворотньому боці плати розміщено три функціональні перемички. Стабілізація напруги виконана на базі stand-up і stand-down dc-dc перетворювача 2149F. В новому поколінні, він побудований на базі мікросхеми TPS63020, оскільки вона потребує менше обв'язки, та має набагато вищий коефіцієнт корисної дії при такій самій ціні. Також міст CP2120 було замінено на SPI конвертер для підключення адресних світлодіодів.

ESP8266 - мікроконтролер китайського виробника Espressif Systems з інтерфейсом Wi-Fi [13]. Окрім Wi-Fi, мікроконтролер відрізняється можливістю виконувати програми з зовнішньої флеш-пам'яті з інтерфейсом SPI [13]. Мікроконтролер ESP8266 володіє наступними характеристиками:

— процесор Tensilica Xtensa на базі 32 бітної архітектури і тактовою частотою 80 МГц;

— вбудований Wi-Fi модуль з підтримкою стандартів IEEE 802.11 b/g/n;

— підтримка режимів Wi-Fi - точка доступу, клієнт;

— підтримка алгоритмів шифрування WEP, WPA та WPA2;

— вбудований стек протоколів TCP/IP;

— 14 цифрових портів вводу-виводу;

— вбудовані SPI, I^2C та UART інтерфейси;

— 10-бітний аналого-цифровий перетворювач;

— напруга живлення 3,0-3,6 вольт.

Процесор не має вбудованої енергонезалежної флеш-пам'яті, тому на модулі поряд з ним, розміщено мікросхему пам’яті. Процесор підтримує до 16 Мбайт зовнішньої пам'яті. Обмін даними з мікросхемою здійснюється через інтерфейс SPI. Процесор, динамічно завантажує необхідні фрагменти програми для виконання, в оперативну пам’ять. Також він підтримує стандартний, подвійний та квадратичний режими роботи з пам’яттю.

Програмні засоби розробки (SDK) складаються з:

— Компілятора для Xtensa LX106 входить в пакет компіляторів GNU Compiler Collection. Компілятор має відкритий вихідний текст. У різних SDK можуть міститися різні збірки цього компілятора, трохи відрізняються підтримуваними опціями.

— Бібліотек для роботи з периферією контролера, стеків протоколів IEEE 802.11 b/g/n Wi-Fi, TCP/IP.

— Засобів, які дають можливість завантажувати файли в пам'ять мікроконтролера та розбивати її на функціональні сегменти.

— Опціонального IDE для розробки.

Компанія Espressif Systems вільно поширює свій комплект програмного забезпечення для розробки. У цей комплект входить компілятор GCC, бібліотеки Espressif і завантажувальна утиліта XTCOM. Бібліотеки поставляються в вигляді скомпільованих бібліотек, без вихідного коду. Espressif підтримує дві версії SDK: перша - на основі операційної системи реального часу, інша на основі зворотніх викликів. Крім офіційної SDK існує ряд проектів альтернативних SDK. Вони використовують бібліотеки Espressif або пропонують власний аналог бібліотек, отриманий методом зворотньої розробки. Найпоширенішими з них є:

— “Esp-open-sdk” [14] - покращена версія SDK з відкритим вихідним кодом, яка містить GCC компілятор і деякі бібліотеки Expressif.

— “GNU toolchain for esp8266” [15] - SDK з можливістю інтеграції в графічну оболонку Microsoft Visual Studio.

— “Unofficial Development Kit” [16] - SDK для операційних систем сімейства Windows. У комплект входить компілятор GCC власної збірки з можливістю інтеграції у графічну оболонку IDE Eclipse.

— “Arduino IDE for ESP8266” [17] - додаток до Arduino IDE, що дозволяє програмувати ESP8266 так само, як будь-які інші модулі Arduino. При цьому доступна мережева функціональність ESP8266. Утиліта ESPTool, яка входить у комплект, дозволяє завантажувати програму у флеш-пам'ять пристрої лише одним натисненням кнопки.

— “Sming” [18] - проект, який дозволяє додавати Arduino сумісні бібліотеки поверх стандартних бібліотек Espressif, не використовуючи препроцесор Arduino, тобто розробка програм ведеться на чистому C.

Для спрощення застосування мікроконтролера в типових проектах можливе використання готових бінарних файлів, придатних до прямого завантаження в флеш-пам'ять. Готові вбудовані програми можна розділити на кілька груп відповідно до концепції їх використання:

— Для роботи під керуванням зовнішнього контролера. В них реалізовано задання параметрів роботи через зовнішній контролер UART.

— З вбудованими інтерпретаторами різноманітних мов високого рівня. Ці вбудовані програми дозволяють передавати через UART команди і таким чином виконувати скрипти розробника пристрою.

— Для інтернету речей. Цей клас вбудованих програм дозволяє з одного боку підключити до ESP8266 набір датчиків і виконавчих пристроїв, а з іншого боку надає необхідну мережеву функціональність для роботи в інфраструктурі Internet of Things.

— Такі, що реалізують віртуальний перехідник UART-WiFi.

На рис. 2.7 [19] показано повний перелік всіх виводів та їх призначення для Wi-Fi модуля на базі мікроконтролера ESP8266. Джерело виконуваної програми модуля визначається станом портів GPIO2 і GPIO0 в момент встановлення високого рівня на виводі EXT_RSTB. Найважливішими є два режими: виконання коду з UART (GPIO2=1, GPIO1=0) та із 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.

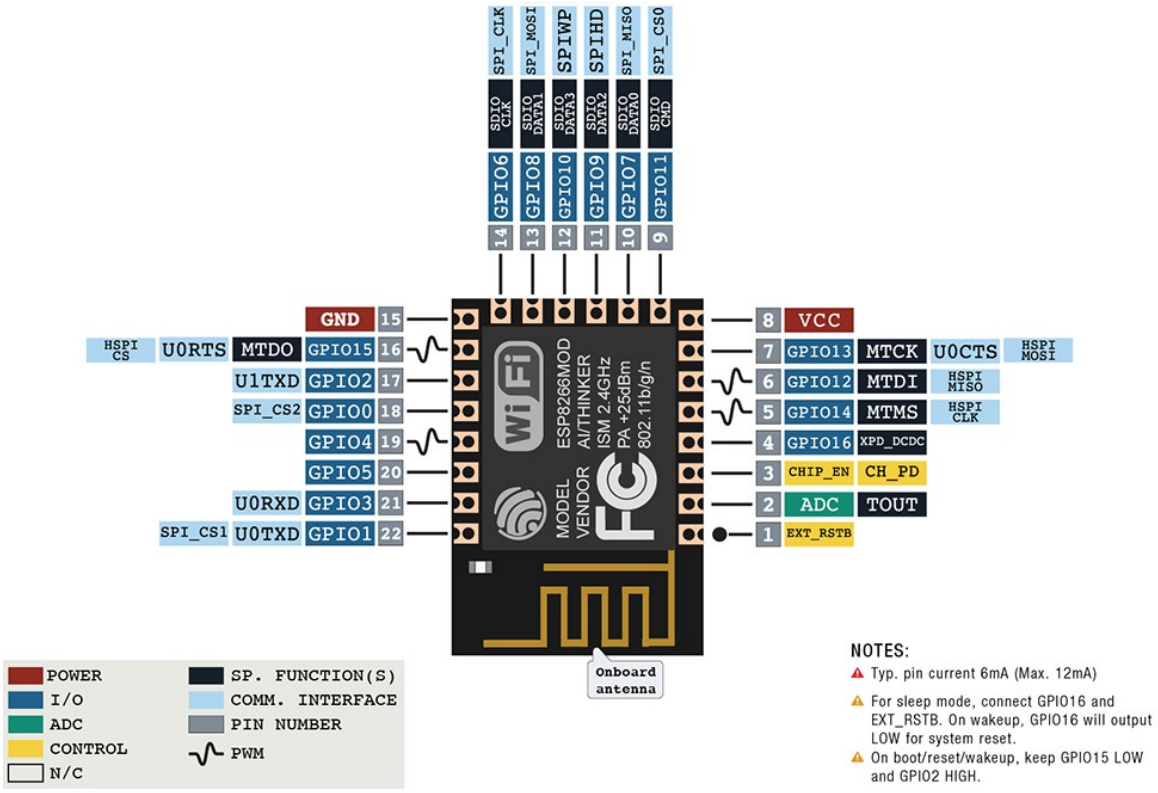

Рис. 2.7. Призначення виводів модуля ESP8266

Слід зазначити, що в пристрої передбачена можливість оновити вбудовану програму під час роботи через Wi-Fi. Для цього розділяють флеш-пам'ять для програм на кілька частин. Одна відводиться менеджеру вбудованих програм, дві інші - під програми користувача. Коли необхідно оновити вбудовану програму, новий образ завантажують у вільну частину флеш-пам'яті. Після ретельної перевірки цілісності щойно завантаженого образу менеджер перемикає маркер, після чого ділянка пам'яті зі старою вбудованою програмою звільняється, а виконання коду починається з нової ділянки.

ESP8266 може працювати як в режимі точки доступу так і в режимі клієнта. При нормальній роботі в локальній мережі ESP8266 працює у режимі клієнта. Для цього пристрою необхідно задати SSID Wi-Fi мережі і, в закритих мережах, пароль доступу. Для початкового налаштування цих параметрів зручно використовувати режим точки доступу. У цьому режимі пристрій видно при стандартному пошуку мереж в планшетах і комп'ютерах. Для того щоб задати параметри мережі, необхідно підключитися до пристрою, за допомогою веб-браузера відкрити HTML сторінку конфігурації та у відповідних полях задати ім’я локальної Wi-Fi мережі та її пароль. Після цього пристрій штатно підключиться до мережі в режимі клієнта.

Після підключення до мережі Wi-Fi платформа повинна отримати IP-адрес локальної мережі. Ці параметри можна задати вручну разом з параметрами Wi-Fi або активізувати будь-який сервіс, який буде динамічно отримувати адресу від DHCP-сервера.

Після налаштування усіх параметрів, звернутися до модуля в локальній мережі можна за його IP-адресою, мережевим ім'ям (якщо увімкнено підтримку протоколу NetBIOS) або сервісу (якщо увімкнено автоматичний пошук сервісів, наприклад через протокол SSDP). Найчастіше доступ до платформи здійснюється через мережу Інтернет. Наприклад користувач з мобільного телефону віддалено перевіряє стан свого “розумного будинку”, звертаючись безпосередньо до пристрою. В цьому випадку пристрій працює в режимі сервера, до якого звертається зовнішній клієнт.

Як правило, платформа на основі ESP8266 знаходиться в локальній мережі офісу або будинку. Вихід в Інтернет забезпечує маршрутизатор, підключений з одного боку до локальної мережі а з іншого - до мережі провайдера інтернет-зв’язку. Провайдер призначає маршрутизатору свою статичну або динамічну IP-адресу і маршрутизатор здійснює трансляцію адрес локальної мережі в мережу провайдера. За замовчуванням така конфігурація забезпечує вільну видимість інтернет-адрес у локальній мережі, але не дозволяє звернутися до локальних адрес з боку мережі Інтернет.

Є три основних способи обійти це обмеження. Більшість сучасних маршрутизаторів дозволяють задати додаткові правила трансляції мережевих адрес між локальною та глобальною мережами. Як правило для цього використовуються технологія DMZ. Обидві технології дозволяють звернутися до сервера в локальній мережі з глобальної мережі, знаючи лише IP-адресу, видану маршрутизатору провайдером. Однак такий підхід не завжди зручний: необхідно вручну налаштовувати маршрутизатор і з'ясовувати його IP-адресу, яка може регулярно змінюватися.

Для вирішення цієї проблеми можна скористатися спеціальними інтернет-сервісами, які використовують технологію DDNS. Вони працюють як спеціальні сервера з фіксованими іменами в мережі Інтернет. Користувачу надається можливість створити на такому сервісі свій обліковий запис з унікальним ім'ям. Параметри, згенеровані сервісом, прописуються в налаштуваннях модуля. Тоді він, у режимі клієнта, періодично звертається до DDNS-сервера, повідомляючи йому ім'я свого облікового запису і свою поточну IP-адресу. Кінцевий користувач у мережі Інтернет звертається до цього ж сервісу і отримує від нього поточну IP-адресу. В даному випадку модуль в мережі видно з доменним ім'ям третього рівня, наприклад roman.ddns.org. Основна проблема DDNS-сервісів - це гарантії існування конкретного сервісу. Як правило, гарантується тільки комерційний сервіс, коли за його використання стягується плата.

2.5. Використання LCD-дисплею

Для моніторингу та виведення налагоджувальної інформації шлюзу LoRa Mesh-мережі, було використано рідкокристалічний LCD-дисплей. Він дозволяє відображати на своєму екрані два рядки по шістнадцять символів. Рідкокристалічний дисплей побудовано на базі контролера HD44780.

Для зменшення кількості з'єднувальних провідників, дисплей з’єднується з платою по шині I^2C за допомогою модуля 2PH84388A, завдяки чому підключення здійснюється всього з допомогою 2 провідників для передачі даних й 2 провідників живлення: виводи SCL і SDA під'єднуються до цифрових портів вводу-виводу D1 і D2 відповідно (рис. 2.8). Модуль живиться напругою +3,3 вольт.

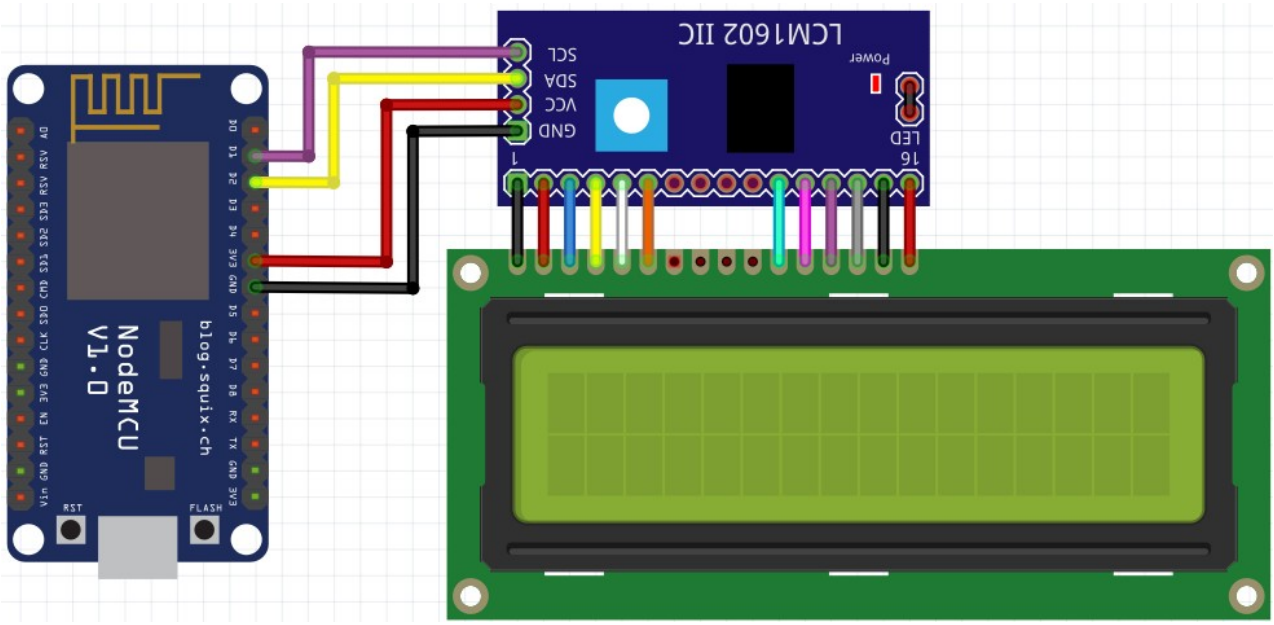

Рис. 2.8. Схема підключення LCD дисплею по шині I^2C

Резистор (голубий прямокутник з білим кругом у центрі) – використовується для регулювання контрастності LCD-дисплея. Перемичка LED призначена для ввімкнення підсвітки дисплею. Виводи 1…16 призначені для підключення даного модуля до виводів LCD-дисплея. Для зміни адреси пристрою потрібно вибрати відповідну комбінацію перемичками A0-A2.

Рідкокристалічний дисплей підключений по схемі із 4-бітною шиною даних до модуля 2PH84388A. Контролер посилає в LCD-дисплей команди, які керують режимами його роботи і ASCII-коди символів, що виводяться на його екрані. В свою чергу, він може посилати контролеру по його запиту інформацію про свій внутрішній стан і дані із своїх внутрішніх блоків пам'яті, але в даній схемі така можливість не передбачається (вивід R/W замкнутий на землю).

I^2C є послідовною шиною, тобто всі дані і адреси передаються по лінії SDA порозрядно [20]. Для передачі біта інформації, використовується тактовий сигнал на лінії SCL. Протягом всього часу дії сигналу SCL (SCL = 1) стан лінії SDA повинен залишатися незмінним [20]. Змінна даних на лінії SDA відбувається за умови відсутності тактового сигналу на лінії SCL (SCL = 0) [20]. Винятком є тільки біти, що призначені для визначення початку (стартовий біт) й завершення (стоповий біт) процесу обміном інформацією. Процес формування стартового біта інформації полягає у зміні рівня сигналу на лінії SDA з логічної “1” у логічний “0”, при SCL = 1, а стопового біта – при зміні сигналу SDA з логічного “0” у логічну “1”, також при SCL = 1 (рис. 2.9 [20]).

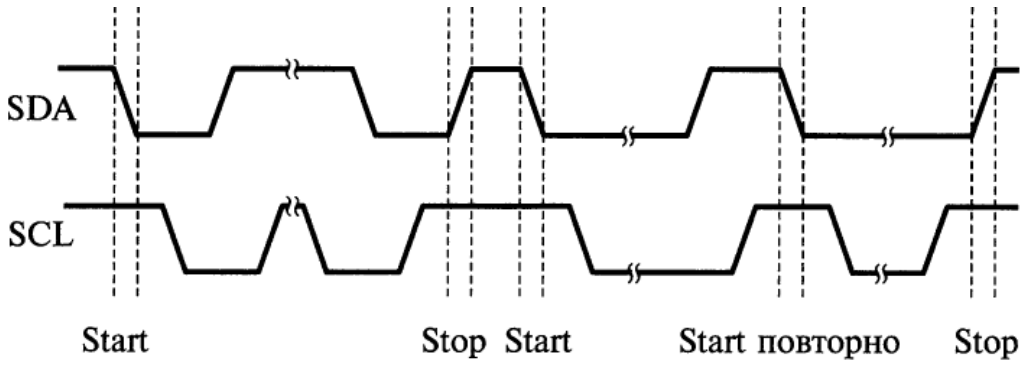

Рис. 2.9. Часова діаграма сигналів I^2C інтерфейсу

На модулі 2PH84388A, запаюючи відповідні перемички, можна змінювати адреси пристроїв. У даному проекті використовується LCD дисплей, що має по замовчуванні адресу 0x27.

Для того щоб передати шиною команду або дані потрібно виставити необхідний логічний рівень на виводі RS. Для супроводжує сигналів на шині

команд/даних LCD-дисплея використовується керуючий сигнал на лінії E [21]. У рідкокристалічний дисплей запис даних проходить по спадному фронту сигналу. Контролеру HD44780 після прийому байту даних або байту команди потребує певного часу для того щоб обробити отриману інформацію [21]. В цей момент модуль 2PH84388A не повинен передавати йому жлдних даних. Якщо модуль передає дані, які є ASCII-кодом символів, що відображаються, то вони записуються [21] в буфер даних (динамічна оперативна пам’ять), який зазвичай складається з вісімдесяти комірок.

Для того, щоб зрозуміти, як здійснюється передача символів на LCD-дисплей розглянемо, для прикладу часову діаграму виводу латинських літер “Q”, “R”, “S” на його екран.

Дані символи зберігаються у постійно запам'ятовувальному пристрої (ПЗП) LCD-дисплея й виводяться на екран простою передачею дисплею їх адреси. На рис. 2.10 зображено фото діаграми знятої з виводів RS, RW, E, D4, D5, D6 і D7, за допомогою аналізатора логічних рівнів.

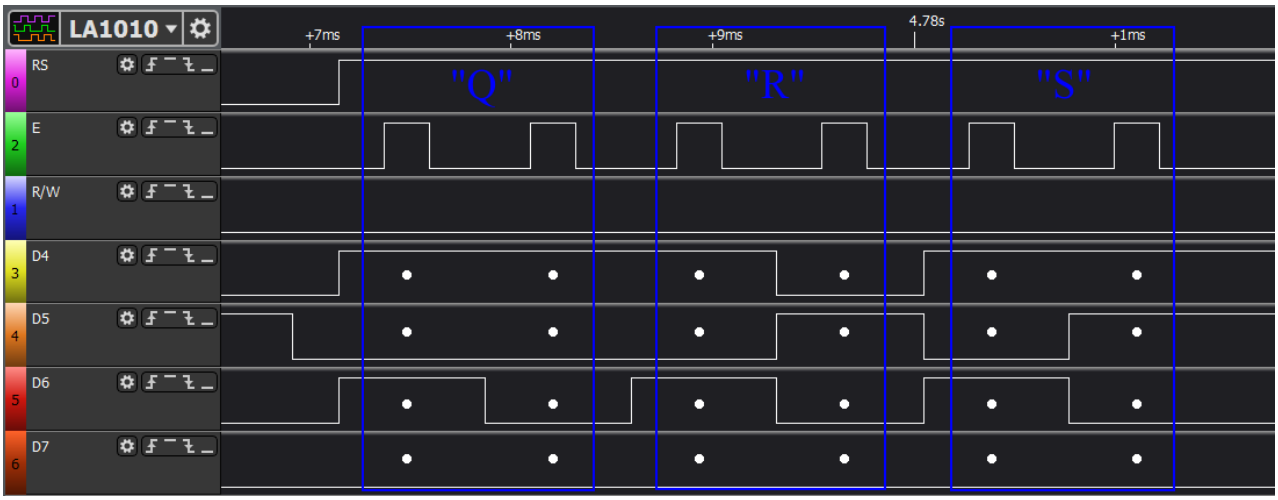

Рис. 2.10. Часова діаграма виводу латинських літер на LCD дисплеї

З діаграми видно, що символи які знаходяться в ПЗП дисплея передаються двома пів байтами, перший з яких визначає номер таблиці стовпців, а другий - номер рядка (див. рис. 2.11). При цьому дані “замикаються” по фронту сигналу на лінії E (Enable), а лінія RS (Register select) знаходиться в

стані високого рівня, що означає передачу даних. Стан низького рівня лінії RS означає передачу інструкцій, що й здійснюється на рис. 2.10, перед передачею кожного символу. Адже в цьому випадку передається код інструкції повернення вказівника на позицію (0,0) LCD-дисплея.

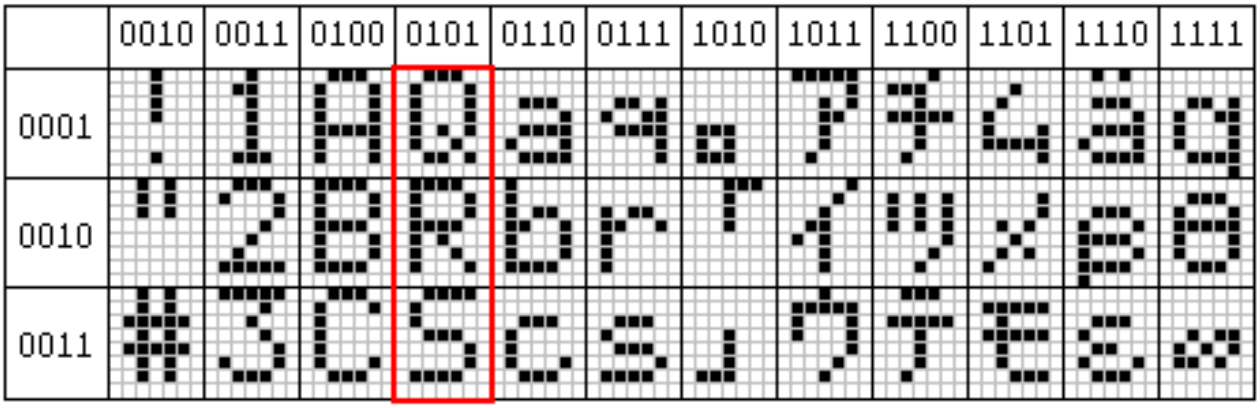

Рис. 2.11. Адреси символів, що містяться у ПЗП дисплея

2.6. Висновки до розділу

У даному розділі обґрунтовано підбір компонентної бази системи контролю розумного будинку. На основі аналізу параметрів та можливостей типового мережевого обладнання для побудови шлюзу вибрано Wi-Fi модуль ESP8266 з оригінальною конфігурацією, що має забезпечити відмовостійкість та автономність мережі загалом. Для кінцевих вузлів мережі обрано платформу Arduino Nano, побудовану на мікроконтролері Atmega 328p та LoRa радіомодулі RFM95 для безпроводового зв’язку. Обґрунтовано алгоритмічні підходи для оптимізації енергоефективності радіомодуля запропонованої модифікації системи контролю розумним будинком на основі трансивера RF96.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОЇ ЧАСТИНИ LoRa MESH-МЕРЕЖІ

3.1. Розробка програмної частин на платформі Arduino

Сучасна безпровідна LoRa Mesh-технологія радіозв'язку призначена для надійної передачі невеликих обсягів даних на великі відстані (кілька кілометрів) з використанням трансивера малої потужності. Модулі RFM95 відносно дешеві, але найпростіший спосіб використовувати цю технологію - придбати плати для розробки, які мають вбудований мікроконтролер ATmega328p. В даній роботі використовується платформа Arduino Nano, до якої під'єднано трансивер за допомогою шини SPI.

3.2 Налаштування послідовного периферійного інтерфейсу

RFM95 комунікує з хост-мікроконтролером, в даному випадку ATmega328p, через 4-дротовий послідовний периферійний інтерфейс (SPI), як залежний пристрій. SPI-сумісний послідовний інтерфейс дозволяє обирати, керувати і слідкувати за станом RFM95 через мікроконтролер. З’єднуючи трансивер з мікроконтролером, інтерфейсом SPI, потрібно використовувати провідники мінімальної довжини, так як частота шини досить висока.

Протокол SPI передбачає, що одним з пристроїв повинен бути керуючим. Зазвичай цю роль виконує мікроконтролер, який контролює підлеглі периферійні пристрої. Згідно протоколу, всі пристрої паралельно підключаються до трьох провідників шини даних, а саме:

— MISO (Master In Slave Out) - передача даних здійснюється від підлеглого пристрою (Slave) до керуючого (Master);

— MOSI (Master Out Slave In) - передача даних здійснюється від керуючого пристрою (Master) до підлеглого (Slave);

— SCK (Serial Clock) - лінія синхронізації процесу передачі даних, на якій генеруються тактові імпульси керуючим пристроєм.

До кожного підлеглого пристрою ще окремо підключається четверта лінія SS (Slave Select), яка призначена для активізації контролером того чи іншого периферійного пристрою. Передача даних між пристроєм і мікроконтролером розпочинається лише у випадку, коли на даній лінії присутній логічний “0”. В іншому випадку дані надіслані мікроконтролером будуть проігноровані. Особливістю такої архітектури є використання однієї спільної шини (MISO, MOSI і SCK) декількома підлеглими пристроями.

Для передачі даних SPI-інтерфейсом трансиверу RFM95 потрібно провести попередню конфігурацію шини. Бібліотека RadioHead підтримує програмну емуляцію SPI, тому його можна під’єднувати до будь-якого з 14 цифрових портів вводу-виводу (окрім RX і TX, оскільки вони зайняті шиною UART) платформи Arduino Nano.

У трансивера ще є лінія DIO0, якою він надсилає мікроконтролеру переривання. За замовчуванням вона підключається до другого цифрового входу платформи. При отримані переривання, мікроконтролер повинен негайно призупинити виконання програми і обробити його. Для перевизначення ліній DIO0 та SS, можна на початку програми оголосити конструктор (рис. 3.1).

```
// Slave Select is pin 10, interrupt is Pin 9
RH_RF95 driver(10, 9);
```

Рис. 3.1. Оголошення конструктора “driver()”

Оскільки трансивер використовує запис словом, пін SS повинен бути підтягнутим до нижнього рівня при передачі всіх 16 бітів. Біти даних на піні MISO передаються в пристрій до підйому в такті на піні SCK поки пін SS має низький логічний рівень.

3.3. Робота з EEPROM пам'яттю

Кожен пристрій у LoRa Mesh-мережі повинен мати унікальний ідентифікатор. Він дозволяє пристроям відрізняти один одного і правильно формувати таблицю маршрутизації.

В процесі розробки та експлуатації вбудована програма мікроконтролера може часто змінюватися. Тому зручно унікальний ідентифікатор зберігати у внутрішній енергонезалежній пам’яті мікроконтролера - EEPROM.

ATmega328P містить 1 Кбайт EEPROM пам'ятті. Вона організована як окремий простір даних, у якому одиночні байти можна читати і записувати. Ресурс EEPROM пам'яті становить приблизно 100000 циклів стирання/запису. Регістри для роботи з пам'яттю доступні в просторі вводу-виводу. Запис даних у EEPROM пам'ять не можна виконувати у першу секунду роботи платформи, оскільки при встановлених конденсаторах високої ємності на лінії живлення напруга може ще не досягти необхідного рівня для коректного проведення цієї операції. При читані або записі в пам’ять центральний процесор зупиняє виконання інструкцій на чотири або два тактові цикли відповідно.

EEPROM пам'ять також може бути пошкоджена при нестабільній напрузі живлення, оскільки її низький рівень може призвести до некоректного виконання інструкцій центральним процесором і роботи внутрішньої пам’яті. Це твердження також стосується і зовнішньої EEPROM пам'яті. Пошкодження даних у пам’яті може виникнути в двох випадках:

— коли запис даних регулярно відбувається в однакові комірки при низькій напрузі живлення;

— процесор може виконати некоректно процедуру запису, якщо напруга занадто низька.

Пошкодження даних у EEPROM пам'яті можна легко уникнути, якщо тримати внутрішній інверсний сигнал AVR RESET активним (у низькому логічному рівні) у період недостатньої напруги живлення. Це можна зробити, увімкнувши внутрішню систему контролю живленням (СКЖ). Якщо вона

виявить зниження напруги під час запису даних, то ця операція буде призупинена до нормалізації живлення.

На рис. 3.2 зображено фрагмент коду мовою С, для запису байту даних у комірку пам'яті. Під час виконання цієї функції потрібно заборонити глобальні переривання командою cli().

```
void EEPROM_write(uint8_t uiAddress, byte ucData) {
  // Очікування на завершення попереднього запису
  while(EECR & (1<<EEPE));
  // Запис адреси і даних в регістри
  EEAR = uiAddress;
  EEDR = ucData;
  // Запис логічної одиниці в регістр EEMPE
  EECR |= (1<<EEMPE);
  // Запис даних в EEPROM пам'ять
  EECR |= (1<<EEPE);
}
```

Рис. 3.2. Запис у EEPROM пам'ять

Також перед виконанням запису в EEPROM пам'ять потрібно переконатися чи всі операції пов’язані з перезаписом флеш-пам'яті завершено. Після завершення роботи з EEPROM пам'ятаю потрібно дозволити глобальні переривання командою sei().

На рис. 3.3 зображено фрагмент коду мовою С, для читання байту даних з комірки пам'яті. На момент виконання функції, глобальні переривання аналогічно потрібно заборонити, як і при виконанні запису.

```
byte EEPROM_read( uint8_t uiAddress) {
  // Очікування на завершення попереднього запису
  while(EECR & (1<<EEPE));
  // Запис адреси в регістр
  EEAR = uiAddress;
  // Читання даних з EEPROM пам'яті
  EECR |= (1<<EERE);
  return EEDR;
}
```

Рис. 3.3. Читання з EEPROM пам'яті

3.4. Синтаксис мови програмування Arduino

Програми для платформи Arduino пишуть на С-подібній мові. Кожна програма повинна містити дві основних частини (функції). Перед першою частиною, як правило, спочатку йдуть директиви, які починаються з символу “#”. Директива “#include” призначена для підключення зовнішніх бібліотек, а “#define” для оголошення ідентифікаторів, та їх значень. Під час компілювання препроцесор замінює всі директиви на відповідні фрагменти коду.

Після директив йде оголошення всіх глобальних змінних. Нарешті слідує функція setup(), яка виступає аналогом функції main() на мові С. Після компіляції препроцесор помічає її як точку входу для завантажувача Arduino. Вона виконується лише один раз під час виконання коду. Зазвичай, у ній ініціалізуються змінні та послідовні інтерфейси, налаштовуються порти вводу-виводу та записується необхідні значення у регістри процесора. Останньою частиною програми є функція loop() - петля. Із назви зрозуміло, що ця функція виконується постійно. Вона необхідна для того, щоб щоразу наприкінці setup() не використовувати конструкцію while(true){}. В цю функцію поміщається основний код програми (рис. 3.4).

```
// Директиви, глобальні змінні, константи...
void setup() {
  // Ініціалізація змінних, портів вводу-виводу...
}

void loop() {
  // Основний код програми
}
```

Рис. 3.4. Структура програми Arduino

Функція setup() обов'язково повинна бути включена в програму, навіть, якщо у ній немає необхідності. Вона викликається один раз, коли програма стартує (рис. 3.5).

```
void setup() {
pinMode (13, OUTPUT); // Встановлення 13-го піна як вихід
}
```

Рис. 3.5. Ініціалізація 13-го піна

Коли завершилося виконання функції setup() наступною починає виконуватися loop(). Постійне виконання цієї функції дозволяє програмі щось змінювати, відслідковувати сигнали з портів вводу-виводу та керувати платою Arduino. На рис. 3.6 зображено фрагмент коду, який дозволяє налаштувати миготіння світлодіода, що підключено до 13-го цифрового виходу, з періодом в одну секунду.

```
void loop() {
  // Встановлення логічної одиниці на 13 цифровому виході
  digitalWrite(13, HIGH);
  delay(1000); // пауза 1 секунда
  // Встановлення логічного нуля на 13 цифровому виході
  digitalWrite(13, LOW);
  delay(1000); // пауза 1 секунда
}
```

Рис. 3.6. Приклад програми “Blink”

Функція - це блок коду, що має ім'я, яке вказує на код, який виконується при виклику функції [22]. Переважно у функції поміщаються фрагменти коду, які потрібно виконувати декілька разів, що дозволяє підтримувати порядок у програмі. Насамперед для створення функції потрібно оголосити її тип. Йому повинен відповідати тип значення, який в подальшому буде повернений функцією. Часто функціям не потрібно повертати ніяких значень, тому для цього існує тип void. Далі після типу функції, вказується ім’я, та в дужках оголошуються її параметри, через які функції передаються конкретні значення.

Кожна функція має початок та кінець. Ці поля повинні бути виділені фігурними дужками. Також ними виокремлюються вирази типу if, for, else, while, do. Фігурні дужки в програмі завжди ідуть парами (відкривши дужку,

ніколи не потрібно забувати її закрити) – тобто є так-званими “збалансованими” дужками (рис. 3.7).

```
тип ім’я_функції(параметри) {
  // Тіло функції
   return // Значення яке повертає функція
}
```

Рис. 3.7. Оголошення функції

В кінці виразу і для відокремлення елементів в програмі повинна використовуватися крапка з комою. У циклі for всі елементи також розділяються крапкою з комою.

Деякі фрагменти коду зручно помічати коментарями. Мова Arduino підтримує два види коментарів. Одним із них це блок коментаря (рис. 3.8). Позначається в коді він наступним чином “/* ... */”.Всі коментарі видаляються препроцесором під час компілювання. Завдяки коментарям, програміст може зрозуміти певну частину коду програми. Кожен блок коментаря повинен починатися символами “/*” і як у випадку фігурних дужок, він повинен закінчуватися групою символів “*/”. Оскільки коментарі на етапі компіляції видаляються препроцесором, то вони не займають пам’яті мікроконтролера.

```
/ * Це “відгороджений” блок коментаря,
він повинен бути збалансований!
Коментар може містити декілька рядків!* /
```

Рис. 3.8. Блок коментаря

Крім блоку коментаря, ще існує однорядковий коментар (рис. 3.9). Він починається символами “//” та закінчується (внутрішнім) кодом переходу на інший рядок [22] (\r\n - для Windows і \n - для Unix та Mac).

```
// Ось так виглядає однорядковий коментар
```

Рис. 3.9. Однорядковий коментар

В процесі відлагодження часто коментують фрагменти коду, щоб компілятор їх не бачив. Доцільним є видалення такого типу коментарів після завершення відлагодження програми.

Змінні - це спосіб іменувати і зберігати числові значення для подальшого використання програмою [22]. На рис. 3.10 зображено фрагмент коду, в якому оголошеній змінні типу uint16_t, спочатку присвоюється нуль, а потім прочитане значення з другого аналогового входу:

```
// Оголошується цілочисельна змінна, які присвоюється 0
uint16_t inputVariable = 0;
// Змінній присвоюється значення другого аналогового входу
inputVariable = analogRead(A2);
```

Рис. 3.10. Оголошення змінної

Якщо в коді критично важлива розрядність цілочисельних типів даних, то краще використовувати int8_t, int16_t, int32_t, int64_t. Їм можуть бути присвоєні як додатні, так і від’ємні значення. Якщо додати префікс “u”, наприклад uint8_t, то всі вісім біт виділяються лише для додатного значення. Загалом є доцільним при написанні коду використовувати саме такі типи, оскільки це пов'язано з можливою несумісністю різних компіляторів і, відповідно, проблемним перенесенням програмного коду на іншу платформу.

Змінні можуть бути оголошені на початку програми перед void setup(), локально всередині функцій, і іноді в блоці виразів такому, як цикл for [22]. Змінна повинна бути оголошена до моменту її використання. В залежності від місця оголошення змінної визначаються її межі (область видимості), тобто можливість деяких частин програми її використовувати.

Зазвичай на початку програми завжди декларуються глобальні зміні. До них мають доступ всі вирази та функції програми. Не варто оголошувати багато глобальних змінних, оскільки вони завжди знаходяться в невеликій оперативній пам’яті платформи Arduino.

Локальні змінні оголошуються у виразах типу if, else, for, while, do та всередині функцій. Вони видимі лише всередині оголошеного виразу чи функції. Таким чином, можна створювати кілька змінних з однаковими іменами в різних частинах однієї програми. Змінна існує в оперативній пам'яті лише під час виконання конкретного виразу чи функції, що значно оптимізує виконання програми.

На рис. 3.11 зображено приклад коду, який наочно демонструє область видимості оголошених змінних.

```
uint8_t var; // "var" видима для будь-якої функції
void setup() {
  // Пуста функція setup()
}

void loop() {
  for (uint8_t i = 0; i < 255; i++) {
    // Змінна "i" видима лише в середині циклу for
  }
  // Тут змінна "i" недоступна
  int16_t d; // ця змінна видима в середині функції loop()
}
```

Рис. 3.11. Область видимості оголошених змінних

У мові Arduino доступні всі чотири базові операції, а саме додавання, віднімання, множення та ділення. Вони повертають суму, різницю, добуток або частку двох операндів відповідно (рис. 3.12).

```
a = a + 0;
s = s - 2:
m = m * 8;
d = d / 4;
```

Рис. 3.12. Базові арифметичні операції мови Arduino

Якщо для ділення використовувати цілочисельний тип, наприклад, при операції 5/2, вираз завжди буде повертати 2 замість 2,5. Також при операціях множення або додавання може виникати переповнення, якщо результат не поміщається у тип оголошеної змінної. Для уникнення цієї проблеми можна скористатися табл. 3.1 [23].

Таблиця 3.1

Цілочисельні типи даних платформи

Тип	Тип, звичайний запис	Діапазон значень
int8_t	char	від -128 до 127
uint8_t	byte, unsigned char	від 0 до 255
int16_t	int	від - 32768 до 32767
uint16_t	unsigned int, word	від 0 до 65535
int32_t	long	від -2147483648 до 2147483647
uint32_t	unsigned long	від 0 до 4294967295

Логічні оператори повертають результат порівняння двох виразів та повертають TRUE або FALSE. Існує три оператори: AND, OR і NOT (рис. 3.13). Мова Arduino дозволяє замість AND і OR використовувати && і || відповідно.

```
// Повертає true, якщо обидва вирази також повертають true
if (x > 0 && x < 5)   // Логічне AND
// Повертає true, будь-який з виразів також повертають true
if (x > 0 || y > 0)   // Логічне OR
// Повертає true, якщо вираз повертає false
if (!x > 0)   // Логічне NOT
```

Рис. 3.13. Базові логічні операції мови Arduino

Для того, щоб програму було зручно читати, в мові Arduino передбачено декілька наперед визначених величин, так званих констант. Їх можна розділити на наступні групи:

— Група true/false – це булеві константи. В програмі ними можна визначати логічні рівні. Зрозуміло, що TRUE відповідає одиниці, а FALSE -

нулю. Змінним типу bool/boolean, які приймають булеві значення, компілятор виділяє вісім біт пам'яті (рис. 3.14). Якщо такій змінній присвоювати значення -1, 3, 10, чи 127, то компілятор застосує неявне перетворення типів і всі значення будуть приведені до TRUE.

```
if (b == TRUE) {
  // Вираз буде істинний, якщо змінна b не рівна нулю
}
```

Рис. 3.14. Приклад використання оператора if

— Група high/low – визначають рівень сигналу виводів. Очевидно, що HIGH відповідає високому логічному рівню, а LOW — низькому. Вони використовуються при читанні або записі логічних рівнів на цифрових портах вводу-виводу (рис. 3.15).

```
// Встановлення високого логічного рівня на 13 виводі
digitalWrite(13, HIGH);
```

Рис. 3.15. Встановлення логічного рівня на піні

— Група input/output – константи, які дозволяють встановити режим роботи цифрових портів на вхід або на вихід. Зазвичай вони використовуються з функцією pinMode().

Для встановлення режиму цифрових портів вводу-виводу, можна використовувати функцію pinMode() або записувати значення напряму в регістри портів. За замовчуванням у платформі Arduino цифрові порти вводу-виводу вже сконфігуровані на вхід. В такому стані вони мають високий внутрішній опір. При підключені до них кнопки, зручно використовувати внутрішні підтягуючі резистори номіналом 22-47 кОм, в залежності від моделі та параметрів мікроконтролера (рис. 3.16).

```
pinMode(13, INPUT);  // Встановити порт 13 на вхід
digitalWrite(13, HIGH);  // Увімкнути підтягуючий резистор
```

Рис. 3.16. Увімкнення внутрішнього підтягуючого резистора

Якщо налаштувати цифрові піни як OUTPUT (вихід), то вони будуть знаходитися в низькоімпедансному стані. В такому режимі вони можуть віддавати лише 35 мА. Цього струм більш ніж достатньо для яскравого світіння світлодіода. Якщо перевантажити вивід, то це може призвести до пошкодження внутрішнього польового транзистора, який керує ним, або навіть до виходу з ладу всієї плати.

Функція digitalRead(pin) використовується зчитування значення заданого цифрового входу (pin). Вона повертає значення HIGH або LOW в залежності від його логічного стану. Протилежною до неї є функція digitalWrite(pin, value), яка встановлює логічний рівень на цифровому виході (pin). Вихід можна задавати у вигляді змінної або константи в діапазоні від 0 до 13.

На рис. 3.17 зображено приклад коду, який дозволяє прочитати значення з кнопки, з'єднаної з цифровим входом, та увімкнути розміщений на платі світлодіод, який під’єднано до 13-го цифрового виходу.

```
uint8_t led = 13; // Вихід на світлодіод
uint8_t pin = 2; // Вхід з кнопки
uint8_t var = 0; // Змінна для зберігання зчитаного значення

void setup() {
  pinMode(led, OUTPUT); // Визначаємо 13 пін як вихід
  pinMode(pin, INPUT);  // Визначаємо 2 пін як вхід
  digitalWrite(pin, HIGH);  // Вмикаємо підтягуючий резистор
}
void loop() {
 var = digitalRead(pin); // Задає var рівним вхідному виводу
 digitalWrite(led, var); // Присвоює led значення кнопки
}
```

Рис. 3.17. Читання логічного рівня на цифровому піні

Функція delay() використовується для того, щоб призупинити виконання програми на заданий час (в мілісекундах), де 1000 дорівнює 1 секунді. Схожою на неї є функція delayMicroseconds(), яка набуває значень в мікросекундах. Для

призупинення виконання програми на менший часовий проміжок доцільно скористатися асемблерною вставкою (рис. 3.18).

```
delay(1000); //призупиняє виконання програми на 1 секунду
delayMicroseconds(10);  //на 10 мікросекунд
__asm("nop") // на один процесорний цикл
```

Рис. 3.18. Способи призупинити виконання програми на деякий час

Для того, щоб ініціалізувати послідовний порт і задати швидкість для послідовної передачі даних, можна скористатися функцією Serial.begin(115200). Типова швидкість обміну для комп'ютерної комунікації – 115200 бод, хоча підтримуються й інші швидкості (рис. 3.19).

```
void setup(){
  // Відкриваємо послідовний порт на швидкості 115200 бод
  Serial.begin(115200);
}
```

Рис. 3.19. Ініціалізація послідовного інтерфейсу

3.5 Керування LoRa трансивером RFM95

Для роботи з LoRa трансивером RFM95 було обрано пакет бібліотек RadioHead. У його комплект входять бібліотеки, які підтримують більшість популярних на сьогоднішній день радіомодулів. Пакет бібліотек є повністю об'єктно-орієнтований і дозволяє надсилати та отримувати повідомлення у формі пакетів даних. Бібліотеки підтримують більшість платформ Arduino, зокрема Arduino Nano та Raspberry Pi.

RadioHead складається з двох основних наборів класів: драйвера та менеджера. Драйвер забезпечує низькорівневий доступ до внутрішніх регістрів трансивера. Він дозволяє ефективно керувати модулем, використовуючи лише прості виклики. В свою чергу, менеджер надає високорівневий доступ для управління процесом надсилання та отримання пакетів даних.

У кожній програмі, яка використовує пакет бібліотек RadioHead, потрібно створити драйвер для забезпечення доступу до радіомодуля, і, як правило менеджер, який буде використовувати цей драйвер для надсилання та отримання повідомлень програмою. Після їх створення обов'язково потрібно ініціалізувати менеджер. Якщо цей етап було успішно виконано, то можна легко надсилати та отримувати повідомлення, використовуючи спеціальні команди.

На рис. 3.20 зображено фрагмент коду мовою С, який демонструє використання менеджера RHMesh. Перед його створенням потрібно прочитати унікальний ідентифікатор пристрою з EEPROM пам'яті та записати його у зміну "nodeId".

```
#include <RHMesh.h>
#include <RH_RF95.h>

// Створення конструкторів
RH_RF95 driver;
RHMesh *manager;

void setup() {
  manager = new RHMesh(driver, nodeId);
  // Ініціалізація менеджера
  manager -> init();
  // Встановлення початкових налаштувань трансивера
  driver.setTxPower(23, false);
  driver.setFrequency(915.0);
  driver.setCADTimeout(500);
}

void loop() {
  // Код основної програми
}
```

Рис. 3.20. Приклад використання пакету бібліотек RadioHead

Драйвер можна використовувати самостійно, без менеджера, але такий режим роботи є ненадійним в плані передачі даних. Усі драйвери мають

однаковий API. У деяких випадках пакет бібліотек дозволяє ініціалізувати більше одного драйвера та менеджера і використовувати їх.

Для використання фіксованої довжини пакетів та забезпечення надійності передачі повідомлень, RadioHead підтримує чотири основних менеджери:

— RHDatagram - дозволяє широкомовно розсилати повідомлення змінної довжини за вказаними адресами без підтвердження їх отримання.

— RHReliableDatagram - дозволяє широкомовно розсилати повідомлення змінної довжини за вказаними адресами з підтвердження їх отримання.

— RHRouter - дозволяє доставляти повідомлення за вказаними адресами через один або більше проміжних вузлів з підтвердження їх отримання, використовуючи попередньо запрограмовану таблицю маршрутизації.

— RHMesh - дозволяє доставляти повідомлення за вказаними адресами через один або більше проміжних вузлів з підтвердження їх отримання, з автоматичним складанням таблиці маршрутизації, використовуючи Mesh-технологію.

Спочатку слід зазначити складність тестованої Mesh-мережі. Доволі важко розмістити вузли таким чином, що безпосередньо могли взаємодіяти між собою тільки деякі вузли, оскільки для реалізації цього потрібно було б їх розмістити на великій відстані один від одного по всьому місту. Корисною передбаченою можливістю для моделювання такої ситуації у бібліотеці RadioHead є декілька тестових мереж, визначених таким чином, щоб можна було змусити деякі вузли не мати можливості взаємодіяти між собою. Коли визначається така мережа, бібліотека RadioHead просто ігнорує повідомлення, отримані від вузлів, які не повинні знаходитися в радіусі дії конкретного модуля. Це дозволяє значно простіше змоделювати роботу мережі (рис. 3.21).

```
#elif RH_TEST_NETWORK==3
  // Ця мережа виглядатиме так 1-2-4
  //                           |   |
  //                           --3--
```

Рис. 3.21. Визначення обмеження для третього вузла мережі

Тобто, вузли 2 і 3 та 1 і 4 не можуть безпосередньо взаємодіяти один з одним [24]. Кожен вузол намагається з'єднатися з будь-яким іншим вузлом у мережі, і в процесі він відстежує таблицю маршрутизації, яка описує, з якими вузлами він може безпосередньо взаємодіяти, і яким вузлам передавати повідомлення, якщо немає прямого з'єднання. Вузол також відстежує рівень сигналу від інших вузлів. У результаті кожен вузол має структуру даних з цією інформацією (рис. 3.22).

{"3": [{"n":1,"r":-95}, {"n":1,"r":0}, {"n":255,"r":0},
{"n":0,"r":0}]}

Рис. 3.22. Зразок таблиці маршрутизації (виражено в JSON) для вузла 3

Таблиця представлена у вигляді масиву з 4 записів, по одному для кожного вузла в мережі. Записи, наведені вище, представляють собою інформацію маршрутизації для цього вузла, сполученого з вузлами 1, 2, 3 і 4 відповідно. Кожен запис має дві властивості. Властивість “n” - це ідентифікація вузла, з яким повинен спілкуватися вузол 2, тобто запис {“n”: 1, “r”: -95} означає, що вузол 3 може спілкуватися з вузлом 1 через вузол 1, тому він успішно може зв'язується безпосередньо з вузлом 1 з рівнем сигналу -95 дБм.

Запис 2 {“n”: 1, “r”: 0} означає, що вузол 3 повинен зв'язуватися з вузлом 2 через вузол 1, оскільки немає прямого зв'язку (саме тому значення RSSI дорівнює 0). Запис 3 {“n”: 255, “r”: 0} має значення “n”=255, що означає сам вузол, тому його можна ігнорувати. Запис 4 {“n”: 0, “r”: 0} має властивість “n”=0, що означає, що вузол 3 ще не знайшов способу спілкування з вузлом 4, тому що цей вузол тимчасово недоступний або шлях до нього ще не було знайдено. З часом, намагаючись відправити повідомлення в будь-який інший вузол, кожен вузол створює таблицю маршрутизації про те, з яким вузлом він може спілкуватися і як саме передавати повідомлення від інших вузлів, а також про рівень сигналу кожного з них. Інформація, надіслана в повідомленнях, є самою таблицею маршрутизації вузла. Саме тому зручно представляти

таблицю маршрутизації як рядок JSON і використовувати скорочені імена властивостей.

3.6. ESP8266, як основа LoRa Mesh-шлюзу

Шлюз дозволяє WMN отримати доступ до мережі Інтернет та керувати нею за допомогою комп’ютера або звичайного смартфона. Шлюз побудовано на базі власної розробленої платформи, яка працює під управлінням мікроконтролера ESP8266, з підтримкою інтерфейсу Wi-Fi. Шлюз підтримує можливість виконувати програми з зовнішньої флеш-пам'яті. Було вирішено розділити пам’ять шлюзу на два блоки: 1 та 3 мегабайти. В першому блоці буде знаходитись виконавчий код програми, а в другому — файлова система Serial Peripheral Interface Flash File System [25] (SPIFFS). Туди можна завантажити всі необхідні файли для правильної роботи серверів. Вони включають в себе різні HTML сторінки, файл стилів CSS, та багато конфігураційних файлів для правильної роботи трансивера, протоколів маршрутизації, MQTT та веб-серверів.

При увімкнені трансивер спочатку ініціалізує файлову систему та завантажує необхідні конфігураційні файли. Вся налагоджувальна інформація виводиться на рідкокристалічний дисплей. Після того він переводить Wi-Fi модуль у режим клієнта і намагається (протягом п’ятнадцяти секунд) підключитися до точки доступу, яку вказано в одному з конфігураційних файлів. Якщо спроба підключення була невдалою, то контролер переводить Wi-Fi модуль у режим точки доступу, після чого запускає MQTT-клієнт. Він в свою чергу намагається підписатися на MQTT-брокер, використовуючи адресу, порт, логін і пароль з відповідних конфігураційних файлів. Останнім етапом запуску шлюзу є ініціалізація веб-сервера та всіх його API.

Для коректної роботи та взаємодії з мережею, необхідно щоб з боку клієнта пристрій підтримував будь-який сучасний веб-браузер (класу Opera,

Google Chrome, Mozilla Firefox тощо) та виконувалися мінімальні апаратні вимоги (256 Мб RAM, 1 ГГц процесор).

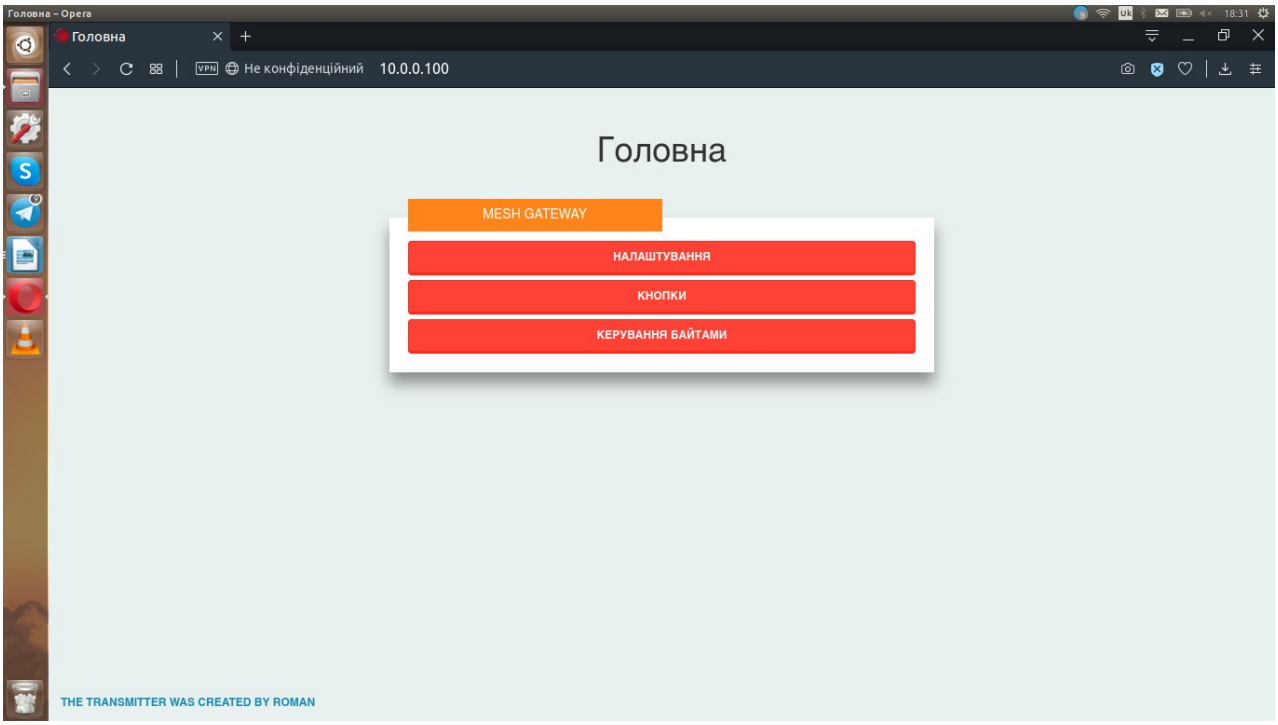

Рис. 3.23. Початкова сторінка LoRa Mesh-шлюзу

Доступ до керування мережею здійснюється через веб-інтерфейс за допомогою звичайного браузера (рис. 3.23). Інтерфейс є досить простим, що дозволяє швидко та зручно працювати з елементами управління пристрою. Для входу на сторінку адміністрування мережі потрібно перейти за відповідною статичною IP адресою пристрою 192.168.0.1, за умови, якщо шлюз перейшов у режим точки доступу. В іншому випадку він буде доступний у переліку пристроїв, які знаходяться у локальній мережі, або після виконання команди в терміналі “gssdp-discover -i wlo1 --timeout=3” (де wlo1 - мережевий пристрій, який знаходиться в локальній мережі з шлюзом), яка дозволить дізнатися динамічний адрес та переглянути службову інформацію. Якщо підключено рідкокристалічний дисплей, то динамічну адресу буде виведено на його екрані. Керування мережею здійснюється за допомогою режиму користувача.

Користувач має доступ до всіх параметрів управління платформою. Вхід здійснюється за допомогою аутентифікації по захищеному каналу зв’язку.

3.7. Робота з протоколом MQTT

MQTT — це протокол прикладного рівня моделі OSI, який працює на стеку протоколів TCP/IP. Він використовується для обміну повідомленнями між пристроями за принципом видавець-підписник [26]. У проекті протокол використовується лише для передачі таблиці маршрутизації у реальному часі, щоб візуалізувати структуру Mesh-мережі.

Першу його версію в 1999 році опублікували Енді Стенфорд-Кларк з IBM і Арлен Ніппер з Cirrus Link. Вони розглядали MQTT як спосіб підтримки зв'язку між пристроями в мережах з обмеженою пропускною здатністю. З урахуванням умов експлуатації протокол зроблений маленьким і легким. Він підходить для пристроїв з низькою обчислювальною потужністю і обмеженим часом автономної роботи. Тому протокол є ідеальним рішенням для систем розумного будинку. Система зв'язку, побудована на MQTT, складається з сервера-видавця, сервера-брокера і одного або декількох підписників. Видавець не вимагає яких-небудь налаштувань за кількістю або розташуванням підписників, які отримують повідомлення. Крім того, їх не потрібно налаштовувати на конкретного видавця. В системі може бути декілька брокерів, що поширюють повідомлення.

MQTT надає можливість створення ієрархії каналів зв'язку – подібно до гілки з листям. Кожного разу, коли у видавця є нові дані для поширення серед підписників, повідомлення супроводжується приміткою контролю доставки. Підписники більш високого рівня можуть отримувати кожне повідомлення, в той час як підписники більш низького рівня можуть отримувати лише повідомлення, які належать тільки до одного або двох базових каналів, що відгалужуються від нижньої частини ієрархії. Це полегшує обмін інформацією розміром від двох байт до 256 мегабайт.

Для того, щоб налаштувати MQTT-клієнт на LoRa Mesh-шлюзі, можна скористатися бібліотекою PubSubClient. Вона дозволяє легко підключитися до MQTT-брокера, а функції publish() та subscribe() дозволяють публікувати або підписуватися на топіки MQTT (Topic Name). На рис. 3.24 зображено приклад мовою С, як можна налаштувати клієнт для підключення до MQTT-брокера.

```
bool mqtt_connect() {
  // Читаємо необхідні дані з конфігураційного файлу
  char *_clientId = jsonRead(configSetup, "SSDP").c_str();
  char *_user = jsonRead(configSetup, "mqttUser").c_str();
  char *_pass = jsonRead(configSetup, "mqttPass").c_str();
  byte tries = 5;
  // Пробуємо встановити з'єднання
  while (!mqtt_client.connected() && --tries) {
    // Якщо з'єднання встановлено
    if (mqtt_client.connect(_clientId, _user, _pass)) {
      // Повертаємо TRUE
      return true;
    } else {
      // Чекаємо одну секунду і повторяємо спробу
      delay(1000);
    }
  }
  // Якщо встановити з'єднання не вдалося то повертаємо FALSE
  return false;
}
```

Рис. 3.24. Підключення шлюзу до MQTT-брокера

Але перед тим як встановлювати з'єднання необхідно визначити IP-адресу та порт на якому запущено сервер (рис. 3.25).

```
bool mqtt_start(void) {
  String _server = jsonRead(configSetup, "mqttServer");
  int _port = jsonReadtoInt(configSetup, "mqttPort");
  mqtt_client.setServer(_server.c_str(), _port);
  return mqtt_connect();
}
```

Рис. 3.25. Ініціалізація MQTT-клієнта

Будь-які дані, опубліковані або отримані MQTT-брокером, будуть закодовані в двійковому форматі, оскільки MQTT є бінарним протоколом. Це означає, що для отримання справжнього вмісту потрібно інтерпретувати повідомлення. Брокери MQTT іноді можуть накопичувати повідомлення, на каналах, де немає поточних підписників. У цьому випадку повідомлення будуть або відкинуті, або збережені, в залежності від інструкцій в керуючому повідомленні. Це корисно в тих випадках, коли новим підписникам можуть знадобитися останні повідомлення.

В якості сервера було обрано MQTT-брокер Mosquitto, який розроблено некомерційною компанією Eclipse Foundation. Він має відкритий код і розповсюджується під ліцензією EPL/EDL. Mosquitto має невелику вагу, і підходить, як для використання на малопотужних одноплатних комп'ютерах так, і на повноцінних серверах.

Для того щоб встановити брокер Mosquitto, спочатку потрібно запустити термінал і додати репозиторій, оновити всі пакети і встановити його. Послідовність команд, які дозволяють це зробити, наведено на рис. 3.26.

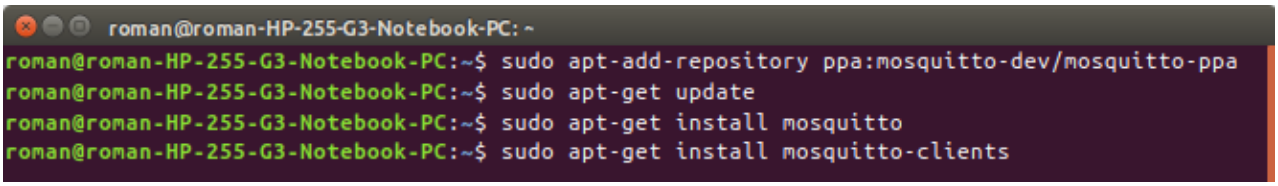

Рис. 3.26. Встановлення брокера Mosquitto

Після того як Mosquitto встановиться, необхідно задати логін та пароль. Для цього потрібно у каталозі /etc/mosquitto/ згенерувати файл mosquitto.pwd, використовуючи команду sudo mosquitto_passwd -c mosquitto.pwd roman, де roman це логін. Тоді потрібно відредагувати поле у конфігураційному файлі, який знаходиться за шляхом /etc/mosquitto/mosquitto.conf. У файлі ключ allow_anonymous встановити у false і нижче вказати шлях до щойно згенерованого файлу - password_file /etc/mosquitto/mosquitto.pwd та зберегти

зміни. Нарешті потрібно перезавантажити брокер Mosquitto і переконатися в успішному його запуску командою service mosquitto status (рис. 3.27).

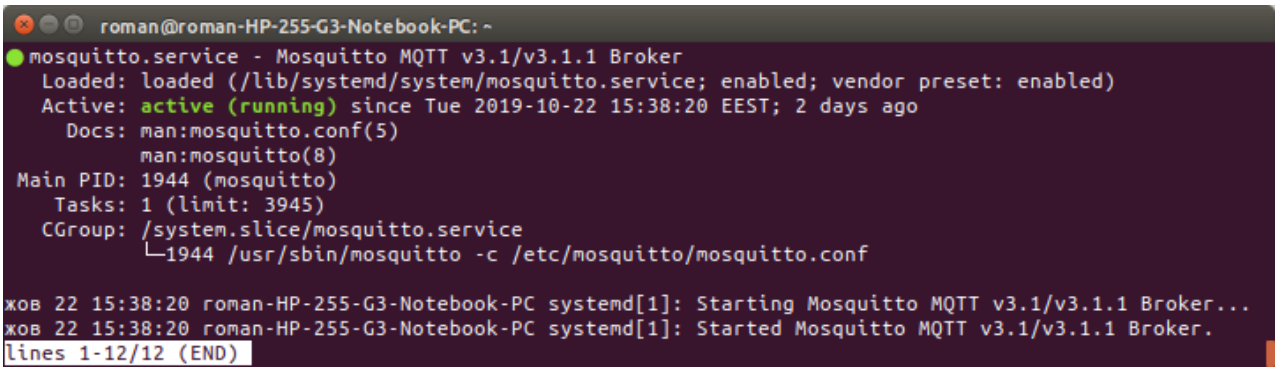

Рис. 3.27. Робота брокера Mosquitto

На завершення потрібно перевірити чи брокер Mosquitto відкрив сокети прослуховування IPv4 і IPv6 на порті 1883. Для цього можна скористатися командою netstat -an | grep 1883, і при необхідності відкрити їх.

3.8. Підключення шлюзу до WMN

Платформа Arduino Nano надсилає шлюзу інформацію про таблицю маршрутизації у вигляді JSON-об’єктів через послідовний інтерфейс. Для реалізації послідовного інтерфейсу на цифрових портах вводу-виводу платформи, можна скористатися бібліотекою SoftwareSerial. За допомогою її програмних засобів можна дублювати функціональність UART. Бібліотека дозволяє програмно створювати кілька послідовних портів, які працюють на швидкості до 115200 бод. Для пристроїв, що працюють з інвертованим сигналом, в бібліотеці передбачено відповідний параметр, що включає інвертування.

UART може використовуватися як для взаємодії компонентів всередині одного пристрою, так і для підключення пристроїв між собою. Для підключення двох пристроїв вихід одного підключають до входу іншого і вхід першого - до

виходу другого [27]. Для коректної роботи UART пристрої повинні мати спільну землю.

Платформу Arduino Nano, яка підключається до шлюзу, потрібно живити напругою 3,3 вольта, оскільки UART-порт мікроконтролера ESP8266 не толерантний до 5 вольт і потребує узгодження логічних рівнів.

Дані між платформою Arduino Nano та шлюзом через UART передаються у вигляді фреймів, кожен з яких складається зі стартового біта, бітів даних і одного або декількох стоп-бітів (рис. 3.28 [27]). Ці додаткові біти потрібні для синхронізації потоку даних. Тобто стартовий і кінцевий біти відділяють один байт інформації, при цьому молодший інформаційний біт передається першим, відразу після стартового. Зустрічаються реалізації UART, які передають по 5, 6, 7, або 9 інформаційних бітів. Відокремлені стартовим і кінцевим біти є мінімальним фреймом. Для синхронізації, кожен приймальний і передавальний UART має джерело опорного тактового сигналу, яке формує часові інтервали [27]. Від його точності залежить стабільність роботи інтерфейсу. З цього випливає, що сума похибок (передавача і приймача) установки тимчасового інтервалу від початку стартового імпульсу до середини кінцевого імпульсу не повинна перевищувати половини бітового інтервалу [27]. Для 8-бітного фрейму 0,5/9,5 = 5% (в реальності не більше 3%).

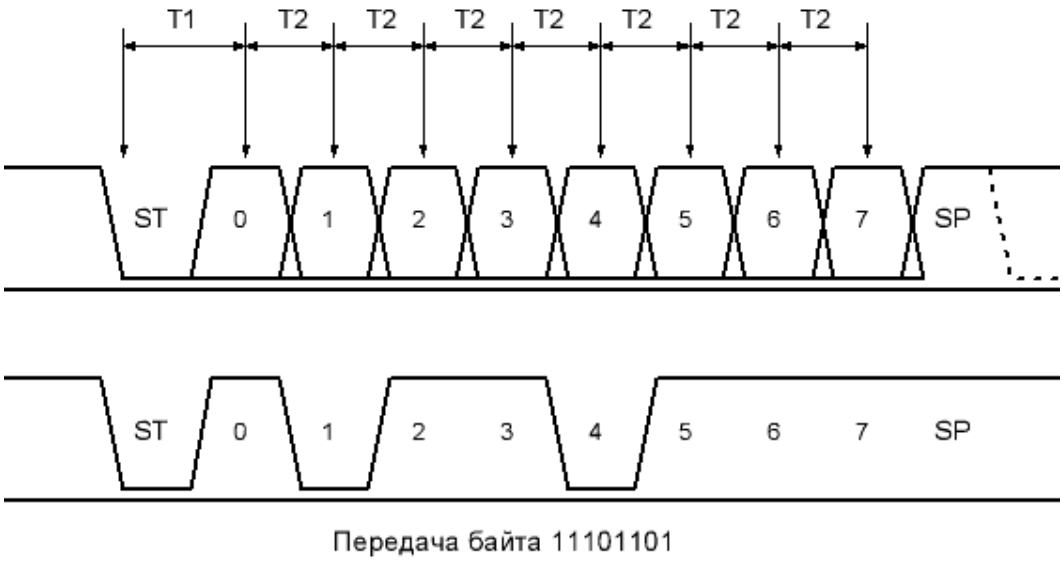

Рис. 3.28. Протокол передачі 8-бітного фрейму UART

Оскільки ця сума помилок приймача і передавача плюс можливі спотворення сигналу в лінії, то рекомендований допуск на точність тактового сигналу UART - не більше 1,5%.

3.9. Візуалізація роботи LoRa Mesh-мережі

Для того, щоб відобразити мережу на веб-сторінці, потрібно отримати всю інформацію про маршрутизацію з кожного вузла. Один з вузлів Mesh-мережі (наприклад вузол 1) можна підключити до Інтернету, використавши вбудований Wi-Fi модуль у контролері ESP8266. Він буде служити в якості шлюзу для цього проекту. Коли вузол 1 приймає таблицю маршрутизації з іншого вузла, він передає дані в JSON форматі через послідовний порт, з 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, як це показано на рис. 3.29

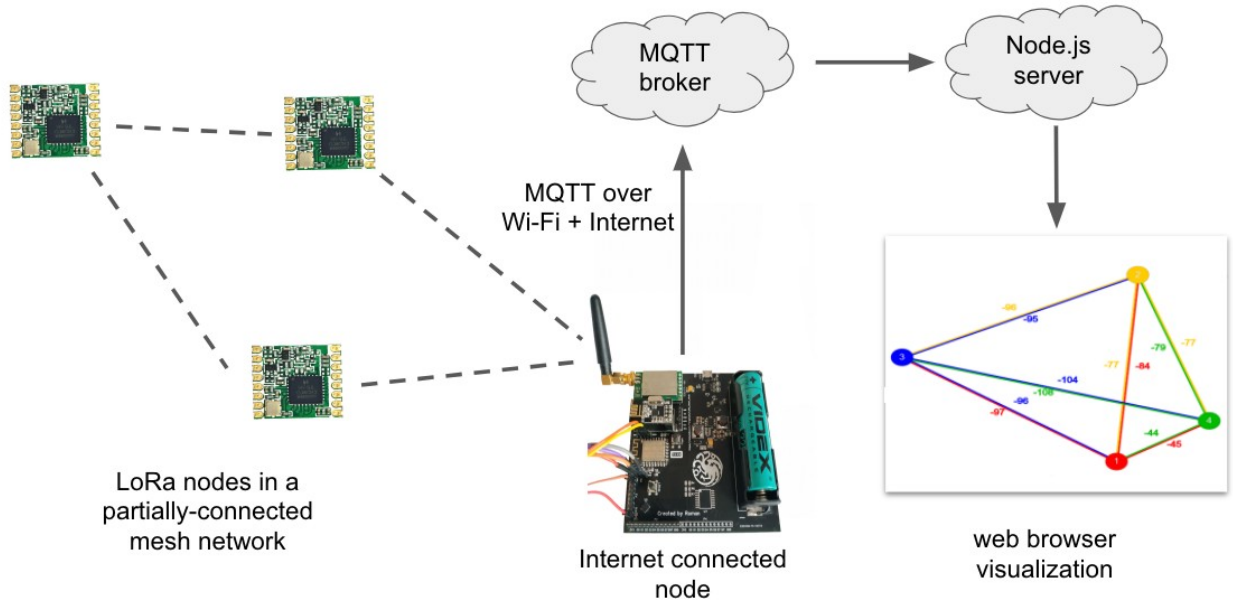

Рис. 3.29. Схема проекту Mesh-мережі

Для того щоб відобразити цю інформації на веб-сторінці потрібно написати Node.js веб-сервер, який підписується на тему MQTT і записує інформацію через websocket до веб-клієнта за допомогою Socket.IO. Для візуалізації створення Mesh-мережі можна використати бібліотеку p5.js.

Для завантаження необхідних залежностей, зручно використовувати менеджер пакунків npm. Він спеціально розроблений для мови програмування JavaScript. За замовчуванням він є менеджером пакунків для середовища виконання Node.js. Його зручно використовувати для глобального встановлення інструментів JavaScript та пакунків, які є локальними залежностями [28] програми візуалізації LoRa Mesh-мережі. Всі залежності знаходяться у файлі проекту package.json Їх можна встановити лише однією командою [29]. Для кожної залежності необхідно вказати діапазон дійсних версій. Використання схеми семантичних версій дозволить уникнути небажаних змін та дозволить автоматично оновлювати пакети [30]. У файлі package.json кожна залежність може визначати діапазон дійсних версій,. На рис. 3.30 наведено список необхідних залежностей та їх версії для коректної роботи сервера.

```
"dependencies": {
    "express": "^4.16.3",
    "jquery": "^3.3.1",
    "mqtt": "^2.18.8",
    "p5": "^0.7.2",
    "rxjs": "^6.3.2",
    "socket.io": "^2.1.1"
  }
```

Рис. 3.30. Залежності, які необхідно встановити

Для встановлення залежностей необхідно відкрити термінал, перейти у каталог з проектом, та виконати команду npm install. Процес завантаження та розпакування необхідних пакунків може зайняти декілька хвилин. Після виконання команди, автоматично створиться каталог node_modules з необхідними залежностями.

Перед запуском програми спочатку потрібно відкрити (будь-який зручним редактором) файл app.js та виконати налаштування сервера (рис. 3.31).

```
var app = express();
var server = require('http').createServer(app);
var mqtt = require('mqtt');
var options = {
  host: 'localhost', // IP адреса MQTT сервера
  port: 1883,        // Порт, на якому він запущений
  username: 'roman', // Логін
  password: 'root'   // Пароль
}
var mqttClient  = mqtt.connect(options)
server.listen(4200); // Порт на якому, буде запущено сервер
```

Рис. 3.31. Налаштування сервера для візуалізації роботи мережі

Після цього потрібно зберегти всі зміни і повторити налаштування на LoRa Mesh-шлюзі. Якщо він у режимі точки доступу, то потрібно під'єднатися до неї і перейти за адресою 192.168.0.1. Якщо ж шлюз у режимі клієнта, то його значок з’явиться у списку мережевих пристроїв, які доступні у локальній мережі. Після подвійного кліку мишкою по ньому з'явиться головна сторінка (див. рис 3.1). Перейшовши на сторінку налаштувань, потрібно аналогічно вказати параметри MQTT-сервера, як це показано на рис. 3.32, зберегти зміни і перезавантажити з’єднання з сервером. Після завершення всіх необхідних

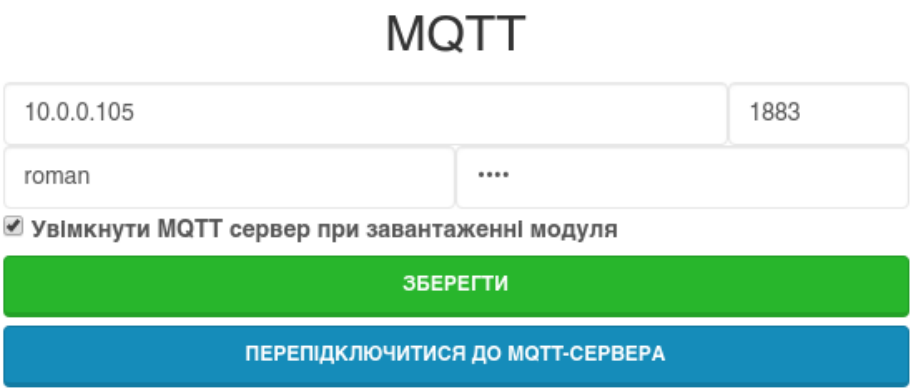

Рис. 3.32. Налаштування шлюзу

маніпуляцій, у підсумку, можна запустити візуалізацію з’єднань LoRa Mesh-мережі командою npm start.

Використовуючи візуалізацію в реальному часі, можна побачити, як формується мережа і як вона переконфігуровується, коли вузли мережі стають недоступними. Суцільна лінія між двома вузлами означає прямий зв'язок. Колір лінії відповідає вузлу, який здійснює маршрутизацію, а число - це рівень сигналу між сусіднім вузлом. З рис. 3.33 видно, що вузли 1 і 3 безпосередньо можуть зв'язуватися. Відстань між вузлами пропорційна силі сигналу - можна бачити, що вузол 4 знаходиться далі від інших вузлів.

Лінії з точками вказують на непрямий зв'язок, наприклад, лінії між вузлами 1 і 4 означають, що вони спілкуються опосередковано. Сині точки на червоній лінії означають, що вузол 1 зв'язується з вузлом 4 через вузол 3 (який є синім). Сині точки на зеленій лінії означають, що вузол 4 здійснює зв'язок з вузлом 1 через вузол 3.

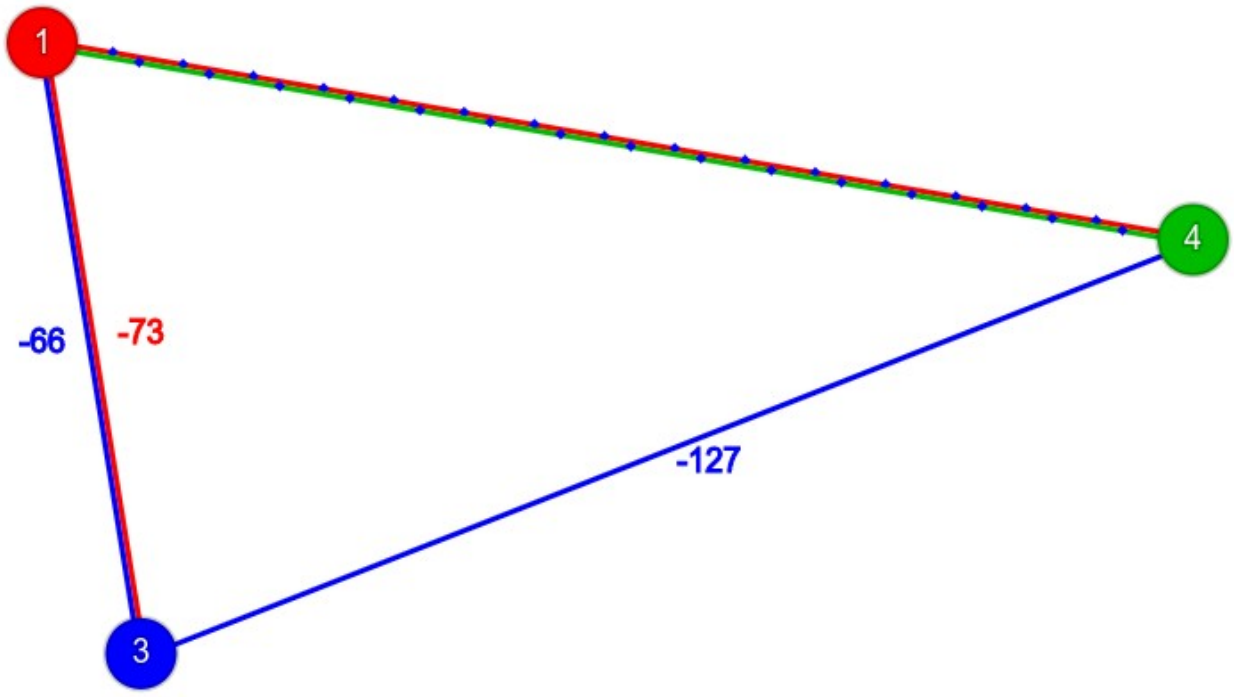

Рис. 3.33. Візуалізація роботи Mesh-мережі

3.10. Приклад віддаленого керування

Однією з важливих особливостей розробленого пристрою є те, що до нього можна підключити блок реле на вісім каналів, оскільки у схемі використовується I2C розширювач портів PCF8574. Для підключення пристрою необхідно під’єднати його кабель живлення до відповідних клем на релейній платі та зафіксувати положення спеціальними штифтами. Модуль може одночасно комутувати до восьми пристроїв зі струмом на одне реле не більше 10 А.

Інтерфейс Mesh-модуля підтримує два способи керування: за допомогою восьми кнопок, але з можливістю вмикати лише одне навантаження, та “керування байтами”, який дозволяє передати будь-яку комбінацію. Також приймачу можна призначити будь-який ідентифікатор, за замовчуванню це “0xFF”. Використовуючи інтерактивний конструктор веб-сторінок шлюзу можна реалізувати свій спосіб взаємодії з мережею (рис. 3.34). Його можна відкрити комбінацією клавіш Ctrl+M.

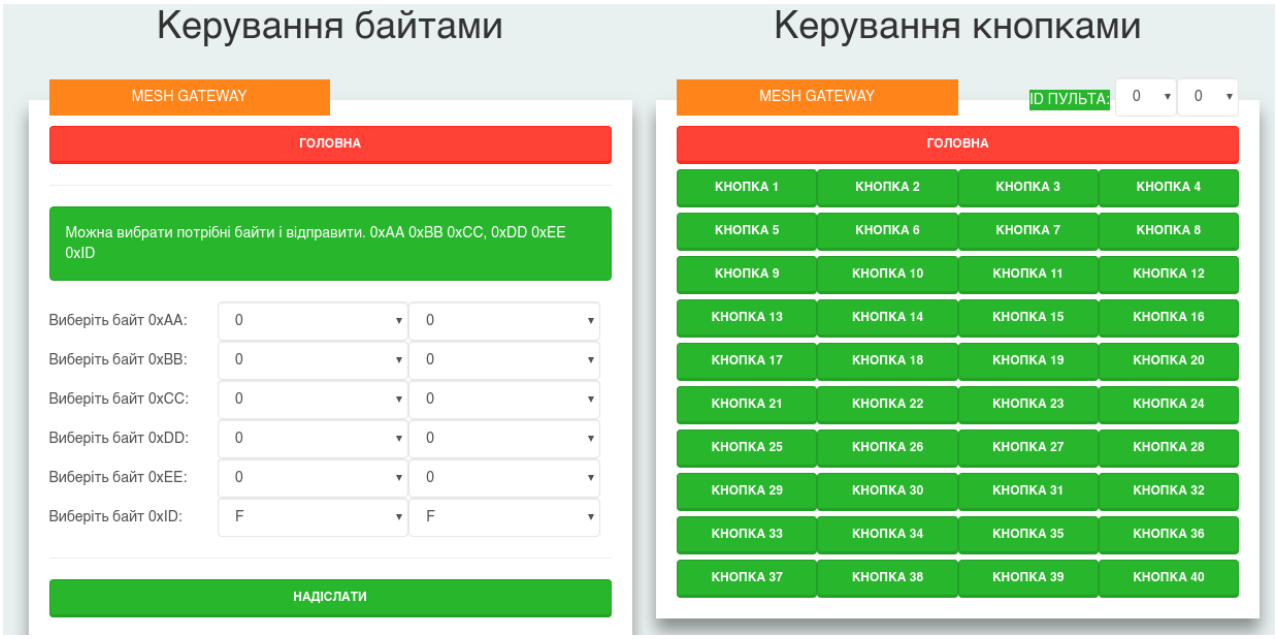

Рис. 3.34. Сторінки віддаленого керування

Вихідний код сторінки зберігається у вигляді JSON-об’єкта. Редагуючи його можна додавати та видаляти необхідні елементи (рис. 3.35).

```
{
  "configs": [
    "/config.live.json",
    "/config.setup.json",
    "/lang/lang.{{lang}}.json"
  ],
  "title": "{{LangButtons}}",
  "class":"col-sm-offset-1 col-sm-10 col-md-offset-2 col-md-8
col-lg-offset-3 col-lg-6",
  "content": []
}
```

Рис. 3.35. Початкова структура сторінки “Керування кнопками”

В об’єкті “configs” вказуються файли в JSON-форматі. Їхня кількість може бути довільною. Вони можуть зберігатися як у файловій системі ESP8266, так і можуть бути завантажені з мережі Інтернет. Для того, щоб вивести дані з цих файлів, потрібно використовувати дві пари фігурних дужок типу {{}}, та вказати в них ім'я об’єкта, який потрібно завантажити, наприклад {{LangButtons}}. У файлі config.setup.json знаходяться початкові налаштування, такі, як назва шлюзу, логіни, паролі, локалізація тощо. Стан датчиків температури, вологості, портів вводу виводу записуються у файл config.live.json. Він створюється в оперативній пам’яті модуля під час завантаження, та існує в ній до його вимкнення. Залежно від параметра {{lang}}, завантажується відповідний файл локалізації.

Об’єктом “class” задаються поточні розміри сторінки. Можна використовувати власні класи або використовувати вже готові класи bootstrap.

У блоці "content": [] знаходиться основне тіло сторінки. Туди вставляються такі елементи як кнопки, прапорці (чекбокси), поля вводу тощо. Слід зазначити, що всі елементи за межами об’єкту “content” не змінюються динамічно, а лише при натиснені кнопки “SAVE” та перезавантажені сторінки.

3.11. Аналіз мікроклімату будинку

До будь якого вузла LoRa Mesh-мережі можна підключити давачі температури, освітленості, атмосферного тиску тощо. Наприклад, давачем DHT11 (DHT22), можна вимірювати температуру і вологість повітря. Для цього в мережевому вузлі потрібно передбачити підтримку конкретного давача. На рис. 3.36 зображено функцію, яка знімає покази температури та вологості повітря і відправляє в JSON-форматі на мережевий шлюз.

```
void sendDataFromDHT(){
  getDataFromDHT(buf, RH_MESH_MAX_MESSAGE_LEN);
  sendAck(1);
  listenIncomingMessages();
```

Рис. 3.36. Відправлення даних з давача на перший мережевий вузол

З коду видно, що функція getDataFromDHT() знімає покази даних і записує в буфер даних (buf). Розмір буфера обмежений і залежить від типу трансивера та його налаштувань. Максимальний розмір для радіомодуля RFM95 становить 256 байт. Функція sendAck() відправляє вміст буфера на конкретний мережевий вузол. Нарешті, функція listenIncomingMessages() перевіряє чи прийшла відповідь про успішне отримання даних шлюзом.

В мережевому шлюзі передбачена можливість будувати графіки від давачів в реальному часі. Для цього в конструкторі сторінки потрібно оголосити об’єкт типу “chart” (рис. 3.37).

```
"type":"chart",
"title": "{{LangTemperatureChart}}",
"state":"getTemperature.json",
"points":10,
"options":"high: 40,low: -10, showArea: true, fullWidth: true",
"style":"height:300px"
```

Рис. 3.37. Приклад створення графіку

В ньому потрібно вказати тип елемента (графік); заголовок графіка (вданому випадку він береться з файлу локалізації lang.ua.json); шлях до файлу в якому знаходяться дані, які потрібно візуалізувати; кількість точок на графіку; додаткові опції такі як верхня і нижня границя поля, тощо; стиль у CSS форматі.

Після завантаження сторінки з графіком, клієнт буде запитувати дані температури у вигляді GET запитів, при цьому в сервері потрібно передбачити відповідь на них. Аналогічним способом можна виводити дані вологості повітря, як це показано на рис. 3.38.

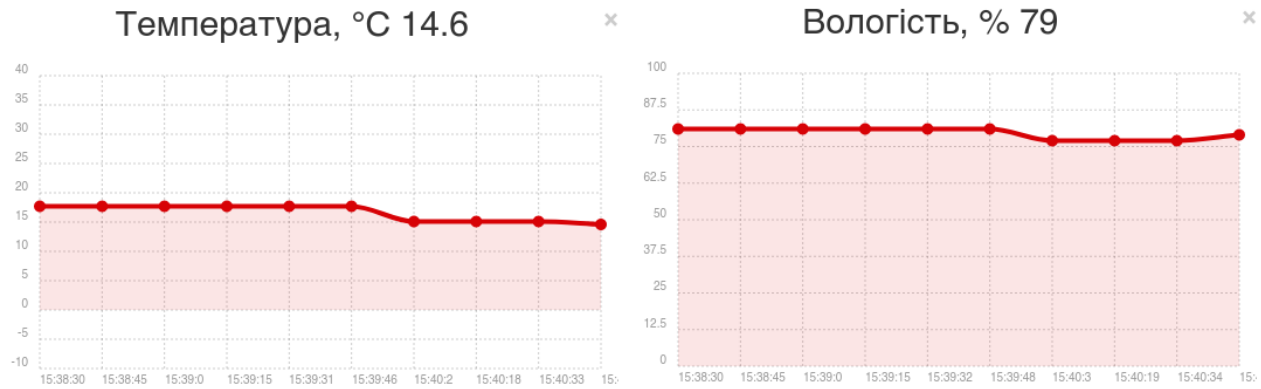

Рис. 3.38. Результати заміру давачем DHT11

3.12. Висновки до розділу

У даному розділі проаналізовано пакет бібліотек, які дозволяють реалізувати безпровідну Mesh-мережу на платформі Arduino Nano та з використанням радіомодуля RFM95. Реалізовано програмний код для мережевих вузлів та шлюзу запропонованої системи контролю розумним будинком. Проведено розробку “front end” та “back end” частин сервера.

Розроблено макет системи контролю розумним будинком з використанням обраної компонентної бази. Для аналізу мікроклімату будинком до одного з кінцевих вузлів під’єднано давач DHT11 та проведено експеримент по збору телеметрії при симуляції відмови декількох мережевих вузлів.

РОЗДІЛ 4
ОБҐРУНТУВАННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ

Економічне обґрунтування дипломної роботи магістра є метою даного розділу. Даний розділ дозволяє встановити доцільність проведення науково–дослідних робіт і економічно обґрунтувати доцільність застосування тих чи інших засобів. Саме проведення економічних розрахунків, спрямованих на визначення економічної ефективності науково–дослідницької роботи (НДР) і прийняття рішення про її подальший розвиток та впровадження або ж недоцільність проведення відповідної розробки.

Метою дипломної роботи магістра є розробка апаратно-програмної системи контролю розумним будинком з використанням LoRa Mesh-технологій.

В економічній частині дипломного проекту будуть проведені такі етапи розрахунку вартості НДР:

- описати технологічний процес розробки із зазначенням трудомісткості кожної операції;
- визначити суму витрат на оплату праці основного і допоміжного персоналу, включаючи відрахування на соціальні заходи;
- визначити суму матеріальних затрат;
- обчислити витрати на електроенергію для науково– виробничих цілей;
- нарахувати суму амортизаційних відрахувань;
- визначити суму накладних витрат;
- скласти кошторис та визначити собівартість НДР;
- розрахувати ціну НДР;
- визначити економічну ефективність та термін окупності продукту.

На основі отриманих розрахунків будуть розроблені техніко-економічні показники проектованого виробництва.

Як відомо, розробка надійної і ефективної інформаційної системи вимагає значних затрат часу. Слід зауважити, що затрати часу залежать від кваліфікації розробника і його можливостей. Розробник повинен у достатній мірі володіти

навиками програмування, вміти адекватно застосовувати математичний апарат, бути добре обізнаним з об'єктом дослідження.

4.1. Визначення стадій технологічного процесу та загальної тривалості проведення НДР

Для оцінки тривалості виконання окремих робіт використовують нормативи часу або попередній досвід. До таких нормативів відносять тривалість написання операцій (команд), які в деяких підприємствах становлять: для одної операції від 30 хвилин до 1,6 годин та 8 годин для п'яти операцій (тривалість зміни).

У разі їх відсутності звертаються до експертних оцінок по встановленню тривалості кожного етапу (стадії):

при трьох оцінках:

$$T_{вс} = \frac{(t_{min} + 4t_{н.й} + t_{max})}{6}, \quad (4.1)$$

при двох оцінках:

$$T_{вс} = \frac{(3t_{min} + 2t_{max})}{5}, \quad (4.2)$$

де $T_{вс}$ – очікуване (середнє) значення тривалості виконання етапу (стадії),

t_{min}, $t_{н.й.}$, t_{max}– відповідно мінімальна, найбільш імовірна і максимальна оцінки тривалості виконання етапу (стадії).

Розробку даної інформаційної системи можна поділити на такі етапи:

- постановка задачі;

- проведення огляд публікацій авторів, які займались питанням систем контролю розумним будинком і наступне їх опрацювання;

- прийняття рішень щодо вибору оптимального шляху розв'язання поставленої задачі;

- аналіз математичної моделі інформаційної системи;

- обґрунтування методів використання LoRa Mesh-технологій в системі контролю розумним будинком;

- розробка архітектури та алгоритмічного забезпечення системи контролю розумним будинком з використанням запропонованих методів та моделі;

- розробка макету системи контролю розумним будинком;

- написання програмного забезпечення для контролера та WI-FI модуля для запропонованої системи;

- налаштування середовища розробки і роботи вже готової програми;

- тестування та оцінка завадостійкості модуля безпровідного зв’язку;

- написання і оформлення документації (електронної та паперової).

Для зручного представлення і визначення загальної тривалості проведення НДР доцільно дані витрат часу по окремих операціях технологічного процесу звести у табл. 4.1.

Витрати часу наукового керівника на виконання окремих стадій (етапів) при недостатній кількості інформації доцільно приймати в межах 5% сумарних витрат часу інженерів на виконання цих стадій (етапів).

Таблиця 4.1

Основні етапи і час їх виконання у НДР

№ п/п	Етап	Середній час виконання етапу, год	
		інженер	керівник
1	Постановка задачі	2	1
2	Проведення огляд публікацій авторів, які займались питанням систем контролю розумним будинком і наступне їх опрацювання	20	8
3	Прийняття рішень щодо вибору оптимального шляху розв'язання поставленої задачі	5	2

Продовж. табл. 4.1

№ п/п	Етап	Середній час виконання етапу, год	
		інженер	керівник
4	Аналіз математичної моделі інформаційної системи	1	1
5	Обґрунтування методів використання LoRa Mesh-технологій в системі контролю розумним будинком	7	2
6	Розробка архітектури та алгоритмічного забезпечення системи контролю розумним будинком з використанням запропонованих методів та моделі	45	5
7	Розробка макету системи контролю розумним будинком	6	1
8	Написання програмного забезпечення для контролера та WI-FI модуля для запропонованої системи	24	1
9	Налаштування середовища розробки і роботи вже готової програми	1	1
10	Тестування та оцінка завадостійкості модуля безпровідного зв’язку	4	1
11	Написання і оформлення документації (електронної та паперової)	83	2
Разом		198	25

Отже, сумарний час виконання операцій технологічного процесу інженером становить 198 годин, а керівником 25 годин [31].

4.2. Визначення витрат на оплату праці та відрахувань на соціальні заходи

Заробітна плата працівника незалежно від виду підприємства визначається його особистим трудовим вкладом, залежить від кінцевих результатів роботи підприємства, регулюється податками і максимальними розмірами не обмежується. Розміри, порядок нарахування і виплати заробітної плати регулюються чинним законодавством України, відповідними указами і

постановами, галузевими інструкціями. Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства. Заробітна плата складається з основної та додаткової оплати праці.

Основна заробітна плата нараховується на виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами і не залежить від результатів господарської діяльності підприємства.

Додаткова заробітна плата – це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов'язані з виплатами за фактично відпрацьований час. Нараховують додаткову заробітну плату залежно від досягнутих і запланованих показників, умов виробництва, кваліфікації виконавців. Джерелом додаткової оплати праці є фонд матеріального стимулювання, який створюється за рахунок прибутку.

Основна заробітна плата складається із прямої заробітної плати та доплати, яка при укрупнених розрахунках становить 25% – 35% від прямої заробітної плати. При розрахунку заробітної плати кількість робочих днів в місяці слід приймати – 25,4 дні/міс., що відповідає 203,2 год./міс. Розмір місячних окладів керівника та інженерів слід приймати згідно існуючих на даний час норм. Основна заробітна плата розраховується за формулою:

$$З_{осн} = Т_С \cdot К_Г, \qquad (4.3)$$

де $Т_С$ – тарифна ставка, грн.;

$К_Г$ - кількість відпрацьованих годин.

Посадові оклади (тарифні ставки) за розрядами Єдиної тарифної сітки визначаються шляхом множення окладу (ставки) працівника 1 тарифного розряду на відповідний тарифний коефіцієнт. У разі коли посадовий оклад (тарифна ставка) визначені у гривнях з копійками, цифри до 0,5 відкидаються, від 0,5 і вище – заокруглюються до однієї гривні.

Законом України “Про Державний бюджет України на 2019 рік” від 23.11.2018 р. №2629 – VIII із змінами, внесеними згідно із Законом № 2696-VIII від 28.02.2019, ВВР, 2019, № 14, ст.66 та № 149-IX від 02.10.2019, установлено у 2019 році мінімальну заробітну плату: у місячному розмірі: з 1 січня - 4173 гривні; у погодинному розмірі: з 1 січня - 25,13 гривні. Приймемо 65 грн. для інженера, для керівника — 81 грн.

Тарифні ставки: керівник проекту – 81 грн./год., інженер – 65 грн./год.

Тоді скориставшись формулою 4.3 розрахуємо основну заробітну плату для інженера та керівника проекту.

Керівник проекту:

$$З_{осн} = 81 \cdot 25 = 2025 \text{ грн.}$$

Інженер:

$$З_{осн} = 65 \cdot 198 = 12870 \text{ грн.}$$

Додаткова заробітна плата становить 10–15% від суми основної заробітної плати:

$$З_{дод} = З_{осн} \cdot К_{допл}, \quad (4.4)$$

де $К_{допл}$ – коефіцієнт додаткових виплат працівникам 0,1.

Керівник проекту:

$$З_{осн} = 2025 \cdot 0,1 = 202,5 \text{ грн.}$$

Інженер:

$$З_{осн} = 12870 \cdot 0,1 = 1287 \text{ грн.}$$

Звідси загальні витрати на оплату праці ($В_{О.П.}$) визначаються за формулою, і становлять:

$$В_{оп} = З_{осн} + З_{дод}, \quad (4.5)$$

Керівник проекту:

$$В_{о.п.} = 2025 + 202,5 = 2227,5 \text{ грн.}$$

Інженер:

$$B_{о.п.} = 12870 + 1287 = 14157 \text{ грн.}$$

Таким чином загальна сума становить 16384,5 грн. Крім того, слід визначити відрахування на соціальні заходи:

- фонд страхування від безробіття – 2,1%;

- пенсійний фонд – 32%;

- фонд соціального страхування – 2,9%;

- фонд соціального страхування від нещасних випадків і професійних захворювань – 1%.

У сумі зазначені відрахування становлять 38 %. Отже, сума відрахувань на соціальні заходи розраховуємо за формулою:

$$B_{С.З.} = ФОП \cdot 0,38, \quad (4.6)$$

де $ФОП$ – фонд оплати праці в гривнях.

Тоді, сума відрахувань на соціальні заходи буде становити:

$$B_{С.З.} = 16384,5 \cdot 0,38 = 6226,11 \text{ грн.}$$

Проведені розрахунки витрат на оплату праці зведемо у табл. 4.2.

Таблиця 4.2

Зведені розрахунки витрат на оплату праці

№ п/п	Категорія праців-ників	Основна заробітна плата, грн.			Додат-кова заробі-тна плата, грн.	Нарах. на ФОП, грн.	Всього витрати на оплату праці, грн. 6=3+4+5
		Тариф на ставка, грн.	К-сть від-працьов. год.	Фактично нарах. з/пл., грн.			
1	2	3	4	5	6	7	8
А	Б	1	2	3	4	5	6
1	Керівник проекту	81	25	2025	202,5	846,45	3073,95
2	Інженер	65	198	12870	1287	5379.66	19536,66
Разом				14895	1489,5	6226,11	22610,61

4.3. Розрахунок витрат на електроенергію

Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$З_E = W \cdot T \cdot S, \quad (4.7)$$

де W – необхідна потужність, кВт; T– кількість годин роботи обладнання; S – вартість кіловат-години електроенергії.

Згідно з постановою НКРЕ України від 05.10.2018 року № 1177 вартість електроенергії становить 308,25 коп./кВт·год.

Потужність ноутбука – 27Вт з підключеним маршрутизатором і комутатором, кількість годин роботи обладнання згідно таблиці 4.1 – 223 год.

$$З_E = 0,027 \cdot 223 \cdot 3,0825 = 18,56 \text{ грн.}$$

4.4. Розрахунок витрат на матеріали

Результати розрахунку затрат на матеріали зводяться в табл. 4.3.

Таблиця 4.3

Визначення величини затрат на матеріали

Найменування матеріальних ресурсів	Одиниця виміру	Норма витрат	Ціна за одиницю грн	Затрати матеріалів грн	Транспортно-заготівельні витрати, грн	Загальна сума витрат на матеріали, грн
Папір А4 ZOOM	Пачка	1	82	82	-	82
Ватман	Штук	10	10	100	-	100
Arduino Nano	Штук	3	85	20	-	20
Модуль RFM95	Штук	4	114,4	457,6	-	457,6
Модуль ESP8266	Штук	1	54,1	54,1		54,1
Плата шлюзу	Штук	1	9,95	9,95	248,85	258,8
Провідники	Пачка	2	14,25	28,5	-	28,5
Разом						1001

4.5. Розрахунок суми амортизаційних відрахувань

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення. Для заміщення зношеної частини основних засобів виробництва підприємства роблять амортизаційні відрахування, тобто відрахування певних грошових сум відповідно до розмірів фізичного і морального зносу засобів виробництва.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Для визначення амортизаційних відрахувань застосовуємо формулу:

$$A = \frac{Б_В \cdot Н_А}{100}, \qquad (4.8)$$

де A – амортизаційні відрахування за звітний період, грн.

$Б_В$ – балансова вартість комп'ютера, на початок звітного періоду, грн.

$Н_А$ – норма амортизації, %.

Для роботи використовується один ноутбук (вартість якого становить 7200 грн.), який працює 223 години.

$$A = \frac{7200 \cdot 15\%}{100\%} = 1080\,грн.$$

4.6. Обчислення накладних витрат

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

Накладні витрати можуть становити 20% від суми основної та додаткової заробітної плати працівників:

$$H_B = B_{O.П.} \cdot 0{,}2, \tag{4.9}$$

$$H_B = 16384{,}5 \cdot 0{,}2 = 3276{,}2\,грн.$$

4.7. Складання кошторису витрат та визначення собівартості НДР

Результати проведених вище розрахунків зведемо у табл. 4.4. Собівартість (C_B) НДР розрахуємо за формулою:

$$C_B = B_{O.П.} + B_{С.З.} + З_{М.В.} + З_E + T_B + A + H_B, \tag{4.10}$$

$$C_B = 6384{,}5 + 6226{,}11 + 1001 + 18{,}56 + 1080 + 3276{,}2 = 27986{,}37\,грн.$$

Таблиця 4.4

Кошторис витрат на НДР

Зміст витрат	Сума, грн.	У % до загальної суми
1	2	3
Витрати на оплату праці (основну і додаткову заробітну плату)	16384,5	58,54
Відрахування на соціальні заходи	6226,11	22,25
Матеріальні витрати	1001	3,58
Витрати на електроенергію	18,56	0,07
Амортизаційні відрахування	1080	3,86
Накладні витрати	3276,2	11,7
Собівартість	27986,37	—

4.8. Розрахунок ціни НДР

Ціну НДР можна визначити за формулою:

$$Ц = \frac{C_В \cdot (1 + P_{рен}) + K \cdot B_{Н.І.}}{K}, \quad (4.11)$$

де $P_{рен}$ – рівень рентабельності, 30 %;

K – кількість замовлень;

$B_{н.і.}$ – вартість носія інформації, грн.

$$Ц = \frac{27986,37 \cdot (1 + 0,3) + 1 \cdot 7}{1} = 36389,28 \text{ грн}.$$

Таким чином ціна рівна 36389,28 грн.

Визначимо величину прибутку за формулою:

$$П = Ц - C_В, \quad (4.12)$$

$$П = 36389,28 - 27986,37 = 8402,91 \text{ грн}.$$

Згідно даної формули отримаємо 8402,91 грн.

4.9. Визначення економічної ефективності і терміну окупності

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу. Економічна ефективність ($E_р$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_Р = \frac{П}{C_В}, \quad (4.13)$$

де $П$ – прибуток;

C_B – собівартість.

$$E_P = \frac{8402,91}{27986,37} = 0,3.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (T_P):

$$T_P = \frac{1}{E_P}, \tag{4.13}$$

$$T_P = \frac{1}{0,3} = 3,3 роки.$$

Про доцільність розробки програми можна сказати при врахуванні критеріїв, які наведено у табл. 4.5

Таблиця 4.5

Техніко-економічні показники НДР

№ п/п	Показник	Значення
1	Собівартість, грн.	27986,37
2	Плановий прибуток, грн.	8402,91
3	Ціна, грн.	36389,28
4	Економічна ефективність	0,3
5	Термін окупності, рік	3,3

У результаті проведення розрахунків можна зробити висновок: розробка матиме оптимальну економічну ефективність 0,3 і термін окупності становитиме більше трьох років (3,3 роки). Варто зазначити, що дані розрахунки носять номінальний характер і основна їх мета оцінити приблизну вартість дослідження та створення даного продукту. Номінальний характер розрахунків зумовлений тим, що даний програмний продукт має дослідницьке призначення.

РОЗДІЛ 5
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1. Охорона праці

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [32].

В даній роботі здійснюється розробка системи контролю розумним будинком з використанням LoRa Mesh - технологій, яка супроводжується виконанням ряду робіт з використанням ЕОМ. Тому необхідним є дотримання правил охорони праці, техніки безпеки та протипожежної безпеки при роботі з комп’ютеризованою технікою.

При роботі з комп'ютером працівник піддається дії ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, НВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики, тощо.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

До діючих нормативних документів, що забезпечують охорону праці користувачів ЕОМ належать:

— НПАОП 0.00-7.15-18 “Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями” [33].

— ДСанПіН 3.3.2.007-98 “Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин” [34].

— Закон України “Про охорону праці” [32].

В законі України “Про охорону праці” вказано, що охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Заходи з охорони праці користувачів ЕОМ розглядають в трьох аспектах: соціальному, психологічному та медичному. У соціальному плані розв'язання цих проблем пов'язане з оптимізацією умов життя, праці, відпочинку, харчування, побуту, розвитком культури, транспорту. Значне місце у профілактиці розладів здоров'я належить психології праці. Тому заходи, пов'язані з формуванням раціональних виробничих колективів, у яких відсутня психологічна несумісність, сприяють зменшенню нервово-психічного перенапруження, підвищенню працездатності та ефективності праці.

Нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин повинні відповідати вимогам СН 4088-86, СН 2152-80, ГОСТ 12.1.005-88, ДСТУ ГОСТ 12.0.230:2008.

Мікрокліматичні умови приміщення, де проводиться дослідження програмного забезпечення для системи контролю розумним будинком з використанням LoRa Mesh - технологій, повинні відповідати нормальним значенням таких показників:

— температура повітря 23-25 С°;

— відносна вологість повітря 40-60%;

— швидкість руху повітря не більш 0,1 м/сек;

— інтенсивність теплового (інфрачервоного) опромінення не повинна перевищувати 35,0 Вт/м^2.

Найповнішим нормативним документом щодо забезпечення охорони праці користувачів персонального комп'ютера є “Державні санітарні правила і норми роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин” ДСанПіН 3.3.2.007-98.

У процесі роботи з комп'ютером необхідно дотримувати правильний режим праці та відпочинку. В іншому випадку у персоналу наголошуються значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, втому і хворобливі відчуття в очах, в попереку, в області шиї і руках.

Штучне освітлення в приміщеннях з робочим місцем, обладнаним візуальними дисплейними терміналами (ВДТ), має здійснюватися системою загального рівномірного освітлення. Як джерело штучного освітлення мають застосовуватись люмінесцентні або світлодіодні лампи.

На особливу увагу заслуговують заходи дотримання протипожежної безпеки в приміщеннях, у яких знаходяться ЕОМ. Так, у всьому приміщені лінії електромережі повинні бути забезпечені від виникнення короткого замикання, а також від перепадів напруги, що може викликати збоїв в роботі електронно–обчислювальної техніки. Приміщення повинні бути оснащені системою автоматичної пожежної сигналізації та вогнегасниками. Під час експлуатації ліній електромережі необхідно повністю виключити можливість виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, не допускати застосування проводів з легкозаймистою ізоляцією і, за можливості, застосовувати негорючу ізоляцію.

Отже, в результаті роботи над системи контролю розумним будинком з використанням LoRa Mesh - технологій було проведено аналіз норм праці, шкідливих та небезпечних чинників, з якими стикається розробник системи, а також описано параметри і характеристики приміщення, заходи, які необхідно виконати для забезпечення належних умов роботи.

5.2. Заходи по підвищенню стійкості об’єктів, оснащених системою контролю розумним будинком, в надзвичайних ситуаціях (НС). Захист персоналу об’єктів та членів їх сімей

Під стійкістю роботи об’єкта промисловості, оснащеним системою контролю розумним будинком, розуміють його здатність в умовах надзвичайних ситуацій випускати продукцію в запланованому об’ємі, а при отриманні слабких і середніх руйнувань, при пожежах, повенях, зараженні місцевості, а також, при порушенні зв’язків по кооперації і постачанню відновлювати виробництво в мінімальні терміни.

На стійкість роботи об'єкта в умовах надзвичайних ситуацій впливають наступні фактори:

— надійність захисту робітників та службовців;

— спроможність інженерно-технічного комплексу об'єкта протистояти у визначеному ступеню вражаючих чинників стихійного лиха, аварій, катастроф та сучасних видів зброї;

— захищеність об'єкта від вторинних вражаючих чинників (пожеж, вибухів, зараження ОР та СДОР);

— надійність системи забезпечення об'єкта всім необхідним для виробництва (сировиною/паливом, комплектуючими вузлами і деталями, електроенергією, водою, газом, тощо);

— стійкість та безперервність управління виробництвом та заходами цивільної оборони;

— підготовленість об'єкта до ведення рятувальних та інших невідкладних робіт до поновлення виробництва.

Перелічені фактори є основними і загальними для усіх об'єктів. Шляхами підвищення стійкості роботи в умовах надзвичайних ситуацій є:

— забезпечення надійного захисту робітників та службовців від вражаючих чинників сучасної зброї, аварій, катастроф і стихійного лиха;

— захист основних виробничих фондів від вражаючих чинників, у тому числі і від вторинних, які виникають внаслідок надзвичайних ситуацій;

— підготовка до відновлення порушеного виробництва;

— підвищення надійності та оперативності управління виробництвом та заходами ЦЗ.

Таким чином, підвищення стійкості роботи об’єктів промисловості в умовах надзвичайних ситуацій досягається завчасним проведенням комплексу інженерно- технічних, технологічних і організаційних заходів, спрямованих на максимальне зниження впливу вражаючих чинників і створення умов для ліквідації наслідків надзвичайних ситуацій.

Інженерно-технічні заходи включають комплекс робіт, направлених на підвищення стійкості виробничих будівель, споруд, технологічного обладнання, комунально-енергетичних систем [35].

Технологічні заходи забезпечують підвищення стійкості об’єкта шляхом спрощення технологічного процесу виробництва кінцевої продукції та виключення або обмеження розвитку аварій.

Організаційні заходи передбачають розробку і планування дій керівного складу, штабу, служб і формувань ЦЗ по захисту робітників і службовців, проведенню рятувальних і невідкладних робіт, відновленню виробництва, а також випуску продукції на обладнанні, що збереглося.

Відповідно до вимог розділу IV Кодексу цивільного захисту України від 2 жовтня 2012 року № 5409-VI, захист робітників і службовців/населення досягається трьома основними способами:

— застосуванням засобів індивідуального захисту,

— укриттям людей в захисних спорудах;

— проведенням евакуаційних заходів для робітників і службовців та членів їх сімей.

Засоби індивідуального захисту забезпечують захист людей при знаходженні на виробничих місцях і на місцевості, яка заражена РР, ОР, НХР і БЗ. Укриття в захисних спорудах — найбільш ефективний спосіб захисту

виробничого персоналу працюючої зміни. Суть способу полягає у своєчасному укритті людей в спеціальних інженерних спорудах, які здатні захистити людей від дій вражаючих факторів або послабити їх дії.

Наявний фонд захисних споруд в повсякденних умовах життєдіяльності використовується для господарських, культурних і побутових потреб у порядку, який забезпечує використання їх за прямим призначенням в установлений короткий термін.

Евакуаційні заходи забезпечують захист членів сімей робітників, службовців і виробничого персоналу непрацюючих змін. Як спосіб захисту, полягає в завчасному (до початку виникнення НС, в період загрози) вивезенні (виведенні населення із місць можливого ураження, зони катастрофічного затоплення (зараження) в безпечні райони на тимчасове або постійне проживання. Евакуація проводиться на державному, регіональному, місцевому або об’єктовому рівні.

Залежно від особливостей НС встановлено такі види евакуації: обов’язкова, загальна або часткова, тимчасова або безповоротна [36].

Населення, що підлягає евакуації поділяється на дві категорії. До першої категорії відносяться робітники і службовій підприємств і установ, що будуть працювати під час війни, продукція яких потрібна для оборони, робітники комунальних підприємств міста. Захист робітників працюючої зміни забезпечується в сховищах на об’єктах. Захист членів сімей і робітників непрацюючої зміни забезпечується в заміській зоні. Для цієї категорії населення евакуаційні заходи називають розосередженням робітників і службовців, що діють за принципом: жити за межами міста, працювати в місті. Тому для них райони розміщення призначаються ближче до міста, рядом з транспортними магістралями з урахуванням того, щоб час на проїзд до роботи і назад в заміську зону не перевищував 4-5 годин.

Отже, впровадження системи контролю розумним будинком в об’єкти промисловості дозволить підвищити їх стійкість в умовах надзвичайної ситуації та забезпечить вищий рівень захисту персоналу об’єктів та членів його сімей.

5.3. Оцінка дії електромагнітного імпульсу (EMI) на елементи системи контролю розумним будинком і методи захисту

У разі застосування ядерної зброї проти України, одним з ключових уражаючих факторів, що вплине на функціонування системи контролю розумним будинком, є EMI ядерного вибуху.

Електромагнітний імпульс (EMI) - це короткий сплеск електромагнітної енергії великої напруженості та широкого спектру частот, який створюється та поширюється протягом дуже короткого часу.

Походження EMI може бути природним або техногенним явищем. Він виникає під час потужного вибуху (переважно атомної бомби), явищ, що викликають раптові збурення магнітного поля Землі, грозових явищ у земній атмосфері чи короткого замикання в електрообладнанні високої потужності. EMI, який виникає під час ядерного вибуху більшість своєї енергії переносить в електромагнітних хвилях з частотою в діапазоні від 3 Гц до 30 кГц за напруженості магнітного поля, що досягає 50 кВ/м. EMI індукує високу електричну напругу в електромережах, електричному і електронному обладнанні. Зростання напруженості спричиняє раптове зростання електричної напруги і виділення великої кількості тепла, внаслідок чого зазнають пошкоджень електронні елементи, електричні кола і навіть лінії електропередачі. Високі напруги також можуть призвести до пробою електричної ізоляції, а при електромагнітному імпульсі, який має більш високих рівень енергії (удар блискавки), може пошкодити фізичні об'єкти, такі як будівлі та конструкції літака.

EMI наднизької потужності такі як імпульсні хвилі, викликають незначний рівень електричного шуму або завад, які можуть впливати на роботу чутливого обладнання. Наприклад, радіомодулям LoRa, які є компонентами системи розумного будинку, може знадобитися більше часу для передачі пакету даних між вузлами мережі.

ЕМІ низької потужності можуть викликати невеликі статичні розряди. Відомо що такий розряд, може вивести з ладу окремі елементи системи контролю розумним будинком на незначний проміжок часу.

ЕМІ середньої потужності можуть безпосередньо впливати на магнітні матеріали та пошкоджувати дані, що зберігаються на носіях інформації, таких як магнітні стрічки та жорсткі диски комп'ютера.

ЕМІ високої потужності такі як удар блискавки, здатні завдати значної шкоди системі контролю розумним будинком, пошкодивши радіомодулі або шини обміном даних мережевого шлюзу.

ЕМІ надвисокої потужності можуть бути спричинені ядерною та електромагнітною зброєю. ЕМІ, що виникає під час вибуху ядерної зброї є настільки потужним, що розглядається як один з факторів ураження від цієї зброї. При наземному і низькому повітряному вибухах вплив ураження від ЕМІ спостерігається на відстані до декількох десятків кілометрів від епіцентру вибуху. Дія ураження від ЕМІ проявляється, насамперед, стосовно до радіоелектронної та електротехнічної апаратури, в тому числі на систему контролю розумним будинком та інші об'єктів, що використовують електричну енергію. Зокрема, струми і напруги, які виникають у функціональних вузлах розробленої системи, внаслідок дії ЕМІ такої потужності, можуть викликати пробій ізоляції, пошкодження трансформаторів та напівпровідників, псування енергонезалежної та Flash пам’яті, перегорання плавких вставок та інших елементів обладнання.

Незважаючи на природу ЕМІ вживаються відповідні заходи для мінімізації їх впливу на елементи системи контролю розумним будинком. Більшість заходів контролю зосереджені на перевірці обладнання до стійкості впливу ЕМІ та захисту його від їх шкоди. Антропогенні джерела електромагнітних імпульсів, окрім зброї, підлягають ряду контрольних заходів з метою зменшення кількості випромінюваної імпульсної енергії.

ЕМІ високої та надвисокої потужностей можуть становити загрозу безпеці персоналу, що експлуатує систему контролю розумним будинком. Тому

під час дії ЕМІ слід уникати прямого контакту з елементами пристрою, які проводять електричний струм. Висока напруженість електричного поля може спричинити пробій повітря та виникнення електричної дуги. Проте напруженість електричного поля до 200 кВ/м вважається безпечною.

Для перевірки дії ЕМІ на систему контролю розумним будинком можна використовувати симуляції впливу ЕМІ:

— Індуковане імпульсне моделювання. Індуковані імпульси наносять набагато менше шкоди ніж ЕМІ середньої та високої потужності, але їх простіше створити та їх вплив на обладнання складніше передбачити. Методика випробування полягає у використані спеціального індукційного пристрою, що складаються з трансформатора. В нього поміщають кабель, який іде напряму до обладнання, що перевіряється. Це дозволяє за рахунок явища електромагнітної індукції наводити синусоїдальні сигнали, що затухає, в кабелі та оцінювати їх вплив на обладнання.

— Моделювання імпульсів загрози. Іноді сам ЕМІ моделюється повторюваним способом. Імпульси можуть відтворюватися при низькій енергії, щоб оцінити реакцію обладнання перед його затуханням або при високій енергії для відтворення фактичних умов загрози. Імітатори ЕМІ можуть бути різних розмірів, в залежності від типу і рівня загрози, яку потрібно змоделювати. У багатьох країнах побудовано високоенергетичні імітатори ЕМІ, які здатні протестувати цілі транспортні засоби, включаючи кораблі та літаки. Майже всі імітатори ЕМІ використовують спеціалізовану версію генератора Маркса.

Отже, для оцінки дії ЕМІ на систему контролю розумним будинком, можна використовувати симуляції впливу ЕМІ для визначення її надійності, а рівень знань про природу і властивості ЕМІ дозволить підвищити стійкість системи загалом.

РОЗДІЛ 6
ЕКОЛОГІЯ

6.1. Ефективність застосування енергозберігаючих технологій

Енергоефективність – це галузь знань, що знаходиться на стику інженерії, економіки, юриспруденції та соціології. Означає раціональне використання енергетичних ресурсів, досягнення економічно доцільної ефективності використання існуючих паливно-енергетичних ресурсів при дійсному рівні розвитку техніки та технології і дотриманні вимог до навколишнього середовища [37].

Під енергоефективністю розуміють ефективне (раціональне) використання енергетичних ресурсів. З точки зору технологічних процесів, це споживання меншої кількості енергії для забезпечення того ж рівня процесів, що використовувалися до введення ефективних заходів. Досить яскравим прикладом енергоефективності є використання світлодіодної лампочки. Адже вона зазвичай використовує в 5 раз менше електроенергії, ніж звичайна лампа розжарювання, виробляючи при цьому освітлення того ж рівня.

Зазвичай, середній будинок споживає 95 мільйонів кілоджоулів енергії на рік, і значна частина цієї енергії витрачається даремно. Основною причиною цього є звички людей, які не вимикають світло, коли виходять з кімнати, забувають вимкнути телевізор, комп’ютер чи інші прилади, коли вони більше не використовуються. Такі дії призводять до втрати електроенергії та збільшення витрат. Зміна цих звичок може допомогти заощадити енергію, але це простіше сказати, ніж зробити.

Найбільш ефективним способом підвищення енергоефективності є зниження споживаних ресурсів. Цього можна досягти виключно впровадженням енергоефективних технологій. Такі системи дозволяють оптимізувати витрату енергоресурсів. Концепція розумного будинку знайома тим, хто вже замислювався над питанням економії. Для більшості людей, ще

кілька років тому ця система здавалася чимось захмарним як з боку фінансів, так і з боку технологічності. Слід зазначити, що впровадження такої технології при будівництві нового будинку або квартири обходиться забудовнику лише на 2-8% дорожче, ніж аналогічна площа, не укомплектована системами економії.

Система розумного будинку дозволяє об'єднувати в єдину систему такі агрегати як сонячні колектори і батареї для виробництва енергії і гарячої води, геотермальні установки, що працюють взимку на обігрів приміщень, а влітку - на кондиціювання.

Мешканці розумного будинку, використовуючи смартфон, можуть самостійно регулювати мікроклімат, підтримувати оптимальну температуру, вологість і чистоту повітря. Така система може самостійно вносити зміни в налаштуванні приладів, виходячи із заданих параметрів і змін навколишнього середовища. Для освітлення використовуються світлодіодні лампи, потужність яких регулюється давачами присутності, освітленості або ж на основі встановлених параметрів часу доби. Навіть якщо мешканець будинку забув вимкнути світло, система самостійно зробить це, оскільки “помітить”, що в приміщенні нікого немає.

Власники будинку можуть налаштувати енергоефективні таймери, детектори руху та димери, щоб змінювати освітлення у конкретно заданий час протягом дня. Після підключення такого освітлення до розумного помічника, його буде можна вмикати та вимикати за допомогою голосових команд.

Використовуючи мобільний додаток, можна створювати різні стратегії економії електроенергії або завантажити готові з мережі Інтернет. Для спрощення цього процесу можна завантажити додаткову підпрограму Google Assistant або Alexa. Завдяки їй можна швидко запланувати увімкнення та вимкнення необхідних приладів протягом дня.

Наприклад, програма “розумна кухня” може вмикати посудомийну машину, коли всі мешканці покинули дім або кожного ранку вмикати кавоварку в конкретно встановлений час.

Посудомийні і пральні машини та сушарки для одягу можуть одночасно працювати у кількох домах. Це регулярно є причиною додаткових навантажень на електромережу. Внаслідок цього загальний опір всіх електроприладів зменшується, що призводить до зайвого нагріву ліній електропередач та зниження напруги в мережі. Через це коефіцієнт корисної дії електроприладів зменшується, тому вони починають споживати більше енергії для виконання поставлених завдань. Для уникнення такої ситуації система розумного будинку, може відслідковувати цей процес і вмикати посудомийну і пральну машини та сушарку для одягу в момент, коли попит на електроенергію є мінімальний.

На рис. 6.1 наведено перелік електроприладів, які найбільше споживають електроенергії в домі.

Рис. 6.1. Співвідношення спожитої електроенергії приладами [38]

Багато приладів споживають енергію навіть тоді, коли вони не використовуються. Ігрові приставки, кавові апарати, кабельні або супутникові тюнери та будь-які інші пристрої, які перебувають у режимі очікування, також споживають електроенергію. В такому випадку система розумного будинку може самостійно відслідковувати час, коли вони не використовуються, і вимикати їм електроживлення. Електроприлади у домі, які можна підключити до розумної системи відключення електроживлення:

— Електрообігрівачі. Електрообігрівач може споживати в середньому від 750 до 1500 Вт. Система розумного будинку дозволить віддалено керувати ним і вмикати у запланований час.

— Кондиціонери. Електроенергія, яку споживає кондиціонер, становить близько 17% від загальної енергії спожитої будинком. Часто він працює більше, ніж потрібно. Для зменшення витрат на підігрів та охолодження повітря в приміщені можна зв’язати кондиціонер із розумним термометром для підтримки ідеальної домашньої температури.

— Освітлення. Воно споживає більше 10% електроенергії будинку. Використання детекторів руху і таймерів для вимкнення освітлення - це дві найпростіші стратегії економії енергії для розумних будинків.

— Пральні та посудомийні машини. Енергозберігаюча технологія для будинків дозволяє контролювати роботу приладів, в моменти пікового навантаження на електромережу.

— Кухонні прилади. Навіть коли кухонна техніка не використовується, дисплеї на мікрохвильовій печі, духовці та холодильнику залишаються увімкненими та споживають електроенергію. Система розумного будинку може переводити прилад у режим енергозбереження, вимикаючи дисплеї, коли пристрій не використовуються.

— Осушувачі та зволожувачі повітря. Вони часто залишаються увімкненими весь час. Використання розумних помічників для їх вимкнення, коли нікого немає вдома, забезпечує економію електроенергії розумного будинку, зберігаючи при цьому комфортний рівень вологості повітря.

Отже, використання енергозберіаючих технологій, дозволяє зменшити витрати енергоносіїв (вода, газ, електроенергія). Технологія розумного будинку не була б такою ефективною, якби вона не використовувала розумних помічники та модулі штучного інтелекту. Завдяки цьому власники такого будинку можуть безперешкодно взаємодіяти з ним. Це дозволяє підвищити енергоефективність розумного будинку в цілому.

6.2. Електромагнітне забруднення довкілля, його вплив на людину. Шляхи його зменшення

В сучасному світі, внаслідок розвитку індустріалізації та розвитку технологій, споживання електроенергії все більше зростає. В результаті навколишнє середовище зазнало електромагнітного забруднення. Через те, що воно невидиме, а його вплив відбувається не відразу, електромагнітному забрудненню не надавали великого значення порівняно з іншими видами забруднення навколишнього середовища [39].

Електромагнітні поля (ЕМП) – це змінні електричні та магнітні поля, що поширюються у просторі у формі хвиль зі швидкістю світла.

Із курсу фізики відомо, що навколо електричного заряду існує електричне поле, а кожний електричний заряд, що рухається, створює в навколишньому просторі магнітне поле. Отже, навколо будь-якого об'єкту, через який протікає постійний чи змінний струм, так само, як і навколо будь-якого магніту, що рухається, існує електромагнітне поле. Інакше кажучи, рух поля одного виду завжди супроводжується появою поля іншого виду: електричне поле, що рухається, створює магнітне, а магнітне поле, що рухається, створює електричне.

Можна вважати, що в електроустановках електричне поле виникає за наявності напруги на струмопровідних частинах, а магнітне - при проходженні струму в проводах.

Електромагнітне забруднення — це сукупність електромагнітних полів, різноманітних частот, що негативно впливають на людину.

Простір, що оточує людину, заповнений різними електромагнітними полями, джерела яких, залежно від їх походження, можна розділити на дві групи: природні та штучні.

До природних джерел належать: електромагнітне поле Землі; космічні джерела радіохвиль (сонячні спалахи, випромінювання зірок тощо); процеси, які відбуваються в атмосфері Землі (блискавки, зміни в іоносфері).

До штучних джерел належать пристрої, які спеціально створені для випромінювання електромагнітної енергії (радіо і телевізійні станції, радіолокаційні установки, системи радіозв'язку та ін.), а також пристрої, що безпосередньо не призначені для випромінювання електромагнітної енергії в простір (лінії електропередач і трансформаторні підстанції, побутова і промислова техніка, оргтехніка тощо).

Таким чином, спектр частот електромагнітних полів, що оточують людину, охоплює діапазон від 50 Гц до $3 \cdot 10^{26}$ Гц.

Однак трансформаторні підстанції та лінії електропередач викликають більше електромагнітне забруднення, ніж прилади, що використовуються в повсякденному житті людини. Внаслідок неправильної урбанізації та незапланованої структуризації, лінії високої напруги та підстанції розміщено в безпосередній близькості від великих міст. Внаслідок цього люди піддаються дії електромагнітного випромінення.

В Україні немає чітко вираженої проблеми з розміщенням ліній електропередач великої потужності поблизу населених пунктів. В основному до міст підходять лінії з напругою не вище 110 кіловольт.

Як вже згадувалося вище, магнітне поле виникає там, де є електричне поле і навпаки. У той час, як електричні поля можуть бути екрановані або послаблені, магнітне поле може проходити через різні матеріали. Тому дослідження впливу електромагнітних полів на здоров'я людини в основному концентрується на магнітному полі. Електромагнітне поле може впливає на живі організми як позитивно, так і негативно.

На жаль, сучасні уявлення про біологічну дію електромагнітного випромінення не дозволяють прогнозувати всі несприятливі наслідки. Багато аспектів проблеми не висвітлені в сучасній літературі та вимагають додаткових досліджень. У зв'язку з цим, відповідно до рекомендацій ВООЗ, доцільно дотримуватися попереджувальної політики, тобто максимально зменшити час перебування у місцях з підвищеним електромагнітним забрудненням.

Більшість побутових електроприладів виготовлені з дотримання державних санітарних правил і норм. Тому вони не несуть ніякої шкоди для здоров'я людини. Загрозу можуть нести прилади з пошкодженими функціональними вузлами, наприклад — несправні мікрохвильові печі. Зазвичай вони добре екрановані та мають систему блокування роботи магнетрона при відкритих дверцятах, яка ще і трикратно дублюється. Але при її поломці (що малоймовірно) бувають випадки, коли власник мікрохвильової печі або некомпетентний майстер, намагаються її відремонтувати, заблокувавши цю систему. Тоді увімкнута така мікрохвильова піч, при випадково незакритих дверцятах, може призвести до тимчасового погіршення самопочуття та зору.

Згідно ДСН 239-96 “Державні санітарні норми і правила захисту населення від впливу електромагнітних випромінювань” електромагнітне випромінювання не повинно перевищувати 10 мкВт/см².

Більшість базових станцій мобільного зв’язку підвищують електромагнітне забруднення довкілля. Але при правильному їх встановлені та експлуатації, вони не несуть ніякої шкоди людині. Слід відзначити, що для зв'язку таких станції між собою використовуються вузьконаправлені антени, тому перебування комах між ними може спричинити їх дезорієнтацію.

Введення в експлуатацію базових станцій мобільного зв’язку покоління 5G дозволить мінімізувати електромагнітне забруднення довкілля. Вони використовують решітчасті антени, що мають вужчу діаграму направленості, що дозволяє знизити потужність базової станції. Адаптивність антен дозволяє коригувати діаграму направленості, тому підвищена електромагнітна забрудненість буде лише в місцях скупчення мобільних пристроїв. Антену, потужністю 60 Ват, слід встановлювати не нижче 3 метрів над висотою будинків, і щоб видимість її в радіусі 31 метра не перекривали ніякі споруди.

Отже, при правильному конструюванні та експлуатації побутових електроприладів, використання сучасних технологій безпровідного зв’язку дозволить зменшити електромагнітне забруднення довкілля та вплив його на здоров’я людини.

ВИСНОВКИ

У даній дипломній роботі магістра проведено розробку системи контролю розумним будинком. При виконані роботи та проведення теоретичних і експериментальних досліджень було отримано наступні результати:

- проведено огляд публікацій авторів, які досліджували проблематику систем контролю розумним будинком, та на основі їх аналізу встановлено переваги та недоліки таких систем;
- знайдено спосіб об’єднати технології Mesh та LoRa для підвищення відмовостійкості системи загалом;
- сформульовано вимоги до системи контролю розумним будинком з використанням LoRa–Mesh-технологій;
- обґрунтовано методи використання LoRa–Mesh-технологій в системі контролю розумним будинком;
- розроблено архітектуру та алгоритмічне забезпечення системи контролю розумним будинком з використанням запропонованих методів;
- розроблено макет системи контролю розумним будинком з використання LoRa радіо-модуля RFM95 на базі трансивера RF96;
- розроблено програмне забезпечення для мережевих вузлів та шлюзу на базі платформи Arduino Nano та WI-FI модуля ESP8266 відповідно;
- протестовано та оцінено завадостійкість модулів безпровідного зв’язку для системи контролю розумним будинком;
- розроблено простий та інтуїтивно зрозумілий веб-інтерфейс для мережевого шлюзу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Моніторинг об‘єктів в умовах апріорної невизначеності джерел інформації. 2015. URL: http://ena.lp.edu.ua:8080/bitstream/ntb/39313/1/monohrafiya1.pdf (дата звернення: 19.11.2019).

2. Ram P. LPWAN, LoRa, LoRaWAN and the Internet of Things. 2019. URL: https://medium.com/coinmonks/lpwan-lora-lorawan-and-the-internet-of-things-aed7d 5975d5d (дата звернення: 3.10.2019).

3. Semtech Acquires Wireless Long Range IP Provider Cycleo. 2012. URL: https://www.design-reuse.com/news/28706/semtech-cycleo-acquisition.html (дата звернення: 21.10.2019).

4. Sanchez-Iborra R., Sanches-Gomes J., Ballesta-Vinas J. Performance Evaluation of LoRa Considering Scenario Conditions. 2018. p. 772.

5. Fargas B. C. GPS-free Geolocation using LoRa in Low-Power WANs. 2017. URL: https://orbit.dtu.dk/files/130478296/paper_final_2.pdf (дата звернення: 29.11.2019).

6. Jacquet P. Optimized Link State Routing Protocol. 2003. URL: https://tools.ietf.org/html/rfc3626 (дата звернення: 24.10.2019).

7. Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding. 2009. URL: https://tools.ietf.org/html/rfc5614 (дата звернення: 14.11.2019).

8. Atmel automotive microcontrollers ATmega128L datasheet. 2011. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/Doc2467.pdf (дата звернення: 12.10.2019).

9. Atmel automotive microcontrollers ATmega328P datasheet. 2015. URL: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (дата звернення: 3.10.2019).

10. Arduino Nano Pin Diagram. 2018. URL: https://components101.com/sites/default/files/component_pin/Arduino-Nano-Pinout.png (дата звернення: 2.11.2019).

11. RFM95/96/97/98(W) - Low Power Long Range Transceiver Module. URL: https://www.alldatasheet.com/view.jsp?Searchword=RFM95 (дата звернення: 27.10.2019).

12. RF96/97/98 - 137 MHz to 1020 MHz Low Power Long Range Transceiver. URL: https://www.alldatasheet.com/view.jsp?Searchword=RF96 (дата звернення: 28.10.2019).

13. ESP8266 Overview. 2017. URL: https://www.espressif.com/en/products/hardware/esp8266ex/overview (дата звернення: 14.10.2019).

14. Esp open sdk. 2018. URL: https://github.com/pfalcon/esp-open-sdk.git (дата звернення: 27.11.2019).

15. Prebuilt Windows Toolchain for ESP8266. 2017. URL: http://gnutoolchains.com/esp8266/ (дата звернення: 10.10.2019).

16. Григор'єв М. Unofficial Development Kit for Espressif ESP8266. 2016. URL: https://programs74.ru/udkew.html (дата звернення: 4.10.2019).

17. Arduino core for ESP8266 WiFi chip. 2017. URL: https://github.com/esp8266/Arduino.git (дата звернення: 16.11.2019).

18. Sming. 2014. URL: https://16ithub.com/SmingHub/Sming.git (дата звернення: 15.10.2019).

19. ESP12F Pin Diagram. URL: http://esp8266-arduinoide.ru/wp-content/uploads/2016/06/esp8266_esp12e _horizontal-01.png (дата звернення: 10.11.2019).

20. Обмен данными по интерфейсу I2C (TWI) на МК AVR. 2011. URL: http://nauchebe.net/2011/03/obmen-dannymi-po-interfejsu-i2c-twi-na-mk-avr/ (дата звернення: 2.10.2019).

21. Робота з рідкокристалічним дисплеєм. 2014. URL: https://dl.tntu.edu.ua/content.php?cid=152751 (дата звернення: 4.10.2019).

22. Караюз І. В., Назарчук І. В., Тимощук О. Л. Програмування і алгоритмічні мови. Алгоритмізація та основи програмування. Київ, 2017. 94 с.

23. uint8_t, uint16_t, uint32_t, uint64_t 의 의 미 . 2011. URL: https://mgoons.tistory.com/4 (дата звернення: 29.10.2019).

24. LoRa Mesh Networking with Simple Arduino-Based Modules. 2018. URL: https://github.com/nootropicdesign/lora-mesh (дата звернення: 15.10.2019).

25. ESP8266 Arduino Core URL: http://arduino.esp8266.com/Arduino/versions/2.0.0/doc/filesystem.html (дата звернення: 25.10.2019).

26. MQTT Version 5.0 OASIS Standard Specification. 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf (дата звернення: 11.11.2019).

27. Универсальный асинхронный приемопередатчик URL: https://cf2.ppt-online.org/files2/slide/b/b1auZXV0H3TWjf5AwLvizD8eyJhU6OQRx97l2gcmn/slide-2.jpg (дата звернення: 13.10.2019).

28. Ellingwood J. How To Use npm to Manage Node.js Packages on a Linux Server. 2014. URL: https://www.digitalocean.com/community/tutorials/how-to-use-npm-to-manage-node-js-packages-on-a-linux-server (дата звернення: 28.11.2019).

29. npm-install. 2016. URL: https://docs.npmjs.com/cli/install (дата звернення: 14.10.2019).

30. npm-semver. 2016. URL: https://docs.npmjs.com/misc/semver (дата звернення: 17.10.2019).

31. Шевченко Л. С. Основи економічної теорії. Харків, 2008. 448 с.

32. Закон України Про охорону праці Відомості Верховної Ради України (ВВР), № 49, 1992. 668 с.

33. Наказ міністерство соціальної політики України № 207 Про затвердження вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями від 14.02.2018.

34. ДСанПІН 3.3.2.007-98: Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин № 7 від 10.12.98.

35. Державні будівельні норми ДБН В.1.2-4-2006 “Інженерно-технічні заходи цивільного захисту”.

36. Постанова КМУ від 30.10.2013 № 841 “Про затвердження Порядку проведення евакуації у разі загрози виникнення або виникнення надзвичайних ситуацій”.

37. Енергоефективність. 2019. URL: http://dtek-pem.com.ua/energoeffectivnost (дата звернення: 5.10.2019).

38. Energy-Saving Strategies for Smart Homes. 2019. URL: https://blog.constellation.com/wp-content/uploads/2019/01/energy-saving-strategies-for-smart-homes.png (дата звернення: 7.10.2019).

39. Seker S., Cerezci O. Radyasyon Kusatmasi. Istanbul, 2000. p. 354.

Додаток А

Тези конференцій

Міністерство освіти і науки України,
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет імені
Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця
(Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

II МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

25-26 квітня 2019 р.

(збірник тез конференції)

Тернопіль 2019

З М І С Т

будовами. Зручність роботів в тому, що їхня будова може бути змінена і покращена для кращої продуктивності чи надійності. Одні роботи потрібні великі, для переміщення великих предметів, чи роботи у важких умовах. І навпаки, малі роботи можуть бути корисними у делікатних роботах, будь то медицина, розвідка чи ремонт важкодоступних систем. Для кожної з робіт можна пристосувати робота, тільки б вистачало потужностей.

Наприклад досить цікавим способом використання роботів було б розгортання мережі зв'язку на певній території. Якщо для встановлення мережі використовувати вежі зв'язку чи стовпи, то це займе багато часу, так як на це все потрібен час і матеріали, які також потрібно доставляти на місце розгортання. Якщо використовувати для цього роботів, мережу можна розгорнути в самі короткі терміни, так як даний спосіб автоматизовано може звірятися з даними геолокацій і максимально продуктивно використовувати території.

Оглядаючи простори інтернету можна наштовхнутися на цікаві рішення для різних задач. Можна знайти невеликого робота, довжиною не більше 20 см в ширину і висотою не більше 10 см, який переміщуватиметься по землі, маючи доступ в різні місця і переносячи на собі певну вагу. Робот який переміщається на ніжках має хорошу проходимість, малу енергозатратність і можливість повісити на робота певну вагу за рахунок бази, яка зручно модифікується. Якщо зробити певну кількість роботів, однакової конструкції, яка складатиметься з бази, акумулятора великої ємності, роздавача бездротового сигналу і модулю геолокації, то такі роботи можуть стати прекрасним способом розгортання мережі, наприклад на фестивалях чи масових заходах, де мережа працює жахливо через велику кількість користувачів одночасно. Якщо роботи, приєднані до сервера, на якому встановлене певне програмне забезпечення, то такі роботи стануть набагато надійнішим способом для покриття, так як в разі необхідності, вони можуть бути спрямовані в точку більшого скупчення абонентів. Даний спосіб також має плюси у згортанні і перенесені мережі на іншу територію, так як не залишає після себе ні слідів, ні будь яких залишків. Також можна моніторити стан мережі, стан роботів і вирішувати проблеми швидше ніж якби це були точки, які встановлення на висотних стовпах чи будь чому схожому.

Література

1. Robot Building for Beginners / Cook D., 2010.
2. Robot Building For Dummies / Roger Arrick, Nancy Stevenson., 2003
3. Probabilistic Robotics / Sebastian Thrun , Wolfram Burgard , Dieter Fox., 2005

УДК 004.7

Буцій Р. - ст. гр. СІм-51
Тернопільський національний технічний університет імені Івана Пулюя

ВПРОВАДЖЕННЯ MESH-ТЕХНОЛОГІЙ В СИСТЕМИ РОЗУМНОГО БУДИНКУ

Butsiy R.
Ternopil Ivan Puluj National Technical University

INTRODUCTION OF MESH TECHNOLOGIES IN BUILDING AUTOMATION CONTROL

Ключові слова: розумний будинок, бездротова mesh-технологія;
Keywords: smart home, wireless mesh technology.

На сьогоднішній день інформаційні технології все більше і більше інтегруються у наше повсякденне життя. У будинки все частіше вбудовуються, так звані, системи розумного будинку. До них висувається ряд вимог, зокрема: надійність, стабільність,

зручність у користуванні та низька ціна. Однією із найважливіших проблем, які виникають при функціонуванні систем є низька завадостійкість, невеликий радіус дії та низький ступінь захищеності, який зумовлений як передаванням даних каналами зв'язку, так і методами контролю, які використовуються системою.

Існуючі систем розумних будинків створюють з використанням стандарту IEEE 802.11a/b/g/n (Wi-Fi). Основними недоліками даного стандарту є висока споживча потужність та необхідність великих обчислювальних ресурсів для ефективного обслуговування стеку протоколів 802.11. Слід зазначити, що для коректної взаємодії та координації роботи кінцевих пристроїв системи є необхідність використання центрального вузла — маршрутизатора, який дозволяє поєднувати дві або більше мережі й керувати процесом маршрутизації. Відмова даного пристрою призведе до повної відмови мережі.

Сьогодні Mesh-технології отримали широкий попит на ринку. В Mesh-мережі всі вузли формально рівноправні. Проте обмін трафіку із зовнішнім оточенням відбувається через єдиний вузол. Такий вузол називають базовою станцією Mesh-мережі: саме на нього покладається частина необхідних для управління мережею функцій. При цьому управління доступом може відбуватися як централізованим способом, під управлінням базової станції, так і на основі механізму розподіленого управління. Це дозволяє зменшити навантаження на центральний вузол і навіть при відмові будь-якого кінцевого пристрою, зокрема базової станції, Mesh-мережа може самостійно переконфігуруватися, без втрат своєї працездатності, тому застосування Mesh-топології у системах розумного будинку дозволить відмовитися від маршрутизаторів та підвищити відмовостійкість таких системи загалом.

Дослідженнями, які стосувались Mesh-технологій, займалось багато зарубіжних вчених, серед яких Wang X., Yu Liu, Kin-Fai Tong та багато інших.

Зокрема, ними було розглянуто способи впровадження Mesh-технологій у сферу так званих "розумних систем", як компонентів розумного будинку та в область інтернету речей. Також ними було розглянуто реалізацію Mesh-топології, в якості альтернативи звичайній топології зірка, яка зараз використовується практично всюди та має ряд критичних недоліків.

Тому дослідження системи контролю розумним будинком з використанням Mesh-технологій та розробка власних радіо-модулів з використанням унікального протоколу передачі даних є актуальною задачею.

На основі аналізу вже існуючих пристроїв, розроблено структурну схему радіо-модуля, яка зображена на рисунку 1. Планується спроектувати та реалізувати два види радіо-модулів. Вони між собою будуть майже ідентичні, тільки в першому буде реалізований шлюз, який дозволить конвертувати мережеві протоколи одного типу фізичного середовища в протоколи іншого фізичного середовища.

Рисунок 1— Блок-схема основного модуля

Радіо-модуль буде являти собою невеличкий (кишеньковий) пристрій з

автономним живленням від літій-іонного акумулятора стандарту 18650. Він буде побудований на базі готового Wi-Fi модуля ESP8266, трансивера MRF49XA та мікроконтролера ATmega328p.

Література

1. Ian Akyildiz X. W. Wireless Mesh Networks / Ian Akyildiz. – London: John Wiley & Sons, 2009.
2. Kin-Fai T. Wireless Mesh Networks in IoT network [Електронний ресурс] / T. Kin-Fai, L. Yu // ResearchGate. – 2018. – Режим доступу до ресурсу: https://www.researchgate.net/publication/318295001_Wireless_Mesh_Networks_in_IoT_networks.
3. Okin J. Wireless Mesh Networks The Internet Revolution: The Not-for-Dummies Guide to the History, Technology, and Use of the Internet / Okin. – Maine: Winter Harbor, 2005. – (Ironbound Press).

УДК 004.051

Воробець Д. – ст. гр. СІм-51
Тернопільський національний технічний університет імені Івана Пулюя

ВПЛИВ НАДІЙНОСТІ АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ НА ЕФЕКТИВНІСТЬ ВИКОРИСТАННЯ ІНФОРМАЦІЙНОЇ ІНФРАСТРУКТУРИ ПІДПРИЄМСТВА

Науковий керівник: к.т.н., доцент Яцишин В.В.

Vorobets D.
Ternopil Ivan Puluj National Technical University

HARDWARE RELIABILITY AFFECT ON THE EFFICIENCY OF INFORMATION INFRASTRUCTURE OF COMPANY

Supervisor: PhD, Assoc. Prof. Yatsyshyn V.

Ключові слова: апаратне забезпечення, надійність, ефективність
Keywords: hardware, reliability, efficiency

Для ефективної роботи будь-якого підприємства, організації чи установи у яких впроваджено комп'ютерні системи, що є основою функціонування інформаційної інфраструктури підприємства, необхідно проводити оцінювання та забезпечення ефективності апаратного забезпечення, як базової складової. При цьому для своєчасного гарантування ефективної роботи комп'ютерних систем необхідно впроваджувати постійні процеси моніторингу стану обладнання, розробляти стратегії розвитку інформаційних систем для одержання максимального ефекту від їх використання.

На практиці для визначення ефективності інформаційних систем, в тому числі і комп'ютерних систем, вхідними даними для оцінювання виступають звіти IT підрозділів (за наявності таких) про кількість комп'ютерів і мережевого обладнання, їх параметри, встановлене програмне забезпечення та ін. На основі аналізу звітної інформації формуються пропозиції щодо необхідності покращення IT інфраструктури, поточних ремонтів техніки, розширення штату IT підрозділів.

Для оптимізації використання апаратних засобів комп'ютерних систем потрібно розробити метод оцінювання ефективності їх використання, який повинен базуватись на строгій формалізованій процедурі моніторингу та аналізу стану апаратного забезпечення та умов його використання.

Вимогами до процедури моніторингу стану апаратного забезпечення є простота реалізації та виконання, оскільки, багато компаній можуть використовувати сторонні IT фірми для обслуговування та розвитку власних IT інфраструктур.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Маріборский університет (Словения)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том ІІ

VIII Міжнародної науково-технічної конференції молодих учених та студентів

27-28 листопада 2019 року

УКРАЇНА
ТЕРНОПІЛЬ – 2019

ЗМІСТ

СЕКЦІЯ: КОМП'ЮТЕРНО-ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ЗВ'ЯЗКУ

УДК 004.7

Р.А. Буцій
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ВПРОВАДЖЕННЯ БЕЗДРОТОВИХ LoRa MESH-МЕРЕЖ НА БАЗІ ПЛАТФОРМИ ARDUINO UNO В СИСТЕМИ IoT

R.A. Butsiy

INTRODUCTION OF WIRELESS LoRa MESH-NETWORKS ON ARDUINO UNO PLATFORM IN IoT SYSTEMS

На сьогоднішній день у світі спостерігається ріст популярності концепції Інтернету речей (IoT). Все більше пристроїв, які оточують нас, починають використовувати цю технологію. Для стабільного функціонування вони повинні мати постійний зв'язок з хмарними сервісами. Це вимагає встановлення декількох мережевих шлюзів щоб збільшити зону покриття для належної роботи кінцевих пристроїв. При такому підході масштабованість мережі ускладнюється і унеможливлюється її мобільність. Для вирішення проблеми зв'язку кінцевих пристроїв з шлюзом потрібно використовувати mesh-технології.

Mesh-мережа - це топологія мережі, де вузли обмінюються повідомленнями один з одним або безпосередньо (якщо вони знаходяться в зоні покриття) або опосередковано через проміжні вузли. Наприклад, якщо вузол 1 хоче надіслати повідомлення вузлу 2, але він занадто далеко від вузла 2, повідомлення автоматично перенаправляється через проміжний вузол, який знаходиться в діапазоні обох вузлів. Фактично, шлях може включати кілька проміжних вузлів. Знаходження маршруту від вузла 1 до 2 обробляється окремим програмним блоком mesh-мережі. Кожен вузол може обмінюватися інформацією з будь-яким іншим вузлом, зокрема і в мережах з частковим з'єднанням. Надійність такого підходу полягає в тому, що пакети даних можна передати між модулями навіть якщо проміжні вийшли з ладу або тимчасово недоступні.

Для зниження енергоспоживання окремого вузла можна використовувати технологію LoRa. Радіомодуль, при радіусі дії до декількох кілометрів, має відносно низьку потужність. LoRa використовує радіочастотні діапазони, що не потребують ліцензування, такі як 433 МГц, 868 МГц (Європа) та 915 МГц (Північна Америка) [1].

Трансивер MRF95 на базі LoRa модема RF96 можуть використовуватися для зв'язку точка-точка або в мережі WAN, яка передбачає зв'язок з централізованою базовою станцією. Існує комплект бібліотек RadioHead, для платформи Arduino Uno з підтримкою даного радіомодуля. В нього входить менеджер RHMesh, який дозволяє передавати пакети даних, через один або більше проміжних вузлів. Менеджер вміє самостійно складати таблицю маршрутизації, використовуючи технологію Mesh.

Даний підхід реалізації IoT дозволить відмовитися від додаткових мережевих шлюзів, підвищити масштабованість та мобільність мережі, а використання технології LoRa розширить радіус зв'язку та зменшить енергоспоживання пристроїв.

Література

1. Sanchez-Iborra R. Performance Evaluation of LoRa Considering Scenario Conditions / R. Sanchez-Iborra, J. Sanches-Gomes, J. Ballesta-Vinas et al // Sensors.– 2018.– 18(3).– p.772.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ »

11–12 **грудня** 2019 **року**

ТЕРНОПІЛЬ
2019

СЕКЦІЯ 2. **ІНФОРМАЦІЙНІ СИСТЕМИ ТА ТЕХНОЛОГІЇ**

УДК 004.7

Р. Буцій
(Тернопільський національний технічний університет імені Івана Пулюя)

СИСТЕМА ЗБОРУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ МІКРОКЛІМАТУ РОЗУМНОГО БУДИНКУ З ВИКОРИСТАННЯМ LoRa MESH-ТЕХНОЛОГІЙ

UDC 004.7

R. Butsiy
(Ternopil Ivan Puluj National Technical University, Ukraine)

MICROCLIMATE DATA GATHERING AND VISUALIZATION SYSTEM OF SMART HOUSE BASED ON LoRaMESH-TECHNOLOGIES

Підтримка наперед визначених характеристик мікроклімату є одним з пріоритетних завдань для розумного будинку. Правильне поєднання базових мікрокліматичних параметрів, зокрема, температури, вологості, тиску, а також відстеження концентрації вуглекислого газу в повітрі, інтенсивності випромінювання, шумового забруднення, є доволі нетривіальною задачею. Для аналізу мікроклімату в приміщеннях встановлюють різні давачі. Більшість виробників систем розумного будинку використовують безпровідний зв'язок для збору метрик та управління опаленням, вентиляцією, тощо. При такому підході в будинках з великою площею, залізобетонними стінами та перекриттям часто виникають проблеми із зв'язком окремих елементів такої системи, що вимагає встановлення додаткових ретрансляторів сигналу від давачів. Для вирішення цієї проблеми доцільно використовувати LoRa-трансивери з підтримкою технології Mesh.

Для перевірки можливості поєднання технологій LoRa та Mesh було зібрано макет системи розумного будинку з використанням трансивера RFM95 і створено програмне забезпечення з підтримкою Mesh, яке візуалізує роботу мережі та результати замірювання температури і вологості (рис. 1 та рис. 2).

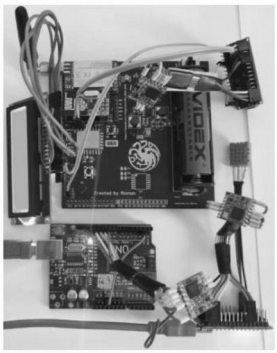

Рис. 1. Макет системи

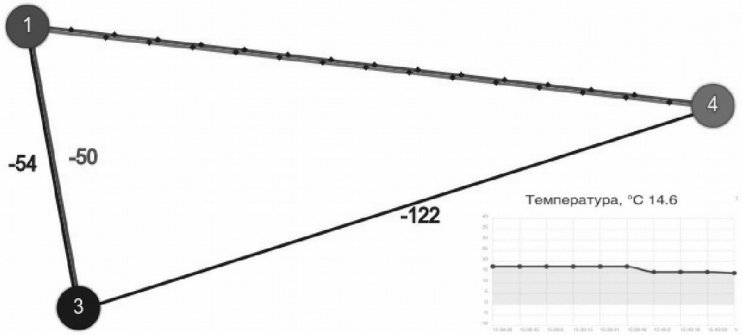

Рис. 2. Візуалізація LoRa-Mesh-мережі

Система складається з мережевого шлюзу (чорна плата на рис. 1 з id-1), вузла з давачем освітленості (плата Arduino UNO з id-3) і модуля вимірювання (плата Arduino Nano з давачем температури та вологості DHT11 з id-4). Розмістивши плату з давачем DHT11 у підвалі із залізобетонним перекриттям видно (див. рис. 2), що вузол 4 не може безпосередньо зв'язатися з мережевим шлюзом, але в радіусі дії обох вузлі знаходиться вузол з id-3, який виступає у ролі посередника між вузлами 1 та 4. Шлюз, використовуючи бібліотеку Chartist.js, візуалізує результати вимірювань у вигляді графіка. Фрагмент таблиці маршрутизації вузлів у JSON-форматі наведено нижче:

```
{"1": [{"n":255,"r":0},{"n":0,"r":0},{"n":3,"r":-50},{"n":3,"r":0}]}
{"3": [{"n":1,"r":-54},{"n":0,"r":0},{"n":255,"r":0},{"n":4,"r":-122}]}
{"4": [{"n":3,"r":0},{"n":0,"r":0},{"n":0,"r":0},{"n":255,"r":0}]}
```

Отже, даний підхід дозволяє використовувати компоненти системи розумного будинку (давачі, розумні лампи та вимикачі тощо), як ретранслятори для передачі даних між вузлами, що не мають безпосереднього зв'язку.

Додаток Б

Лістинг програми керування розумним будинком

```
#include "DHT.h"
#include <EEPROM.h>
#include <RHRouter.h>
#include <RHMesh.h>
#include <RH_RF95.h>
#define RH_HAVE_SERIAL
#define N_NODES 4
#define DHTPIN 3 // Вивід підключення давача

uint8_t nodeId;
uint8_t routes[N_NODES]; // Таблиця маршрутизації
int16_t rssi[N_NODES]; // RSSI вузлів
RH_RF95 driver;
RHMesh *manager;
char buf[RH_MESH_MAX_MESSAGE_LEN];
DHT dht(DHTPIN, DHT11);

void setup() {
  randomSeed(analogRead(0));
  Serial.begin(115200);
  while (!Serial);
  nodeId = EEPROM.read(0);
  Serial.print(F("initializing node: "));
  manager = new RHMesh(driver, nodeId);
  if (!manager->init()) {
    Serial.println(F("failed"));
  } else {
    Serial.println("done");
  }
  driver.setTxPower(23, false);
  driver.setFrequency(915.0);
  driver.setCADTimeout(500);
```

```
  Serial.println("RF95 ready");
  for(uint8_t n = 1;n <= N_NODES; n++) {
    routes[n-1] = 0;
    rssi[n-1] = 0;
  }
  dht.begin();
}

const __FlashStringHelper* getErrorString(uint8_t error) {
  switch(error) {
    case 1: return F("invalid length");
    break;
    case 2: return F("no route");
    break;
    case 3: return F("timeout");
    break;
    case 4: return F("no reply");
    break;
    case 5: return F("unable to deliver");
    break;
  }
  return F("unknown");
}

void updateRoutingTable() {
  for(uint8_t n = 1; n <= N_NODES; n++) {
    RHRouter::RoutingTableEntry *route =
                                        manager->getRouteTo(n);
    if (n == nodeId) {
      routes[n-1] = 255;
    } else {
      routes[n-1] = route->next_hop;
      if (routes[n-1] == 0) rssi[n-1] = 0;
    }
  }
```

```
}

void getRouteInfoString(char *p, size_t len) {
  p[0] = '\0';
  strcat(p, "[");
  for(uint8_t n = 1; n <= N_NODES; n++) {
    strcat(p, "{\"n\":");
    sprintf(p+strlen(p), "%d", routes[n-1]);
    strcat(p, ",");
    strcat(p, "\"r\":");
    sprintf(p+strlen(p), "%d", rssi[n-1]);
    strcat(p, "}");
    if (n < N_NODES) {
      strcat(p, ",");
    }
  }
  strcat(p, "]");
}

void printNodeInfo(uint8_t node, char *s) {
  Serial.print(F("node: "));
  Serial.print(F("{"));
  Serial.print(F("\""));
  Serial.print(node);
  Serial.print(F("\""));
  Serial.print(F(": "));
  Serial.print(s);
  Serial.println(F("}"));
}

void sendDataFromDHT(){
  getDataFromDHT(buf, RH_MESH_MAX_MESSAGE_LEN);
  sendAck(1);
  listenIncomingMessages();
}
```

```
bool getDataFromDHT(char *p, size_t len) {
  int16_t t = dht.readTemperature()*100;
  uint8_t h = dht.readHumidity();
  p[0] = '\0';
    strcat(p, "{\"M\":");
    strcat(p, "\"DHT11\",\"t\":");
    sprintf(p + strlen(p), "%d", t);
    strcat(p, ",");
    strcat(p, "\"h\":");
    sprintf(p + strlen(p), "%d", h);
    strcat(p, "}");
  return true;
}

void receiveAckNextHop(uint8_t n) {
  Serial.println(F(" OK"));
  RHRouter::RoutingTableEntry *route =
                                        manager->getRouteTo(n);
  if (route->next_hop != 0) {
    rssi[route->next_hop-1] = driver.lastRssi();
  }
}

void sendAck(uint8_t n) {
  Serial.print(F("->"));
  Serial.print(n);
  Serial.print(F(" :"));
  Serial.print(buf);
  if (nodeId == 1) printNodeInfo(nodeId, buf);
  uint8_t error =
         manager->sendtoWait((uint8_t *)buf, strlen(buf), n);
  if (error != RH_ROUTER_ERROR_NONE) {
    Serial.println();
    Serial.print(F(" ! "));
    Serial.println(getErrorString(error));
```

```
  } else {
    receiveAckNextHop(n);
  }
}

void listenIncomingMessages() {
    unsigned long nextTransmit =
                                millis() + random(3000, 5000);
    while (nextTransmit > millis()) {
      int waitTime = nextTransmit - millis();
      uint8_t len = sizeof(buf);
      uint8_t from;
      if (manager->recvfromAckTimeout((uint8_t *)buf, &len,
                                          waitTime, &from)) {
        buf[len] = '\0';
        Serial.print(from);
        Serial.print(F("->"));
        Serial.print(F(" :"));
        Serial.println(buf);
        if (nodeId == 1) printNodeInfo(from, buf);
        RHRouter::RoutingTableEntry *route =
                                    manager->getRouteTo(from);
        if (route->next_hop != 0) {
          rssi[route->next_hop-1] = driver.lastRssi();
        }
      }
    }
  }

void loop() {
  for(uint8_t n = 1; n <= N_NODES; n++) {
    if (n == nodeId) continue;
    getRouteInfoString(buf, RH_MESH_MAX_MESSAGE_LEN);
    sendAck(n);
    listenIncomingMessages();
```

```
    updateRoutingTable();
  }
  sendDataFromDHT();
}

/*
    Gateway
    Розробив Roman
    Канал на YouTube: https://goo.gl/x8FL2o
    Github: https://github.com/RomanButsiy/lora-mesh
    Схема: https://easyeda.com/Targaryen/LoRa_Mesh_Node
    2019 Roman
*/

#include <ESP8266HTTPUpdateServer.h>
#include <LiquidCrystal_I2C.h>
#include <ESP8266WebServer.h>
#include <PubSubClient.h>
#include <ESP8266WiFi.h>
#include <ESP8266SSDP.h>
#include <ArduinoJson.h>  // Ставимо через менеджер бібліотек
#include <RH_RF95.h>
#include <Wire.h>
#include <time.h>
#include <FS.h>
#define BUFSIZE RH_RF95_MAX_MESSAGE_LEN

ESP8266HTTPUpdateServer httpUpdater;
ESP8266WebServer HTTP(80);
File fsUploadFile;
WiFiClient espClient;
PubSubClient mqtt_client(espClient);
String configSetup = "{}";
String configJson = "{}";
bool mqtt_State = false;
```

```
void setup() {
  gateway_init();
}
void loop() {
  HTTP.handleClient();
  mqtt_listener();
}

void gateway_init(void) {
  Serial.begin(115200);
  while (!Serial);
  FS_init();
  configSetup = readFile("config.json", 4096);
  jsonWrite(configJson, "SSDP",
                              jsonRead(configSetup, "SSDP"));
  WIFIinit();
  Time_init();
  SSDP_init();
  HTTP_init();
  MQTT_init();
  LANG_init();
  BUTTONS_init();
  DHT_init();
}

void BUTTONS_init(void) {
  HTTP.on("/ButtonsControl", HTTP_GET, [](){
    HTTP.send(200, "text/plain", "OK");
  });
  HTTP.on("/setChartRefreshDHT", HTTP_GET, [](){
    uint16_t data = HTTP.arg("set").toInt();
    if ((data >= 1000) && (data <= 65000)) {
      jsonWrite(configSetup, "ChartRefreshDHT", data);
      saveConfig();
      HTTP.send(200, "text/plain", "OK");
```

```
    } else {
      HTTP.send(400, "text/plain", "Out of range");
    }
  });
}

void FS_init(void) {
  SPIFFS.begin();
  {
    Dir dir = SPIFFS.openDir("/");
    while (dir.next()) {
      String fileName = dir.fileName();
      size_t fileSize = dir.fileSize();
    }
  }
  HTTP.on("/list", HTTP_GET, handleFileList);
  HTTP.on("/edit", HTTP_GET, []() {
    if (!handleFileRead("/edit.htm"))
       HTTP.send(404, "text/plain", "FileNotFound");
  });
  HTTP.on("/edit", HTTP_PUT, handleFileCreate);
  HTTP.on("/edit", HTTP_DELETE, handleFileDelete);
  HTTP.on("/edit", HTTP_POST, []() {
    HTTP.send(200, "text/plain", "");
  }, handleFileUpload);
  HTTP.onNotFound([]() {
    if (!handleFileRead(HTTP.uri()))
      HTTP.send(404, "text/plain", "FileNotFound");
  });
}
String getContentType(String filename) {
  if (HTTP.hasArg("download"))
     return "application/octet-stream";
  else if (filename.endsWith(".htm")) return "text/html";
  else if (filename.endsWith(".html")) return "text/html";
```

```
  else if (filename.endsWith(".json"))
     return "application/json";
  else if (filename.endsWith(".css"))
    return "text/css";
  else if (filename.endsWith(".js"))
    return "application/javascript";
  else if (filename.endsWith(".png")) return "image/png";
  else if (filename.endsWith(".gif")) return "image/gif";
  else if (filename.endsWith(".jpg")) return "image/jpeg";
  else if (filename.endsWith(".ico")) return "image/x-icon";
  else if (filename.endsWith(".xml")) return "text/xml";
  else if (filename.endsWith(".pdf"))
    return "application/x-pdf";
  else if (filename.endsWith(".zip")
    return "application/x-zip";
  else if (filename.endsWith(".gz"))
    return "application/x-gzip";
  return "text/plain";
}

bool handleFileRead(String path) {
  if (path.endsWith("/")) path += "index.htm";
  String contentType = getContentType(path);
  String pathWithGz = path + ".gz";
  if (SPIFFS.exists(pathWithGz) || SPIFFS.exists(path)) {
    if (SPIFFS.exists(pathWithGz))
      path += ".gz";
    File file = SPIFFS.open(path, "r");
    size_t sent = HTTP.streamFile(file, contentType);
    file.close();
    return true;
  }
  return false;
}
```

```
void handleFileUpload() {
  if (HTTP.uri() != "/edit") return;
  HTTPUpload& upload = HTTP.upload();
  if (upload.status == UPLOAD_FILE_START) {
    String filename = upload.filename;
    if (!filename.startsWith("/")) filename = "/" + filename;
    fsUploadFile = SPIFFS.open(filename, "w");
    filename = String();
  } else if (upload.status == UPLOAD_FILE_WRITE) {
    if (fsUploadFile)
      fsUploadFile.write(upload.buf, upload.currentSize);
  } else if (upload.status == UPLOAD_FILE_END) {
    if (fsUploadFile)
      fsUploadFile.close();
  }
}

void handleFileDelete() {
  if (HTTP.args() == 0)
    return HTTP.send(500, "text/plain", "BAD ARGS");
  String path = HTTP.arg(0);
  if (path == "/")
    return HTTP.send(500, "text/plain", "BAD PATH");
  if (!SPIFFS.exists(path))
    return HTTP.send(404, "text/plain", "FileNotFound");
  SPIFFS.remove(path);
  HTTP.send(200, "text/plain", "");
  path = String();
}

void handleFileCreate() {
  if (HTTP.args() == 0)
    return HTTP.send(500, "text/plain", "BAD ARGS");
  String path = HTTP.arg(0);
```

```
  if (path == "/")
    return HTTP.send(500, "text/plain", "BAD PATH");
  if (SPIFFS.exists(path))
    return HTTP.send(500, "text/plain", "FILE EXISTS");
  File file = SPIFFS.open(path, "w");
  if (file)
    file.close();
  else
    return HTTP.send(500, "text/plain", "CREATE FAILED");
  HTTP.send(200, "text/plain", "");
  path = String();
}

void handleFileList() {
  if (!HTTP.hasArg("dir")) {
    HTTP.send(500, "text/plain", "BAD ARGS");
    return;
  }
  String path = HTTP.arg("dir");
  Dir dir = SPIFFS.openDir(path);
  path = String();
  String output = "[";
  while (dir.next()) {
    File entry = dir.openFile("r");
    if (output != "[") output += ',';
    bool isDir = false;
    output += "{\"type\":\"";
    output += (isDir) ? "dir" : "file";
    output += "\",\"name\":\"";
    output += String(entry.name()).substring(1);
    output += "\"}";
    entry.close();
  }
  output += "]";
  HTTP.send(200, "text/json", output);
```

```
}
void MQTT_init(void) {
  HTTP.on("/mqtt", HTTP_GET, []() {
  jsonWrite(configSetup, "mqttState",
                              HTTP.arg("mqtt-state").toInt());
  jsonWrite(configSetup, "mqttPort",
                               HTTP.arg("mqtt-port").toInt());
  jsonWrite(configSetup, "mqttServer",
                                    HTTP.arg("mqtt-server"));
  jsonWrite(configSetup, "mqttUser", HTTP.arg("mqtt-user"));
  jsonWrite(configSetup, "mqttPass", HTTP.arg("mqtt-pass"));
  saveConfig();
    HTTP.send(200, "text/plain", "OK");
  });
  HTTP.on("/mqtt_reconnect", HTTP_GET, []() {
    mqtt_State = true;
    if (mqtt_start()) {
      HTTP.send(200, "text/plain", "OK");
    } else {
      HTTP.send(503, "text/plain", "Service Unavailable");
    }
  });
  mqtt_State = jsonReadtoInt(configSetup, "mqttState");
  if (mqtt_State) {
    mqtt_start();
  }
}

bool mqtt_start(void) {
  String _server = jsonRead(configSetup, "mqttServer");
  int _port = jsonReadtoInt(configSetup, "mqttPort");
  mqtt_client.setServer(_server.c_str(), _port);
  return mqtt_connect();
}
```

```
void mqtt_listener(void) {
  if (mqtt_State) {
    if (!mqtt_client.connected()) {
      if (!mqtt_connect()) {
        return;
      }
    }
    mqtt_client.loop();
    if (Serial.available()) {
      Serial.setTimeout(100);
      String _serial = Serial.readStringUntil('\n');
      if (_serial.startsWith("node: ")) {
        bool node = (_serial.indexOf("M") == -1);
        node = node && (_serial.indexOf("r") != -1);
        node = node && (_serial.indexOf("n") != -1);
        if (node) {
          char msg[BUFSIZE];
          int end = _serial.indexOf('\r');
          if (end > 0) {
            snprintf (msg, BUFSIZE, _serial.substring(6
                                              , end).c_str());
          } else {
            snprintf (msg, BUFSIZE,
                                 _serial.substring(6).c_str());
          }
          mqtt_client.publish("mesh_gateway/data", msg);
        }
      } else {
        readDHT(_serial);
      }
    }
  } else {
    if (Serial.available()) {
      Serial.setTimeout(100);
```

```
      String _serial = Serial.readStringUntil('\n');
      if (!_serial.startsWith("node: ")) {
        readDHT(_serial);
      }
    }
  }
}
void readDHT(String &_serial) {
  if (_serial.indexOf("DHT11") != -1) {
    uint8_t found = _serial.indexOf(':') + 1;
    _serial = _serial.substring(found);
    jsonWrite(configJson, "Temperature",
                                jsonReadtoInt(_serial, "t"));
    jsonWrite(configJson, "Humidity",
                                jsonReadtoInt(_serial, "h"));
  }
}

bool mqtt_connect() {
  byte tries = 5;
  String _clientId = jsonRead(configSetup, "SSDP");
  String _username = jsonRead(configSetup, "mqttUser");
  String _password = jsonRead(configSetup, "mqttPass");
  S_log("MQTT connecting", 0);
  S_log("", 1, false);
  while (!mqtt_client.connected() && --tries) {
    if (mqtt_client.connect(_clientId.c_str(),
                    _username.c_str(), _password.c_str())) {
      delay(1000);
      mqtt_State = true;
      return true;
    }
  }
  mqtt_State = false;
  return false;
```

```
}
void DHT_init() {
  HTTP.on("/getTemperature.json", HTTP_GET, []() {
    String data = graf((float)(jsonReadtoInt(configJson,
                                        "Temperature")/100.0));
    jsonWrite(data, "refresh",
                     getTimeChartRefresh("ChartRefreshDHT"));
    HTTP.send(200, "application/json", data);
  });
  HTTP.on("/getHumidity.json", HTTP_GET, []() {
    String data = graf(jsonReadtoInt(configJson,
                                                  "Humidity"));
    jsonWrite(data, "refresh",
                     getTimeChartRefresh("ChartRefreshDHT"));
    HTTP.send(200, "application/json", data);
  });
}

uint16_t getTimeChartRefresh(String name) {
  uint16_t _time = jsonReadtoInt(configSetup, name);
  if (_time >= 1000) {
    return _time;
  } else {
    return 10000;
  }
}

void HTTP_init(void) {
  HTTP.on("/config.live.json", HTTP_GET, []() {
    jsonWrite(configJson, "time", GetTime());
    jsonWrite(configJson, "date", GetDate());
    HTTP.send(200, "application/json", configJson);
  });
  HTTP.on("/config.setup.json", HTTP_GET, []() {
    HTTP.send(200, "application/json", configSetup);
```

```
  });
  HTTP.on("/restart", HTTP_GET, []() {
    String restart = HTTP.arg("device");
    if (restart == "ok") {
      HTTP.send(200, "text / plain", "Reset OK");
      ESP.restart();
    }
    else HTTP.send(200, "text / plain", "No Reset");
  });
  httpUpdater.setup(&HTTP);
  HTTP.begin();
}

void LANG_init(void) {
  HTTP.on("/lang", HTTP_GET, []() {
  jsonWrite(configSetup, "lang", HTTP.arg("set"));
  saveConfig();
  HTTP.send(200, "text/plain", "OK");
  });
  }

void SSDP_init(void) {
  HTTP.on("/description.xml", HTTP_GET, []() {
    SSDP.schema(HTTP.client());
  });
  HTTP.on("/ssdp", HTTP_GET, []() {
    String ssdp = HTTP.arg("ssdp");
    configJson=jsonWrite(configJson, "SSDP", ssdp);
    configJson=jsonWrite(configSetup, "SSDP", ssdp);
    SSDP.setName(jsonRead(configSetup, "SSDP"));
    saveConfig();
    HTTP.send(200, "text/plain", "OK");
  });
  SSDP.setDeviceType("upnp:rootdevice");
  SSDP.setSchemaURL("description.xml");
```

```
  SSDP.setHTTPPort(80);
  SSDP.setName(jsonRead(configSetup, "SSDP"));
  SSDP.setSerialNumber("000000000000");
  SSDP.setURL("/");
  SSDP.setModelName("SSDP-MeshGateway");
  SSDP.setModelNumber("000000000001");
  SSDP.setModelURL("https://goo.gl/x8FL2o");
  SSDP.setManufacturer("Roman Butsiy");
  SSDP.setManufacturerURL("https://goo.gl/x8FL2o");
  SSDP.begin();
}

void WIFIinit() {
  HTTP.on("/ssid", HTTP_GET, []() {
  jsonWrite(configSetup, "ssid", HTTP.arg("ssid"));
  jsonWrite(configSetup, "password", HTTP.arg("password"));
  saveConfig();
  HTTP.send(200, "text/plain", "OK");
  });
  HTTP.on("/ssidap", HTTP_GET, []() {
  jsonWrite(configSetup, "ssidAP", HTTP.arg("ssidAP"));
  jsonWrite(configSetup, "passwordAP",
                                    HTTP.arg("passwordAP"));
  saveConfig();
  HTTP.send(200, "text/plain", "OK");
  });

  WiFi.mode(WIFI_STA);
  byte tries = 11;
  String _ssid = jsonRead(configSetup, "ssid");
  String _password = jsonRead(configSetup, "password");
  if (_ssid != "") {
    WiFi.begin(_ssid.c_str(), _password.c_str());
    while (--tries && WiFi.status() != WL_CONNECTED) {
      delay(1000);
```

```
    }
    if (WiFi.status() != WL_CONNECTED) {
      StartAPMode();
    } else {
  } else StartAPMode();
}

bool StartAPMode() {
  IPAddress apIP(192, 168, 0, 1);
  WiFi.disconnect();
  WiFi.mode(WIFI_AP);
  WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
  String _ssidAP = jsonRead(configSetup, "ssidAP");
  String _passwordAP = jsonRead(configSetup, "passwordAP");
  WiFi.softAP(_ssidAP.c_str(), _passwordAP.c_str());
  return true;
}

String jsonRead(String &json, String name) {
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(json);
  return root[name].as<String>();
}

int jsonReadtoInt(String &json, String name) {
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(json);
  return root[name];
}

String jsonWrite(String &json, String name, String volume) {
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(json);
  root[name] = volume;
  json = "";
```

```
  root.printTo(json);
  return json;
}

String jsonWrite(String &json, String name, int volume) {
  DynamicJsonBuffer jsonBuffer;
  JsonObject& root = jsonBuffer.parseObject(json);
  root[name] = volume;
  json = "";
  root.printTo(json);
  return json;
}

void saveConfig (){
  writeFile("config.json", configSetup );
}

String readFile(String fileName, size_t len ) {
  File configFile = SPIFFS.open("/" + fileName, "r");
  if (!configFile) {
    return "Failed";
  }
  size_t size = configFile.size();
  if (size > len) {
    configFile.close();
    return "Large";
  }
  String temp = configFile.readString();
  configFile.close();
  return temp;
}

String writeFile(String fileName, String strings ) {
  File configFile = SPIFFS.open("/" + fileName, "w");
  if (!configFile) {
```

```
    return "Failed to open config file";
  }
  configFile.print(strings);
  configFile.close();
  return "Write sucsses";
}

String graf(int datas) {
  String root = "{}";
  DynamicJsonBuffer jsonBuffer;
  JsonObject& json = jsonBuffer.parseObject(root);
  JsonArray& data = json.createNestedArray("data");
  data.add(datas);
  root = "";
  json.printTo(root);
  return root;
}

String graf(float datas) {
  String root = "{}";
  DynamicJsonBuffer jsonBuffer;
  JsonObject& json = jsonBuffer.parseObject(root);
  JsonArray& data = json.createNestedArray("data");
  data.add(datas);
  root = "";
  json.printTo(root);
  return root;
}
```