Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

Пояснювальна записка

до бакалаврськогї дипломної роботи
бакалавра

на тему «Розподілена мережа для автоматичного управління доступом»

Виконав: студент 2 курсу, групи
КІ 22мс
спеціальності 123 — Комп'ютерна інженерія
Ляховецький Д.С.
Керівник к.т.н., доц. каф. ОТ
Колесник І.С.
" — " _____ 2024р.
Рецензент к.т.н., зав. каф. МБІС
Карпінець В.В.
" — " _____ 2024р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
« 17 » 06 2024р.

ВНТУ – 2024 року

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 123 – «Комп'ютерна інженерія»
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О. Д. Азаров
«12» березня 2024 р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Ляховецькому Денису Сергійовичу

1 Тема роботи «Розподілена мережа для автоматичного управління доступом »

Керівник роботи Колесник Ірина Сергіївна, к.т.н., доцент, затверджені наказом вищого навчального закладу від «12» березня 2024 р. №80

2 Строк подання студентом проекту 07.06.20.24 р.

3 Вихідні дані до проекту: технічні параметри розподіленої мережі для автоматичного управління доступом.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, огляд та аналіз розподілених мереж ESP-MESH, проектування автоматизованої мережі для автоматичного управління доступом, результати проектування та їх тестування, висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) структура системи контролю доступу, структура бази даних, мережа ESP-MESH, реалізована за допомогою інструмента ESP-IDF

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1– 3	Колесник І.С., к.т.н., доцент кафедри ОТ		

7 Дата видачі завдання 12.03.2024 р. Календарний план виконання роботи наведено в таблиці 7.1.

Таблиця 1 — Календарний план

з/п	Назва етапів виконання бакалаврського проекту	Строк виконання етапів роботи	Примітка
1	Постановка задач проекту	27.02.24	
2	Огляд інформаційних джерел	06.03-20.03.24	
3	Огляд та аналіз розподілених мереж ESP-MESH	27.03-7.04.24	
4	Проектування автоматизованої мережі для автоматичного управління доступом	10.04-21.04.24	
5	Результати проектування та їх тестування	24.04-12.05.24	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	15.05-26.05.24	
7	Аналіз виконання проекту. Висновки. Додатки	29.05-09.06.24	
8	Перевірка якості виконання бакалаврського проекту та усунення недоліків	07.06.24	

Студент _____ Ляховецький Д.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ І.С. Колесник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.9

Ляховецький Д.С.

Розподілена мережа для автоматичного управління доступом. Бакалаврська дипломна робота зі спеціальності 123 — комп'ютерна інженерія, освітня програма — комп'ютерна інженерія. Вінниця: ВНТУ, 2024, 68 с.

На укр.мові. Бібліогр.: 19 назв, рис. 28, табл. 4.

У бакалаврській дипломній роботі представлено розробку розподіленої мережі для автоматичного управління доступом.

У ході проектування у першій частині роботи досліджено концепцію ESP-MESH та структуру мережі ESP-MESH, яка відрізняється від традиційних інфраструктурних мереж Wi-Fi тим, що вузли не потрібно підключатися до центрального вузла. Вузли несуть взаємну відповідальність за ретрансляцію один одного.

У другій частині описано проектування розподіленої мережі для автоматичного управління доступом, а саме, реалізацію мережі ESP32-WiFi-Mesh, розробку системи керування користувачами та хмарний сервер.

Третя частина присвячена перевірці результатів проектування та їх тестуванню. Тестування стосувалося, в основному, функцій контролю доступу та самовідновлення пористих мереж

Розробка розподіленої мережі для автоматичного управління доступом є функціонально завершено, але залишає модивість подальшої модернізації.

Ключові слова: розподілена мережа, автоматичне управління, структура, користувач, хмарний сервіс.

.

ANNOTATION

Lyakhovetsky D.S.

Distributed network for automatic access control. Bachelor thesis in the specialty 123 — computer engineering, educational program system programming. Vinnitsa, VTNU, 2024, 68 p.

In the Ukr.leng. Libr.name 19, figure 28, table 4

The bachelor thesis presents the development of a distributed network of automatic access control.

During the design, the first part of the work explored the concept of ESP-MESH and the structure of the ESP-MESH network, which differs from traditional Wi-Fi infrastructure networks in that nodes do not need to be connected to a central node. Nodes are mutually responsible for relaying each other.

The second part describes the design of a distributed network for automatic access control, namely the implementation of the ESP32-WiFi-Mesh network, the design of the user management system and the cloud server.

The third part is devoted to the verification of design results and their testing. Testing mainly concerned the access control and self-healing functions of porous networks

The development of the distributed network of automatic access control is functionally complete, but leaves room for further modernization.

Keywords: distributed network, automatic control, structure, user, cloud service.

ЗМІСТ

ВСТУП

Системи контролю доступу ідентифікують людину та дозволяють або забороняють доступ людини для входу в приміщення, тим самим забезпечуючи повний захист та безпеку системи. Сучасні системи контролю доступу — це електронні системи, які призначені для керування через мережу з можливістю зміни списку користувачів, яким дозволено входити до приміщення, а також із веденням історії доступу.

Традиційна система контролю доступу використовує кабель передачі даних або Wi-Fi для прямого підключення до маршрутизатора для доступу до бази даних на сервері. Поруч із кожною машиною контролю доступу має бути маршрутизатор, що призводить до складної проводки або складних методів встановлення мережі. Таким чином, прокладання провідної мережі є складним і дорогим, що призводить до високої вартості розгортання системи контролю доступу. Для вирішення цієї проблеми пропонується метод використання бездротової MESH мережі, що самоорганізується. Вузли в мережі можуть підключатися один до одного та передавати інформацію. Теоретично відстань зв'язку між вузлами може досягати понад 100 метрів, якщо між вузлами немає таких бар'єрів, як стіни. У реальному житті через перешкоди відстань зв'язку може досягати десятків метрів. У пористій мережі ESP-WiFi може бути більше 1000 вузлів, тому можна реалізувати мережу ESP-WiFi з великим радіусом дії. У мережі встановлюємо вузол як кореневий вузл, кореневий вузол може використовуватися як точка доступу для підключення до Інтернету, а інші вузли можуть надсилати свої власні запити доступу до Інтернету кореневому вузлу, а потім кореневий вузол надсилатиме запит назад відправнику. На основі цього методу розроблено систему контролю доступу на базі ESP-WiFi-Mesh.

У роботі описано метод функціонування пористої мережі та процес реалізації системи контролю доступу на основі мікроконтролера ESP32 та пористої мережі.

Об’єктом дослідження є ESP-Mesh розподілені мережі, що можуть бути застосовані для автоматичного управління доступом.

Предмет дослідження є методи та засоби автоматичного управління доступом через розподілені мережі.

Мета роботи – проектування розподіленої мережі для автоматичного управління доступом.

Задачі роботи є наступними:

— отримати навики розробки веб-сторінок;

— освоїти можливість розробки веб-сторінок з використанням мов HTML+CSS та JavaScript;

— реалізувати розгортання та використання сервера;

— вивчити функції та використання хмарного сервера;

— освоїти метод роботи та використання бази даних на основі MySQL.

— оглянути та вивчити можливості мови PHP для обробки запитів доступу до веб-сторінки.

Практичне значення. Розподілена мережа для автоматичного управління доступом використовується для створення систем контролю доступу, що дозволяють управляти правами доступу до приміщень, об'єктів або послуг за допомогою централізованої системи, що забезпечує безпеку та зручність користувачів. Це дозволяє ефективно керувати доступом працівників, клієнтів або відвідувачів у різноманітних сферах діяльності, від бізнесу та корпорацій до громадських місць.

1 ОГЛЯД ТА АНАЛІЗ РОЗПОДІЛЕНИХ МЕРЕЖ ESP-MESH

1.1 Концепція ESP Mesh

Традиційна інфраструктурна мережа Wi-Fi реалізує мережеву топологію «зірка» і є мережею з багатоточковим з'єднанням, в якій один центральний вузол, відомий як точка доступу (AP), безпосередньо підключений до всіх інших вузлів, відомих як станції. AP відповідає за арбітраж та пересилання передач між станціями (див. рис. 1.1). Деякі точки доступу також ретранслюють передачі у зовнішню IP-мережу та з неї через маршрутизатор. Традиційні інфраструктурні мережі Wi-Fi страждають на брак обмеженої зони покриття через вимогу, щоб кожна станція перебувала у зоні дії для прямого підключення до точки доступу [1-22].

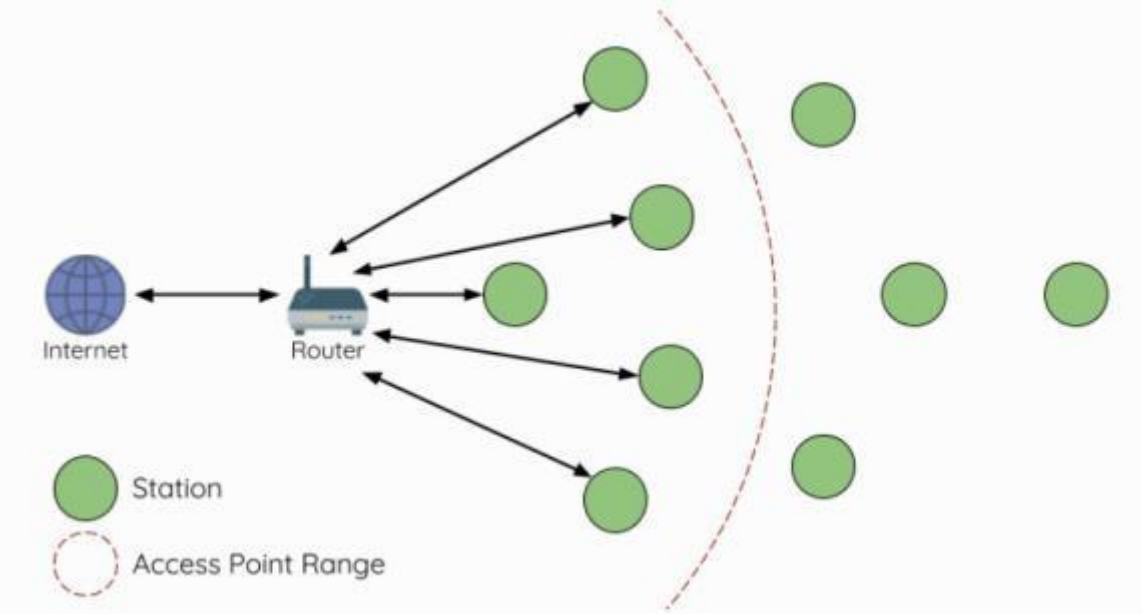

Рисунок 1.1 — Традиційна мережева архітектура Wi-Fi (топологія «зірка»)

Уявіть, що ми знаходимося у великій заводській будівлі і, якщо нам потрібно підключити наше обладнання до мережі або дозволити їм спілкуватися один з одним, нам потрібно прокласти мережеві кабелі в кожному куті заводу або використовувати кабелі для їхнього з'єднання. Це спосіб мало того, що має більше обмежень на відстані, але також вимагає

ремонту та електромонтажу, що обходиться дорожче. Ми знаємо, що у традиційній мережі Wi-Fi всі вузли мають бути підключені до маршрутизатору для зв'язку друг з одним. Чи можуть вузли підключатися один до одного, щоб кожен вузол став станцією передачі та відправником інформації? Звичайно, так, і мережа ESP Mesh призначена для цього (див. рис. 1.2).

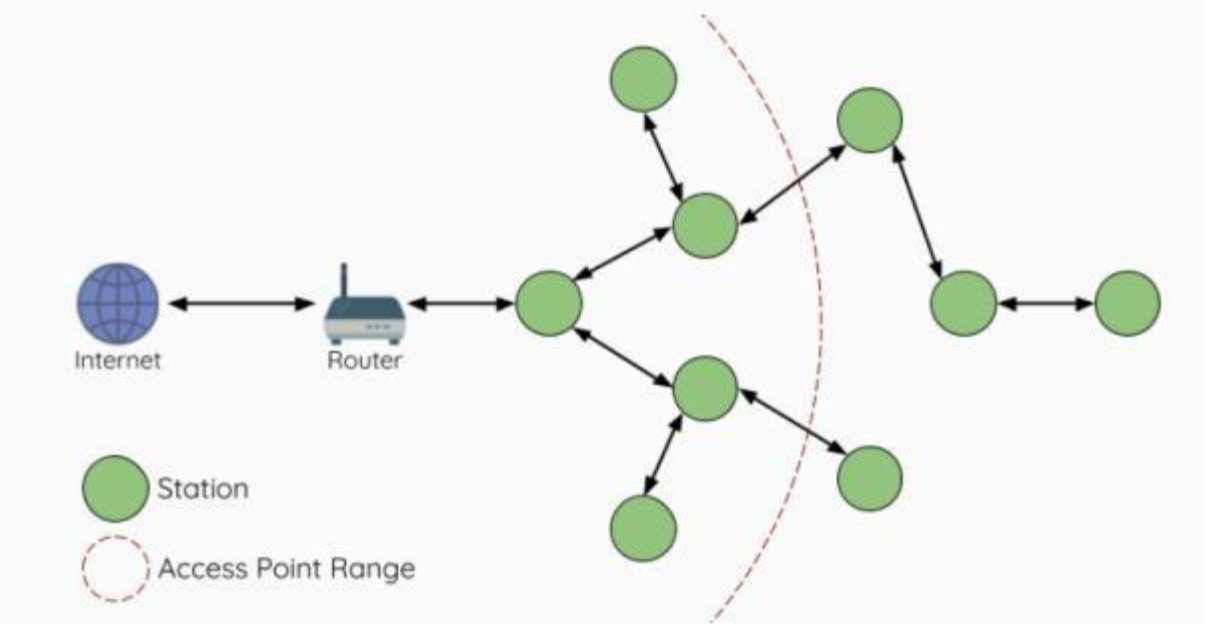

Рисунок 1.2 — Мережева архітектура ESP-MESH

1.2 Структура мережі ESP-MESH

ESP-MESH відрізняється від традиційних інфраструктурних мереж Wi-Fi тим, що вузли не потрібно підключатися до центрального вузла. Натомість вузлам дозволено підключатися до сусідніх вузлів. Вузли несуть взаємну відповідальність за ретрансляцію один одного. Це дозволяє ESP-MESH мережі мати набагато більшу зону покриття, оскільки вузли, як і раніше, можуть забезпечувати взаємодію без необхідності перебувати в межах досяжності центрального вузла. Аналогічним чином, ESP-MESH також менш схильний до перевантаження, оскільки кількість вузлів, дозволених у мережі,

більше не обмежується єдиним центральним вузлом.

Таблиця 1.1 — Мережева термінологія MESH-сетей

Найменування	Опис
Вузол (Node)	Будь-який пристрій, який належить або може належати мережі ESP-MESH
Кореневий вузол (Root Node)	Верхній вузол у мережі
Батьківський вузол (Parent Node)	Зворотня концепція дочірніх вузлів
Дочірній вузол (Child Node)	Будь-який вузол, який можна досягти шляхом повторення від батьківського вузла до дочірнього вузла.
Споріднений вузол (Descendant Node)	Вузли з тим самим батьківським вузлом
Підсіти(Subnetwork)	Підмережа – це підрозділ мережі ESP-MESH, який складається з вузла та всіх його нащадків. Таким чином, підмережа кореневого вузла складається зі всіх вузлів мережі ESP-MESH.
MAC-адреса (MAC Address)	Адреса керування доступом до середовища передачі.Використовується для унікальної ідентифікації кожного вузла чи маршрутизатора у мережі ESP-MESH.
Висхідне з'єднання (Upstream Connection)	З'єднання від вузла до його батьківського вузла
Східне з'єднання (Downstream Connection)	З'єднання вузла з одним із його дочірніх вузлів

ESP-MESH побудований на інфраструктурному протоколі Wi-Fi і може розглядатися як мережевий протокол, який поєднує безліч окремих мереж Wi-Fi в одну WLAN. Wi-Fi станція обмежена одним підключенням до AP у будь-який час (висхідне з'єднання), а AP може бути підключена до кількох станцій одночасно (низхідне з'єднання). Однак ESP-MESH дозволяє вузлу діяти одночасно як станція та точка доступу. Таким чином, вузол ESP-MESH може використовувати свій інтерфейс softAP для створення декількох низхідних підключень, а інтерфейс станції — для одного висхідного

з'єднання. Це, звісно, призводить до топології мережі як дерева з ієрархією батько-нащадок, що з кількох рівнів [2].

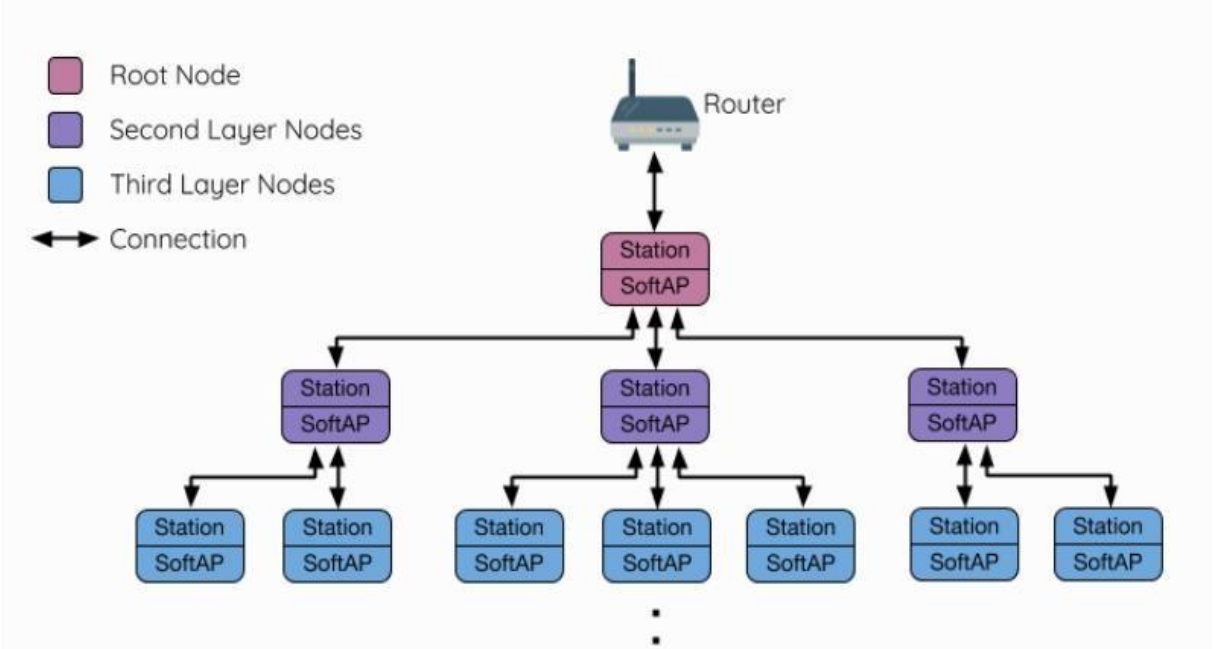

Рисунок 1.3 — Топологія дерева ESP-MESH

Кореневий вузол — кореневий вузол є верхнім вузлом у мережі та єдиним інтерфейсом між мережею ESP-MESH та зовнішньою IP-мережею. Кореневий вузол підключений до традиційного маршрутизатора Wi-Fi і ретранслює пакети даних у зовнішню IP-мережу або зовнішньої IP-мережі на вузли в мережі ESP-MESH. У мережі ESP-MESH може лише один кореневий вузол, а висхідне з'єднання кореневого вузла може бути підключено лише маршрутизатор [1].

Кінцевий вузол — це вузол, який не допускає жодних дочірніх вузлів (немає підключень до низхідного потоку). Отже, кінцеві вузли можуть надсилати або отримувати лише власні пакети даних, але не можуть пересилати пакети даних інших вузлів. Якщо вузол знаходиться на максимально допустимому рівні мережі, його буде призначено як кінцевий вузол.

Це не дозволяє вузлам формувати будь-які низхідні з'єднання, тим

самим гарантуючи, що до мережі не будуть додані додаткові рівні. Оскільки для будь-якого низхідного з'єднання потрібен інтерфейс softAP, деякі вузли (тільки станції), які не мають інтерфейсу softAP, також будуть призначені як кінцеві вузли [3].

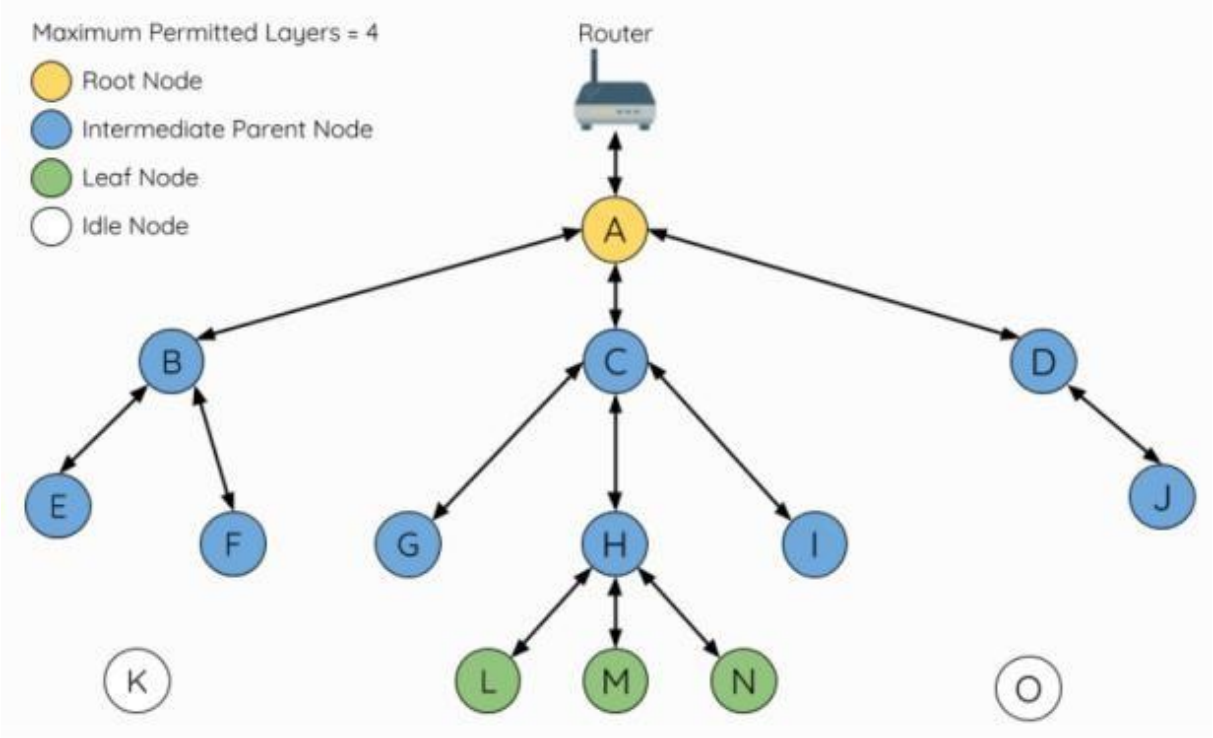

Рисунок 1.4 — Тип вузла ESP-MESH

Проміжний батьківський вузол — сполучний вузол, який не є ні кореневим вузлом, ні кінцевим вузлом є проміжним батьківським вузлом. Проміжний батьківський вузол повинен мати одне висхідне з'єднання (єдиний батьківський вузол), але може мати від нуля до багатьох низхідних з'єднань (від нуля до багатьох дочірніх вузлів). Отже, проміжний батьківський вузол може відправляти, так і приймати пакети даних, а також пересилати пакети даних, відправлені з його висхідних і низхідних з'єднань. Як показано на малюнку вище, вузли B до J є проміжними батьківськими вузлами. Проміжні батьківські вузли без низхідних підключень (наприклад, вузол E/F/G/I/J) не еквівалентні кінцевим вузлам, тому що їм, як і раніше,

буде дозволено формувати низхідні з'єднання в майбутньому.

Вільний вузол — вузли, які не приєдналися до мережі, призначаються як незайняті вузли. Неактивний вузол намагатиметься сформувати висхідне з'єднання з проміжним батьківським вузлом або спробувати стати кореневим вузлом за правильних обставин.

Кадр маяка та поріг RSSI [1]. Кожен вузол ESP-MESH, який може формувати низхідне з'єднання, буде періодично передавати кадри маяка Wi-Fi. Вузли використовують кадри маяка, щоб дозволити іншим вузлам виявляти його присутність та розуміти його статус. Неактивний вузол прослуховуватиме фрейм маяка, щоб згенерувати список потенційних батьківських вузлів, а недіючий вузол сформує висхідне з'єднання з одним з них.

Переважний батьківський вузол. Коли у незайнятого вузла є кілька кандидатів у батьківський вузол (потенційні батьківські вузли), недіючий вузол формуватиме висхідне з'єднання з кращим батьківським вузлом. Переважний батьківський вузол визначається відповідно до таких критеріїв:

— де є батьківський вузол-кандидат;

— кількість низхідних підключень (дочірніх вузлів), які належать батьківському вузлу-кандидату.

Вибір бажаного батьківського вузла завжди віддаватиме пріоритет кандидатам у батьківський вузол на найменшому рівні мережі (включаючи кореневий вузол). Це допомагає мінімізувати загальну кількість рівнів у мережі ESP-MESH при формуванні висхідних з'єднань. Наприклад, враховуючи вузол другого рівня та вузол третього рівня, вузол другого рівня завжди буде першим вибором [3].

Якщо в одному шарі є кілька кандидатів у батьківський вузол, першим буде обраний кандидат у батьківський вузол із найменшою кількістю дочірніх вузлів. Цей стандарт забезпечує балансування кількості низхідних з'єднань між вузлами одному рівні.

Таблиця маршрутизації. Кожен вузол у мережі ESP-MESH підтримуватиме свою власну таблицю маршрутизації, яка використовується для правильної маршрутизації пакетів ESP-MESH до правильного вузла призначення. Таблиця маршрутизації конкретного вузла буде містити MAC-адреси всіх вузлів у підмережі конкретного вузла (включаючи MAC-адреси самого конкретного вузла). Кожна таблиця маршрутизації внутрішньо розділена кілька подтаблиц, і кожна подтаблица відповідає підмережі кожного дочірнього вузла.

Якщо взяти як приклад малюнок 5, таблиця маршрутизації вузла B складатиметься з MAC-адрес вузлів від B до I (тобто підмережі, еквівалентної вузлу B). Таблиця маршрутизації вузла B внутрішньо розділена на дві підтаблиці, включаючи вузли від C до F і вузли від G до I (тобто еквівалентні підмережам вузлів C і G відповідно).

ESP-MESH використовує таблицю маршрутизації, щоб визначити, чи слід пересилати пакети ESP-MESH висхідним або низхідним потоком відповідно до таких правил.

Якщо MAC-адреса пункту призначення повідомлення знаходиться в таблиці маршрутизації поточного вузла і не є поточним вузлом, вибирається підтаблиця, що містить MAC-адресу пункту призначення, та пакет даних пересилається у низхідному напрямку на відповідний суб-вузол. вузол підтаблиці.

Якщо MAC-адреса відсутня в таблиці маршрутизації поточного вузла, пакет даних пересилається висхідним потоком до батьківського вузла поточного вузла. Повторення цього призведе до того, що пакет даних досягне кореневого вузла, де таблиця маршрутизації повинна містити всі вузли. мережі.

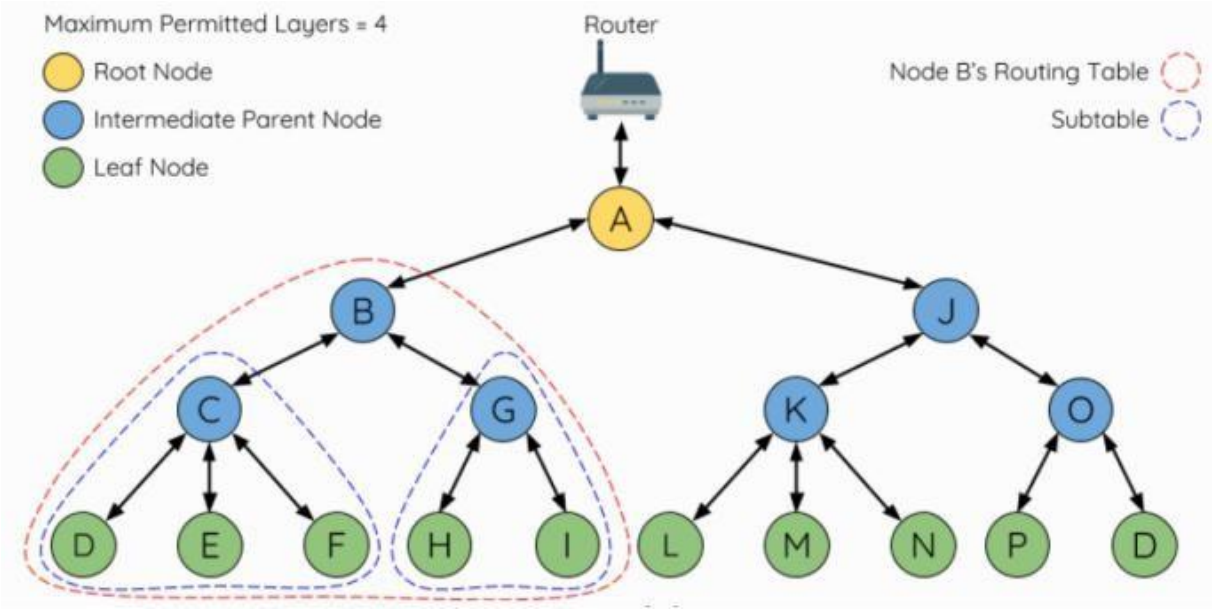

Рисунок 1.5 — Приклад таблиці маршрутизації ESP-MESH

1.2 Процес доступу до мережі ESP Mesh

Перед тим, як розпочнеться процес побудови мережі ESP-MESH, певні частини конфігурації мають бути узгоджені кожному вузлі мережі. Кожен вузол має бути налаштований з тим самим ідентифікатором Mesh Network ID, конфігурацією маршрутизатора і конфігурацією softAP [1].

Процес побудови мережі ESP-MESH включає вибір кореневого вузла, а потім формування низхідних підключень шар за рівнем, поки всі вузли не приєднаються до мережі. Процес побудови мережі ESP-MESH можна резюмувати так, як показано на рисунку 1.6.

Кореневий вузол може бути вказаний під час налаштування або динамічно вибраний залежно від рівня сигналу між кожним вузлом та маршрутизатором. Після вибору кореневий вузол підключиться до маршрутизатора та почне створювати низхідні з'єднання. Посилаючись на наведений вище малюнок, вузол A обраний кореневим вузлом, тому вузол A формує висхідне з'єднання з маршрутизатором.

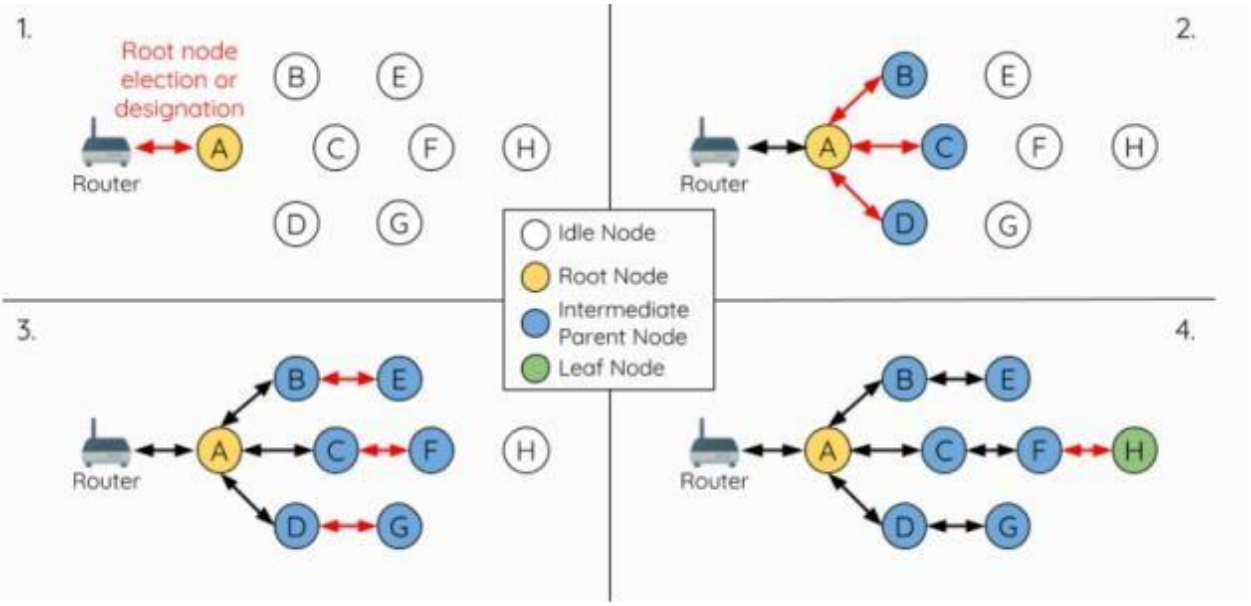

Рисунок 1.6 — Процес побудови мережі ESP-MESH

Після підключення кореневого вузла до маршрутизатора вільні вузли в межах діапазону кореневого вузла почнуть з'єднуватися з кореневим вузлом, таким чином утворюючи другий рівень мережі. Після підключення вузол другого рівня стає проміжним батьківським вузлом (за умови, що максимально допустима кількість шарів > 2) і таким чином утворює наступний рівень. Зверніться до малюнку вище, вузли від B до D знаходяться в межах дії кореневого вузла. Отже, вузли від B до D утворюють висхідну сполуку з кореневим вузлом і стають проміжним батьківським вузлом.

Незайняті вузли, що залишилися, будуть пов'язані з проміжними батьківськими вузлами в діапазоні, таким чином, утворюючи новий рівень в мережі. Після підключення недіючий вузол стане проміжним батьківським вузлом або листовим вузлом, залежно від максимально допустимого рівня мережі. Повторюйте цей крок, поки в мережі не залишиться вузлів, що не використовуються, або поки не буде досягнуто максимально допустимого рівня мережі. Посилаючись на малюнок вище, вузли E/F/G підключаються до вузлів B/C/D відповідно і самі стають проміжними батьківськими вузлами.

Щоб мережа не перевищувала максимально допустиму кількість шарів,

вузли на найбільшому рівні автоматично стають листовими вузлами після їхнього підключення. Це запобігає з'єднанню будь-яких інших недіючих вузлів з листовими вузлами, тим самим запобігаючи утворенню нових шарів. Однак, якщо у незайнятого вузла немає інших потенційних батьківських вузлів, він залишиться бездіяльним невизначено довго. На рис. 1.6 представлений приклад побудови мережі ESP-MESH з максимально допустимою кількістю рівнів мережі, що дорівнює чотирьом. Отже, коли вузол H підключений, він стає кінцевим вузлом, щоб запобігти формуванню будь-яких низхідних підключень.

2 ПРОЕКТУВАННЯ РОЗПОДІЛЕНОЇ МЕРЕЖІ ДЛЯ АВТОМАТИЧНОГО УПРАВЛІННЯ ДОСТУПОМ

2.1 Принцип роботи системи

Щоб реалізувати систему доступу, як показано на рисунку 2.1, поділимовсю систему на три частини для розробки:

— реалізація мережі ESP32-WiFi-Mesh;

— розробка системи керування користувачами;

— хмарний сервер.

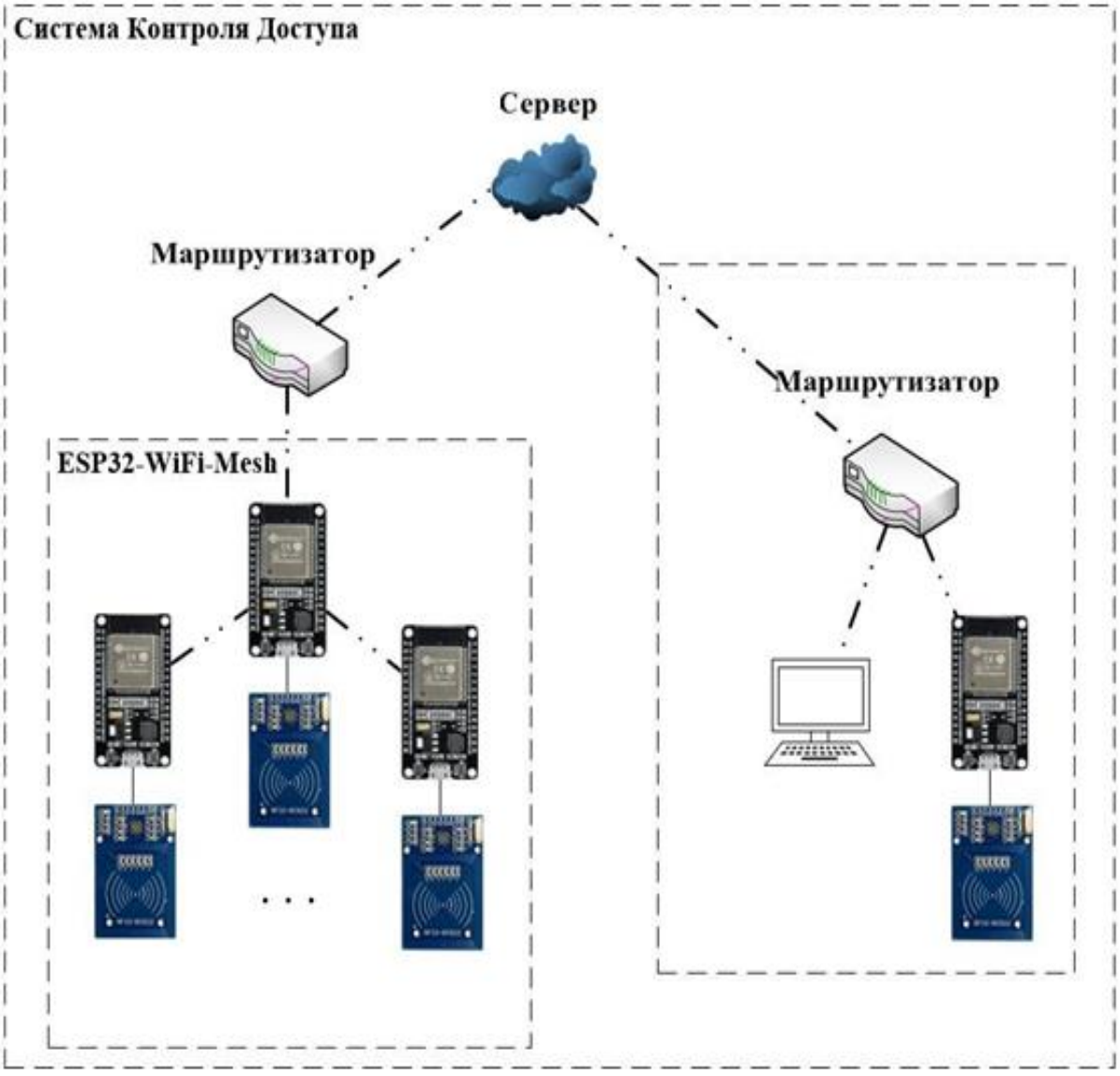

Рисунок 2.1 — Структура системи контролю доступу

2.1.1 Мережа ESP32-WiFi-Mesh

У мережі ESP32-WiFi-Mesh кожен мікроконтролер ESP32 підключається один до одного для формування пористої мережі. Вони вибирають кореневий вузол у відповідності до рівня сигналу маршрутизатора, а потім кореневий вузол підключається до маршрутизатора, щоб вся пориста мережа може бути підключена до Інтернету. Коли кореневий вузол виходить із ладу або коли мережа починає працювати нестійко, вони переобирають кореневий вузол для самоорганізації мережі. Коли користувач використовує картку керування доступом для отримання доступу до пристрою, контролер повинен визначити права доступу цього користувача до об'єкта. Для цього, якщо пристрій є дочірнім вузлом мережі, воно передає інформацію про доступ на кореневий вузол, а кореневий вузол використовує протокол HTTP для передачі інформації про доступ на сервер за допомогою методу GET, а потім сервер обробляє інформацію про доступ і відправляє результат. Далі відбувається передача отриманої інформації пристрою, який ініціював передачу. Якщо пристрій, до якого здійснюється доступ, є кореневим вузлом, ви можете надіслати запит HTTP безпосередньо, щоб отримати результат зворотного зв'язку HTTP.

2.1.2 Система управління користувачами

У системі управління користувачами розроблено веб-сторінку для перевірки ідентифікаційної інформації адміністратора та управління інформацією про користувачів (на даний момент доступні тільки функції реєстрації та видалення), при її використанні мікроконтролеру ESP32 і термінальному пристрою необхідно бути в тій же локальній мережі. Доступ до веб-сторінки здійснюється шляхом доступу до IP-адреси мікроконтролера через локальну мережу.

2.1.3 Хмарний сервер

На сервері хмар використано мову PHP і базу даних MySQL для розробки внутрішньої програми обробки запитів доступу користувачів до об'єктів. У програмі обробляємо дані з мережі ESP32-WiFi-Mesh за протоколом HTTP та дані із системи управління користувачами та формуємо відповідь. Програма формує відповідь на основі вмісту бази даних на сервері. Також підтримуються функції додавання, зміни, видалення та вибірки.

2.2 Введення та параметри апаратного обладнання

RFID (радіочастотна ідентифікація) – радіочастотна ідентифікація У роботі також використовуємо модуль RFID-RC522, який використовується для зчитування інформації пропускної карти у системі контролю доступу. Принцип його роботи наступний: RFID складається з 2 частин:

— транспондера/мітки, який прикріплюється до об'єкта;

— бездротового приймача, який використовується для зчитування даних з транспондера/мітки.

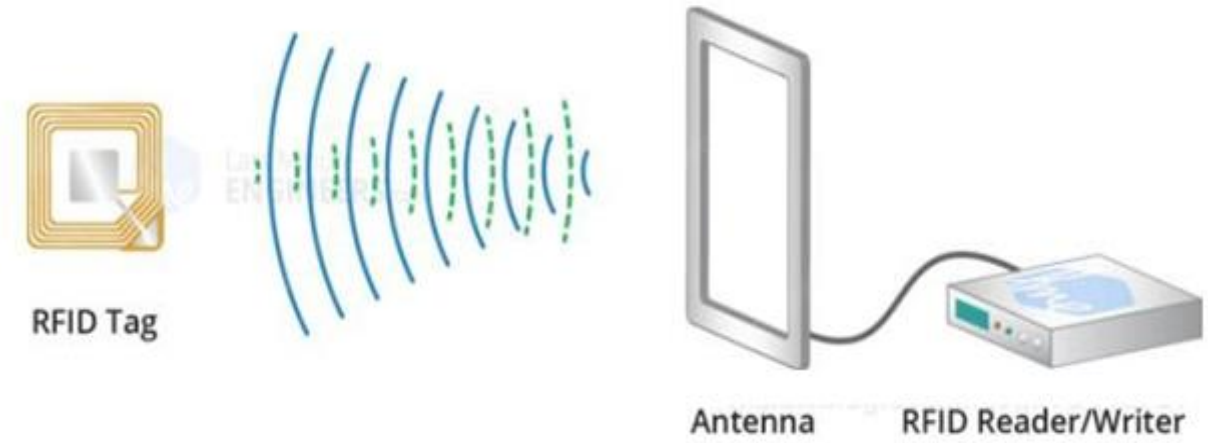

Рисунок 2.2 — Принцип роботи схеми RFID

Використовуємо RFID-пристрій RFID-RC522, який обмінюється даними з чіпом через послідовний периферійний інтерфейс SPI (Serial

Peripheral Interface) зі швидкістю 10 Мбіт/с, а також підтримує протоколи I2C та UART. У цьому модулі є вивід переривання. Робоча напруга цього модуля становить від 2,5 до 3,3 В. Нижче наводиться опис кожного виводу, всього їх 8.

Таблиця 2.1 — Призначення виводів RFID-RC522

Найменування виводу	Опис
VCC	Для підключення до 2,5-3,3 В, якщо підключаєтеся до інтерфейсу 5В, цей модуль RC522 може вийти з ладу;
RST	Вхідний вивід скидання та точки зупинки. RC522 відключається на низьких частотах, включаючи генератор, вхідні контакти та відключення послідовного периферійного інтерфейсу;
MOSI (Master Out Slave In)	Цей висновок є SPI (послідовним периферійним інтерфейсом) цього модуля.
SCK	Отримання імпульсного сигналу SPI.
GND	Земля
SS/SDA/Rx	Під час запуску SPI цей вивід є вхідним сигналом. Коли це інтерфейс протоколу I2C, це порт даних послідовного порту, а коли це UART, це вхідний порт послідовного порту.
IRQ	Вивід попередження про переривання. Коли мітка RFID знаходиться поруч із пристроєм, вона запускається цим виводом.
MISO/SCL/Tx	Цей інтерфейс (Master In Slave Out), коли включений SPI (послідовний периферійний інтерфейс). При використанні інтерфейсу протоколу I2C цей вивід є годинником послідовного порту, під час використання інтерфейсу протоколу UART цей вивід є портом виведення послідовних даних.

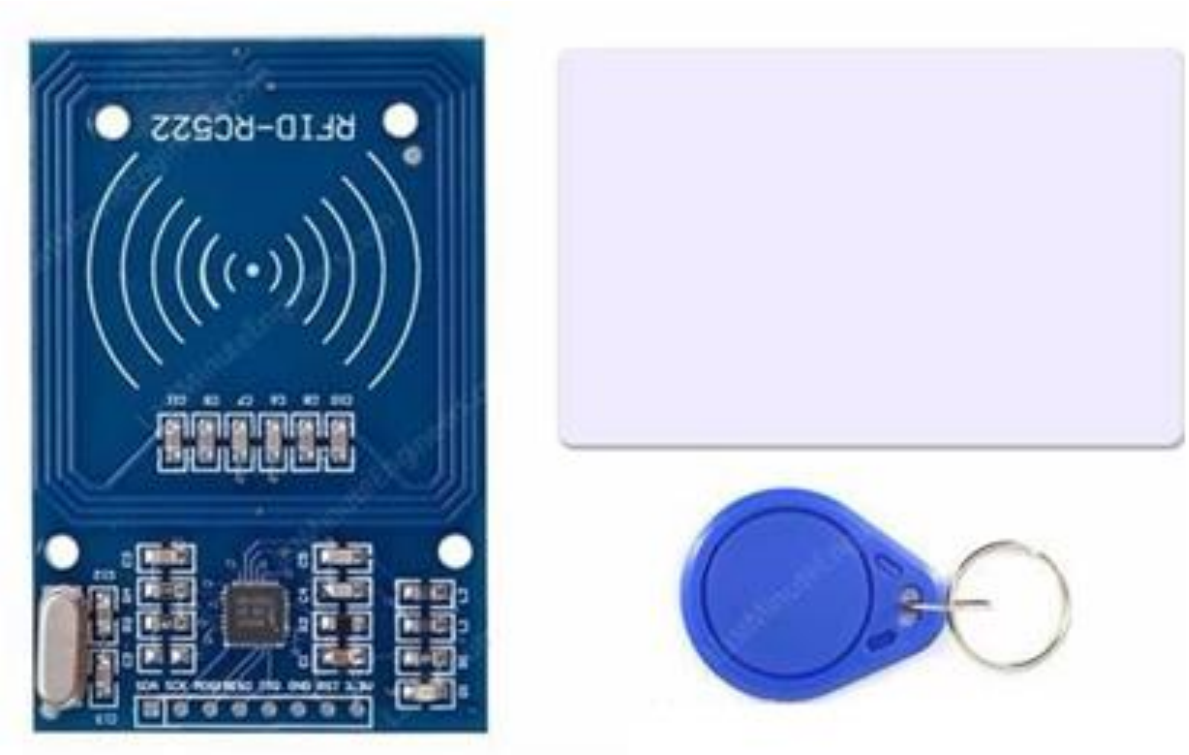

Рисунок 2.3 — Зчитувач RFID-RC522

ESP32-WROOM-32 — мініатюрний високопродуктивний суміщений Wi-Fi + BT + BLE модуль від компанії Espressif, розроблений для широкого спектру застосувань, починаючи від малопотужних мережних датчиків до найскладніших додатків, таких як кодування голосу, потокова передача музики та MP3 кодування .

ESP32-WROOM-32 виконаний на базі популярного двоядерного чіпсету ESP32, з регульованою частотою від 80 МГц до 240 МГц, можливістю індивідуального управління та живлення.

Модуль розроблений для переносної та автономної електроніки та додатків інтернет-речей, виконаний у мініатюрному корпусі 25,5 мм х 18 мм, має на борту Flash пам'ять, кварц 40 МГц та PCB антену, що забезпечує відмінні RF характеристики. ESP32-WROOM має багату периферію, що включає такі інтерфейси як UART, SPI, I²C, I²S, роз'єм для SD карти, інфрачервоний порт, інтерфейс для підключення ємнісної сенсорної панелі. Однією з особливістю модуля є наднизьке споживання і гнучкий

вибір режимів, що дозволяє отримати цифри до 20мкА (deep sleep mode). Модуль підтримує весь стек протоколів стандартів WiFi 802.11n та BT4.2, забезпечуючи цей функціонал через інтерфейси SPI/SDIO або I²C/UART.

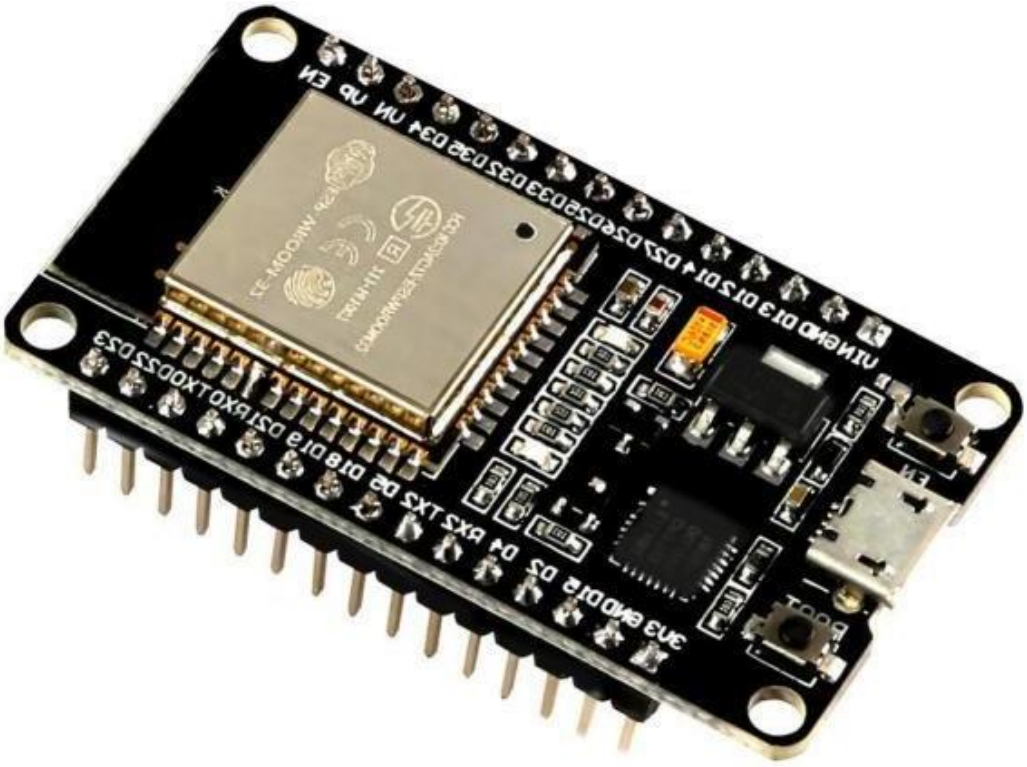

Рисунок 2.4 — Мікроконтролер ESP32

Відмінні особливості:

— підтримка dual mode Bluetooth: "classic" та BLE;

— швидкість Wi-Fi: 802.11 b/g/n до 150 Мбіт/с;

— підтримка режимів Wi-Fi: клієнт, точка доступу, Sniffer, Wi-Fi Direct;

— мінімальна чутливість -98 dBm;

— широкий діапазон робочих температур: -40 ° C ... +125 ° C;

— енергоспоживання до 20мкА (deep sleep mode);

— бездротове оновлення ПЗ;

— можливість підключення 4 x 16MB зовнішньої QSPI Flash та SRAM.

Технічні характеристики:

— протоколи: 802.11 b/g/n/d/e/i/k/r (802.11n до 150 Мбіт/с);

— A-MPDU та A-MSDU, підтримка захисного інтервалу в 0.4 сек;

— частотний діапазон: ГГц 2.4 ~ 2.5;

— Bluetooth Протоколи: Bluetooth v4.2 BR/EDR та BLE specification;

— радіо NZIF приймач з чутливістю: -98 dBm;

— передавач: Class-1, Class-2 та Class-3 AFH.

— аудіо: CVSD та SBC.

— апаратні засоби та інтерфейси: SD, UART, SPI, SDIO, I²C, LED PWM, Motor PWM, I²S, I²C, IR.

— GPIO, сенсорний датчик, ADC, DAC, LNA підсилювач.

— датчики на борту: Hall sensor, температурний датчик.

— генератори: кварцовий 26 МГц та 32 кГц.

— живлення: 2.2 ~ 3.6 В.

— робочий струм, мА середній: 80.

— діапазон робочих температур: -40 ° C ~ 85 ° C.

— програмне забезпечення: режими Wi-Fi Station/softAP/SoftAP+station/P2P.

— захист: WPA/WPA2/WPA2-Enterprise/WPS.

— шифрування: AES/RSA/ECC/SHA.

— оновлення ПЗ: UART Download / OTA (по мережі) / download and write firmware via host.

— розробка програм: Cloud Server Development/SDK for custom firmware development.

— мережеві протоколи: IPv4, IPv6, SSL, TCP/UDP/HTTP/FTP/MQTT.

— налаштування користувача: AT instruction set, cloud server, Android/iOS App.

2.3 Програмна реалізація

Для програмної реалізації системи контролю доступу декомпозуємо її на кілька завдань, а саме:

реалізація мережі WiFi-Mesh;

— система управління користувачами;

— внутрішня програма хмарного сервера.

2.3.1 Мережа WiFi-Mesh

Інтегроване середовище розробки (IDE): Microsoft Visual Studio Інструменти розробки: ESP-IDF

ESP-IDF — це офіційна платформа розробки IoT від Espressif для SoC серій ESP32, ESP32-S та ESP32-C. Він надає самодостатній SDK для розробки будь-яких універсальних програм на цих платформах з використанням мов програмування, як C і C++.

Скомпільована мова: C

Важливі бібліотечні файли та їх роль:

— "esp_mesh.h" — ця бібліотека містить все необхідне для реалізації мережі ESP-Mes;

— "esp_http_client.h" — esp_http_client надає API для надсилання запитів HTTP/S із програм ESP-IDF;

— "Freertos.h" — FreeRTOS — Це багатозадачна операційна система реального часу (ОСРВ) для вбудованих систем. Freertos.h надає API для використання функції FreeRTOS;

— "rc522.h" — Бібліотека C для взаємодії ESP32 зі зчитувачем RFID-карток MFRC522.

Повний листинг програми наведено у додатку Б. Оскільки загальний код дуже довгий, всі методи із символами "..." у коді не вказано, наведеному у додатку Б.

На основі цих бібліотечних файлів реалізували кілька основних функцій:

espread(void):

якщо це дочірній вузол, то читати повідомлення, призначені для себе в мережі, якщо це кореневий вузол, він викличе функцію task1().

espwirte(mess,toAddr):

mess — повідомлення для надсилання, toAddr — адреса пористої мережі пристрою призначення, ця функція реалізує функцію пористої мережі для відправки повідомлень.

tag_handler(sn):

sn-отримати адресу RFID-мітки, коли карта контролю доступу використовується на зчитувачі карт, інформація про карту буде зберігатися в sn, якщо пристрій є дочірнім вузлом, викличте функцію espwrite() для надсилання URL-адреси кореневого вузла. Якщо пристрій є кореневим вузлом, функція espwrite() буде викликатися з параметром toSelf, встановленим у true.

http_rest_with_url(url):

Створення http-клієнта для доступу до цільової URL-адреси та повернення інформації зворотного зв'язку http.

task1(par)):

par — це дані структури даних writeData. Ці дані містять URL-адресу http і адресу в мережі. Коли цей метод викликається, метод викликає метод http_rest_with_url (url), і після отримання зворотного зв'язку http, якщо параметр toSelf дорівнює false , відправити повідомлення атомарному вузлу, без обробки, якщо toSelf має значення true.

Після запуску програма спочатку виконає функцію ініціалізації app_main(), потім запустить функцію espwrite() через FreeRTOS. При доступі до пристрою буде запущено функцію tag_handler(). Логіка виконання програми така:

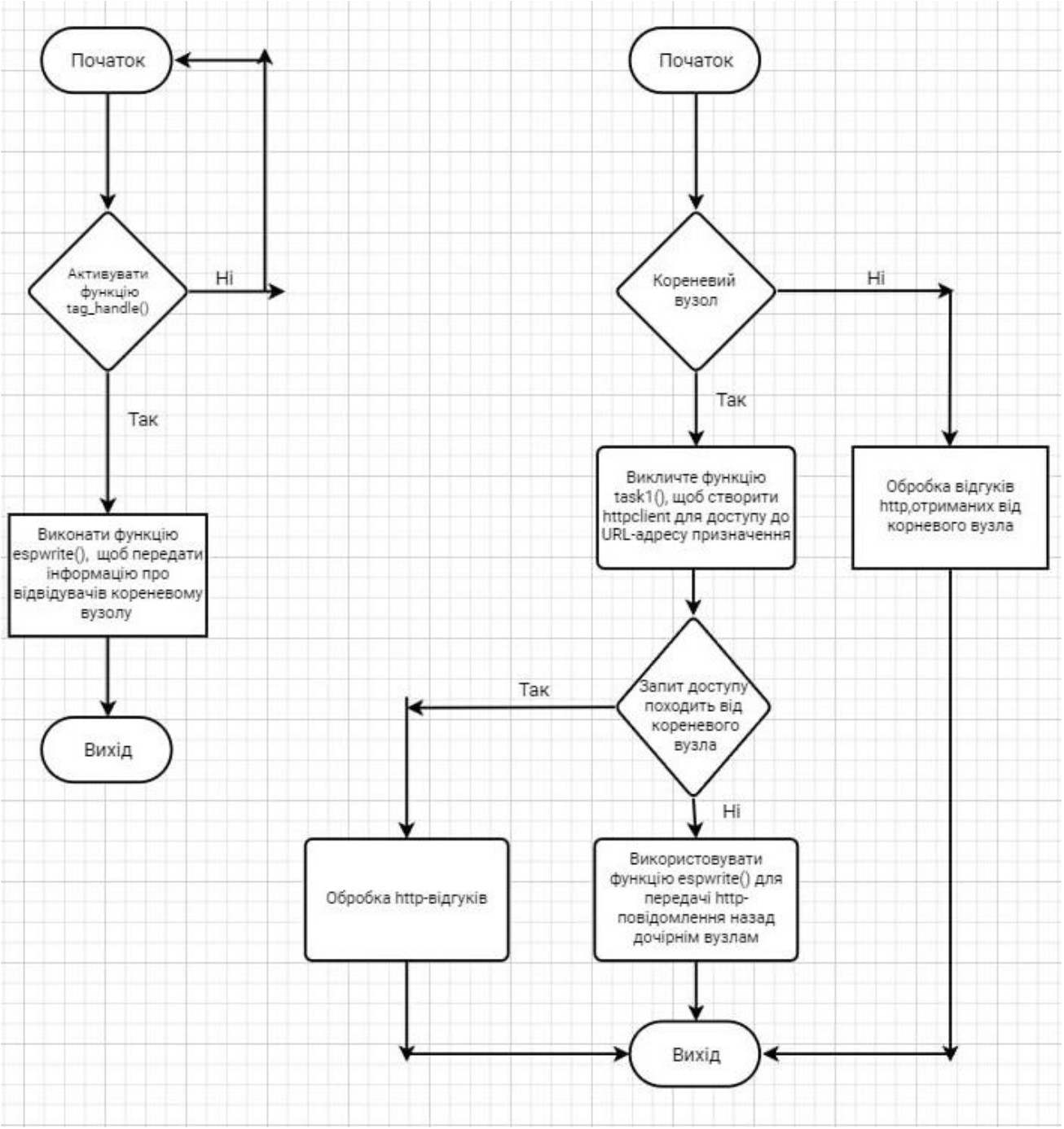

Рисунок 2.5 — Логічна схема роботи програми пористої мережі

2.3.2 Система управління користувачами

Інтегроване середовище розробки (IDE): Microsoft Visual Studio, Arduino. Скомпільована мова: HTML+CSS, javascript, C/C++.

Нижче наведено важливі бібліотечні файли, які використовуються в проекті.

Відмова від відповідальності: файли CSS у цьому проекті взяті з проектів з відкритим вихідним кодом:https://gitee.com/bilibili-ayang/login.

jQuery — це швидка, невелика та багатофункціональна бібліотека JavaScript. Вона значно спрощує такі речі, як обхід та маніпулювання документами HTML, обробку подій, анімацію та Ajax, за допомогою простого у використанні API, який працює у багатьох браузерах.

ESPAsyncWebServer.h — асинхронний сервер HTTP та WebSocket для ESP8266/ESP32/Arduino.

MFRC522.h — бібліотека для MFRC522 та інших модулів на базі RFID RC522. Дозволяє зчитувати та записувати різні типи карт радіочастотної ідентифікації (RFID) на контролері за допомогою зчитувача на базі RC522, підключеного через послідовний інтерфейс периферійного інтерфейсу (SPI).

Інтегроване середовище розробки (IDE):

— Microsoft Visual Studio: одне з найбільш потужних та універсальних середовищ розробки програмного забезпечення, Visual Studio надає розробникам широкий спектр інструментів для створення додатків на різних мовах програмування; це середовище дозволяє зручно керувати проектами, відстежувати помилки та оптимізувати код для досягнення найкращої продуктивності;

— Arduino: платформа для створення інтерактивних електронних пристроїв. Вона широко використовується для розробки прототипів і навчання електроніці, Arduino включає апаратне забезпечення (плати з мікроконтролерами) та програмне середовище (IDE), що робить розробку зручною та доступною навіть для початківців.

Далі слід розглянути скомпільовані мови, такі, як, HTML + CSS, JavaScript, C/C++.

- HTML + CSS: основні технології для створення веб-сторінок. HTML (HyperText Markup Language) відповідає за структуру веб-сторінок, тоді як CSS (Cascading Style Sheets) визначає їх зовнішній вигляд, використання цих

технологій дозволяє створювати красиві, зручні та адаптивні інтерфейси для користувачів;

- JavaScript: мова програмування, яка додає інтерактивності веб-сторінкам. З її допомогою можна реалізовувати динамічні елементи, обробляти події та виконувати складні обчислення безпосередньо в браузері користувача. JavaScript є невід'ємною частиною сучасного веб-розробки;

- C/C++: мови програмування, що використовуються для розробки системного та прикладного програмного забезпечення, вони забезпечують високу продуктивність і контроль над апаратними ресурсами, що є критично важливим для роботи з мікроконтролерами та іншими вбудованими системами.

Важливі бібліотечні файли, які використовуються в проекті.

jQuery — це швидка, невелика та багатофункціональна бібліотека JavaScript. Вона значно спрощує такі завдання, як обхід та маніпулювання документами HTML, обробка подій, анімація та Ajax. Використання jQuery дозволяє розробникам швидко та ефективно створювати динамічні веб-додатки, що працюють у багатьох браузерах. Завдяки простому у використанні API, розробники можуть писати менше коду, досягаючи тих самих результатів.

ESPAsyncWebServer.h: — асинхронний сервер HTTP та WebSocket для мікроконтролерів ESP8266/ESP32/Arduino. Ця бібліотека дозволяє створювати високопродуктивні веб-сервери, які можуть обробляти численні запити одночасно без блокування основного потоку. Це особливо важливо для вбудованих систем, де обмежені апаратні ресурси потребують ефективного управління обчислювальними задачами.

MFRC522.h — бібліотека для роботи з модулями на базі RFID RC522. Вона дозволяє зчитувати та записувати різні типи карт радіочастотної ідентифікації (RFID) за допомогою зчитувача, підключеного через послідовний інтерфейс SPI. Це відкриває широкі можливості для створення

систем контролю доступу, автоматизації та інших рішень, що використовують технологію RFID.

Відмова від відповідальності: файли CSS у цьому проекті взяті з проектів з відкритим вихідним кодом: [bilibili-ayang login](https://gitee.com/bilibili-ayang/login). Використання відкритих ресурсів дозволяє швидко інтегрувати готові рішення, зберігаючи при цьому час та ресурси на розробку власних стилів.

Система управління користувачами, описана в цьому документі, поєднує в собі різноманітні технології та бібліотеки для створення інтерактивних та ефективних веб-додатків. Використання перевірених бібліотек та інструментів забезпечує стабільність та високу продуктивність системи. Інтегроване середовище розробки, такі як Microsoft Visual Studio та Arduino, дозволяють легко керувати проектами, відстежувати помилки та оптимізувати код для досягнення найкращих результатів. Скомпільовані мови, такі як HTML, CSS, JavaScript, C та C++, забезпечують гнучкість та потужність при створенні як веб-інтерфейсів, так і системного програмного забезпечення. Використання бібліотек, таких як jQuery, ESPAsyncWebServer.h та MFRC522.h, дозволяє швидко реалізувати необхідну функціональність та забезпечити високу продуктивність та надійність системи.

2.3.3 Відображення сторінок керування користувачами

На веб-сторінці реалізували функцію входу адміністратора, і ми реалізували функцію реєстрації та видалення користувачів на сторінці управління користувачами, а інформацію про зчитування карток зчитувача RFID-карток можна отримувати в режимі реального часу на сторінці.

Нижче наведено інтерфейс входу адміністратора:

Рисунок 2.6 — Інтерфейс входу адміністратора

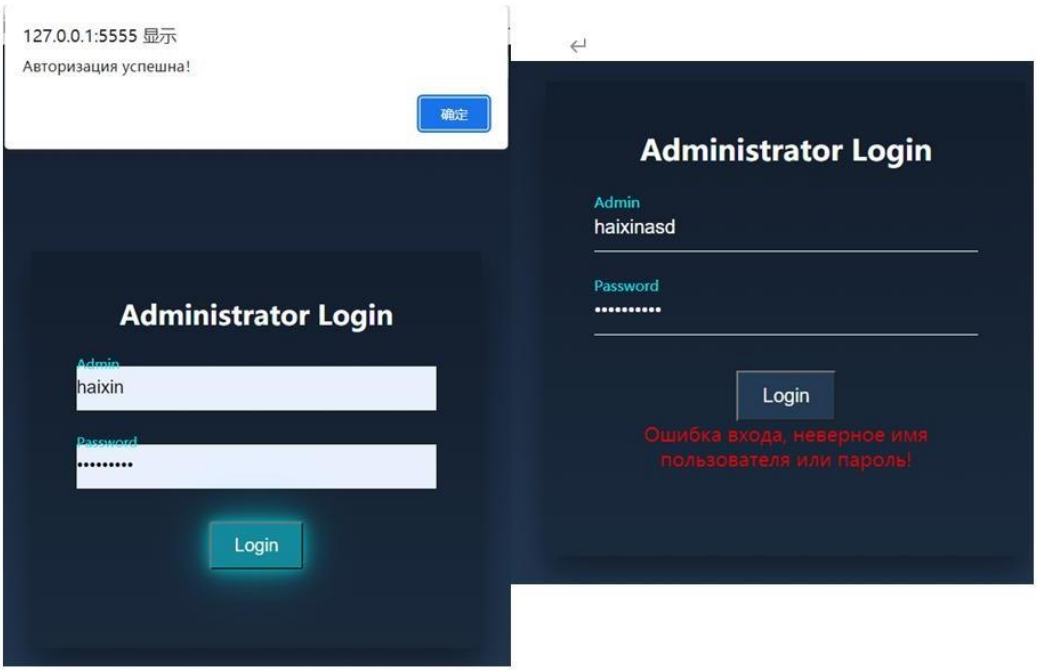

Рисунок 2.7 — Успішний вхід адміністратора та помилка входу

Після успішного входу потрапляєте в інтерфейс управління користувачами.

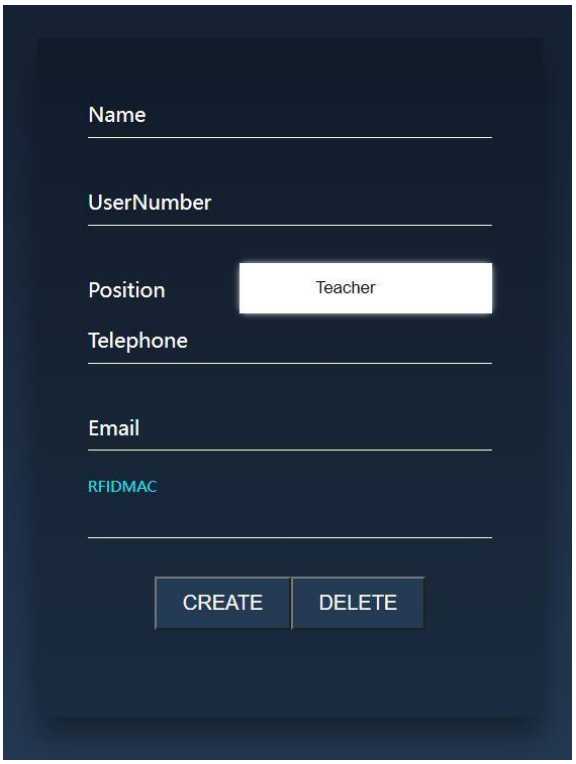

Рисунок 2.8 — Інтерфейс керування користувачами

Як видно на рисунку 2.8, коли адміністратор реєструє користувача, необхідно ввести ім'я користувача, ідентифікатор користувача, посаду, номер мобільного телефону та електронну адресу. Ця інформація про користувача буде прив'язана до RFID-картки користувача.

Варто відзначити, що реалізував функцію відображення інформації про карту, лічену пристроєм зчитування карт RFID, на веб-сторінці режимі реального часу, фактичний ефект роботи виглядає наступним чином:

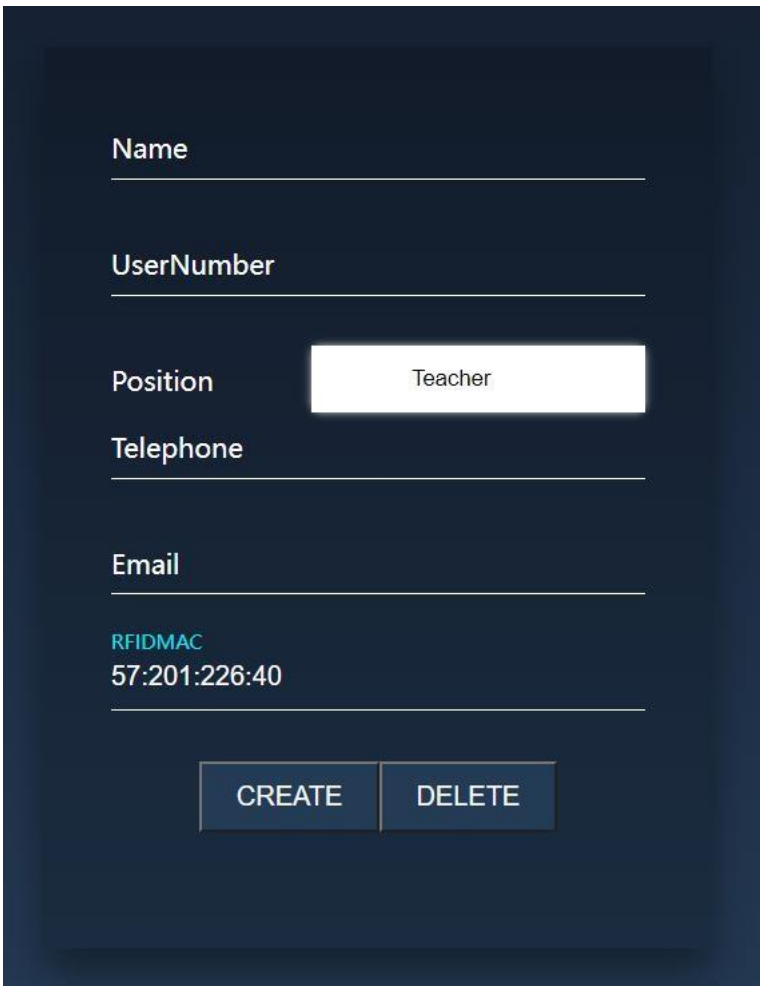

Рисунок 2.9 — Функція читання адреси у карті в режимі реального часу

Створено нову веб-сторінку в каталозі /rfidmac, створивши нову сторінку на AsyncWebServer ESP32. Ця веб-сторінка відображає інформацію про карту, що зчитується пристроєм RFID, в режимі реального часу, тому головна веб-сторінка може читати RFID з цієї веб-сторінки час від часу адресу карти.

```
server.on("/rfidmac", HTTP_GET, [](AsyncWebServerRequest *request){
  request->send_P(200, "text/plain", rfidmac.c_str());
});

server.begin();
```

Рисунок 2.10 — Лістинг створення веб-сторінки /rfidmac в esp32(Arduino)

```
setInterval(function () {
  var xhttp;
  if (window.XMLHttpRequest) {
    //  IE7+, Firefox, Chrome, Opera, Safari код выполнения браузера
    xhttp = new XMLHttpRequest();
  }
  else {
    // IE6, IE5 код выполнения браузера
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
  }

  xhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("RFIDMAC").value = this.responseText;
    }
  };
  xhttp.open("GET", "/rfidmac", true);
  xhttp.send();
}, 100);
```

Рисунок 2.11 — Лістинг створення основної веб-сторінки

Лістинг створення основної веб-сторінки зчитує інформацію про RFID-карту на веб-сторінці /rfidmac в режимі реального часу (javascript).

У керуванні користувачами реалізував дві функції створення користувача та видалення користувача. У функції створення користувача, після збирання інформації про користувача та підтвердження правильності формату, браузер буде використовувати метод HTTP POST для передачі інформації на сервер хмар. Отримані хмарним сервером дані будуть оброблені відповідним чином, а потім результати обробки будуть повернуті до браузера. В даний час є три результати (1 — успішне створення, 2 — створення не виконано, 3 — виникла проблема з сервером). У функції видалення користувача потрібно лише надати інформацію про карту RFID. Після надсилання на хмарний сервер хмарний сервер видаляє дані, пов'язані з картою. Є три результати (1 — видалення успішно, 2 — помилка видалення, 3 — проблема із сервером).

Варто зазначити, що при використанні цього пристрою потрібен комп'ютер, який може отримати доступ до браузера, і два пристрої повинні бути підключені до одного і того ж маршрутизатора, а браузер отримує

доступ до веб-сторінки через IP-адресу ESP32.

2.4 Створення хмарних серверів та серверні програми

Сервер, який використовується у розробці —сервер додатків Tencent Cloud, сервер знаходиться в Сінгапурі, операційна система сервера — Linux, його встановлено на сервер HTTP сервер Apache, розроблено програму фонової обробки на PHP і використовував базу даних MySQL для зберігання даних.

Apache HTTP-сервер — вільний веб-сервер. Apache є кросплатформним програмним забезпеченням, підтримує операційні системи Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS. Основними перевагами Apache вважаються надійність та гнучкість конфігурації.

PHP — скриптова мова загального призначення, що інтенсивно використовується для розробки веб-додатків. В даний час підтримується переважною більшістю хостинг-провайдерів є одним з лідерів серед мов, які використовуються для створення динамічних веб-сайтів.

MySQL — вільна реляційна система управління базами даних. Я подав заявку на доменне ім'я для сайту:dongbtest.site

2.5 Проектування бази даних

У базі даних ”acs” ми створили чотири таблиці, а саме: user, administrator, accessrecord і room, які структури даних і взаємозв'язку показані рисунку 2.12.

У таблиці user зберігаються ім'я користувача, повноваження, контактна інформація та інформація про утримувану RFID-картку. У таблиці room зберігається ідентифікатор кімнати, ім'я кімнати та рівень дозволів, потрібний для доступу до кімнати. Таблиця accessrecord використовується для зберігання інформації про доступ користувача до кімнати в певний час. administrator — це незалежна таблиця, яка використовується для зберігання

облікового запису адміністратора та пароля для подальшої перевірки особистості адміністратора.

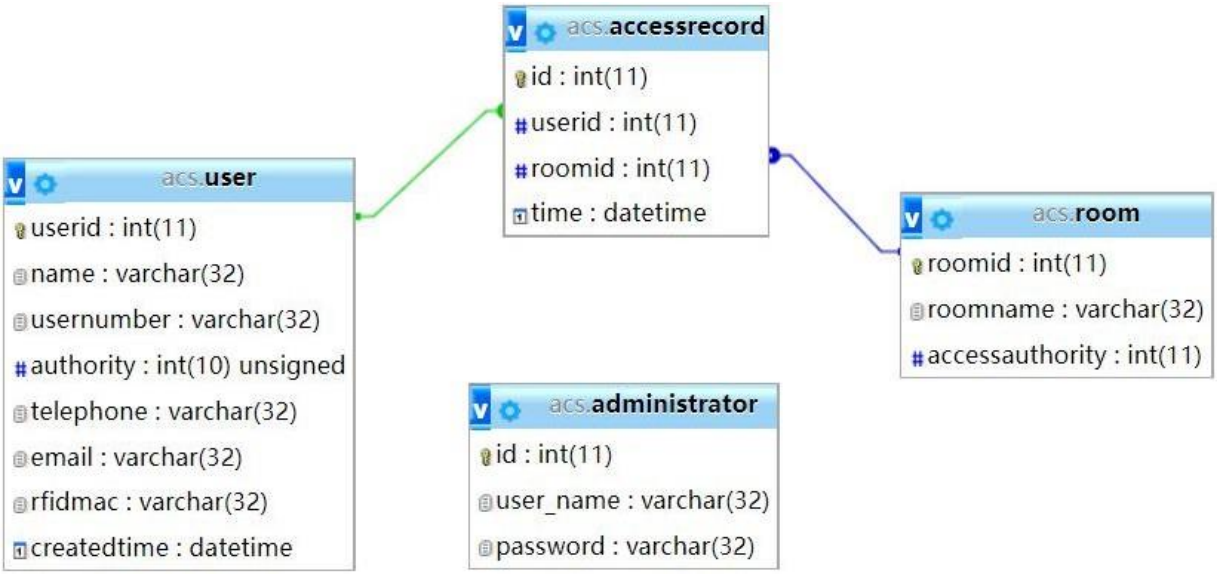

Рисунок 2.12 — Структура бази даних

2.6 Розробка обробника доступу

Програма PHP розміщена в Додатку В. У програмі використовуємо розширення MySQL для роботи з базою даних MySQL.

Розширення MySQLi — це драйвер реляційної бази даних, що використовується в мові PHP-сценарів для забезпечення інтерфейсу з базами даних MySQL.

Для підключення до бази даних MySQL був використаний наступний метод:

mysqli_connect(dbhost, dbuser, dbpass); де dbhost — ім'я сервера;

dbuser — ім'я користувача dbpass — пароль;

PHP використовує функцію mysqli_query для виконання команд бази MySQL;

mysqli_query(connection,query,resultmode);

deconnection — Необхідний параметр. Він вказує використовуване з'єднання MySQL;

Query — SQL запит для виконання.

Resultmode — необов'язковий параметр, константа, може бути будь-яким із наступних значень;

MYSQLI_USE_RESULT (використовується, якщо необхідно отримати багато даних);

MYSQLI_STORE_RESULT (за замовчуванням).

У програмі PHP реалізовано три режими: перший режим — режим входу адміністратора. Програма PHP реалізує доступ до бази даних, перевіряє правильність облікового запису та пароля та видає результати зворотного зв'язку. Другий режим — режим створення інформації про користувача, у програмі реалізовано функцію додавання інформації про користувача до бази даних. Третій — режим керування доступом, програма перевіряє, чи існує користувач у бібліотеці, та оцінює, чи може користувач отримати доступ до кімнати призначення, і дає зворотний зв'язок.

Якщо відповідна операція успішно пройшла, сервер повертає результат 1, якщо операція не вдалася, він повертає результат 0, а якщо виникла проблема з підключенням до бази даних, він повертає 2.

3 РЕЗУЛЬТАТИ ПРОЕКТУВАННАЯ ТА ЇХ ТЕСТУВАННЯ

3.1 Початок роботи

При показі програми почнемо з системи управління користувачами. Вікно введення логіна адміністратора показано на рисунку 3.1.

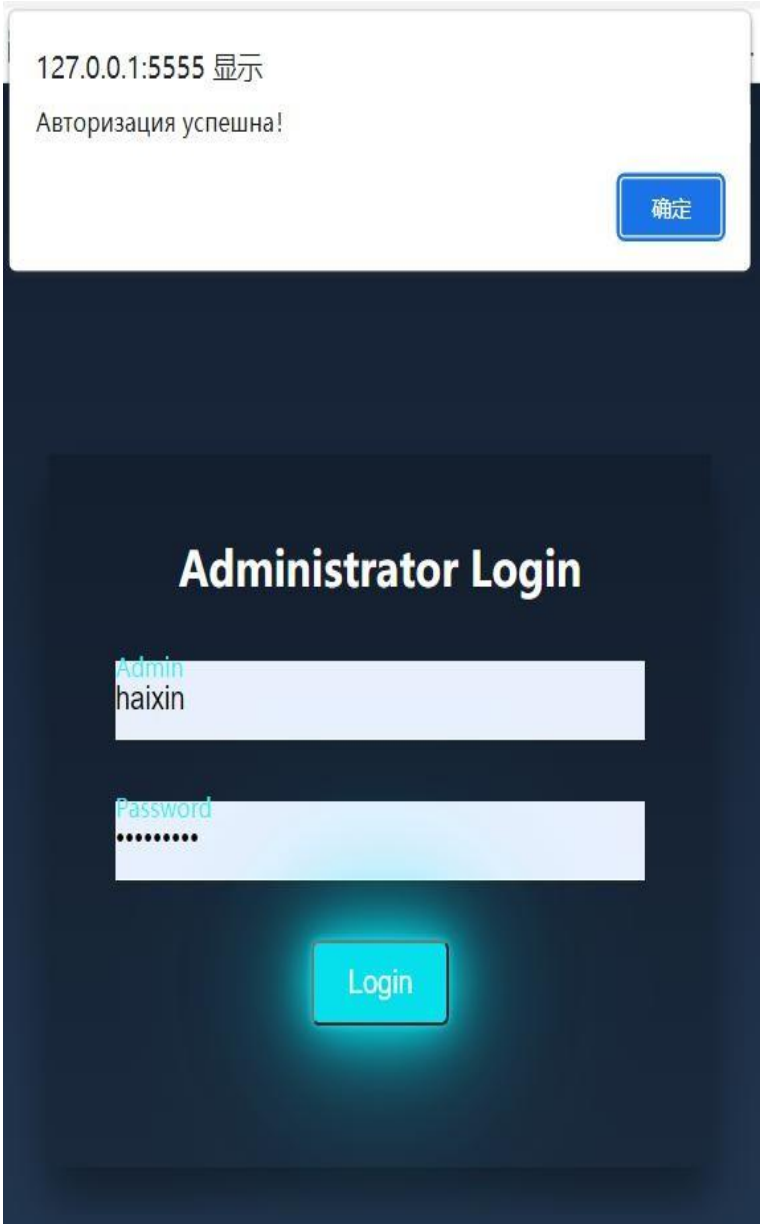

Рисунок 3.1 — Вхід адміністратора

Після успішного входу ви потрапите на сторінку управління користувачами, як показано на рисунку 3.1, ми покажемо процес створення користувача.

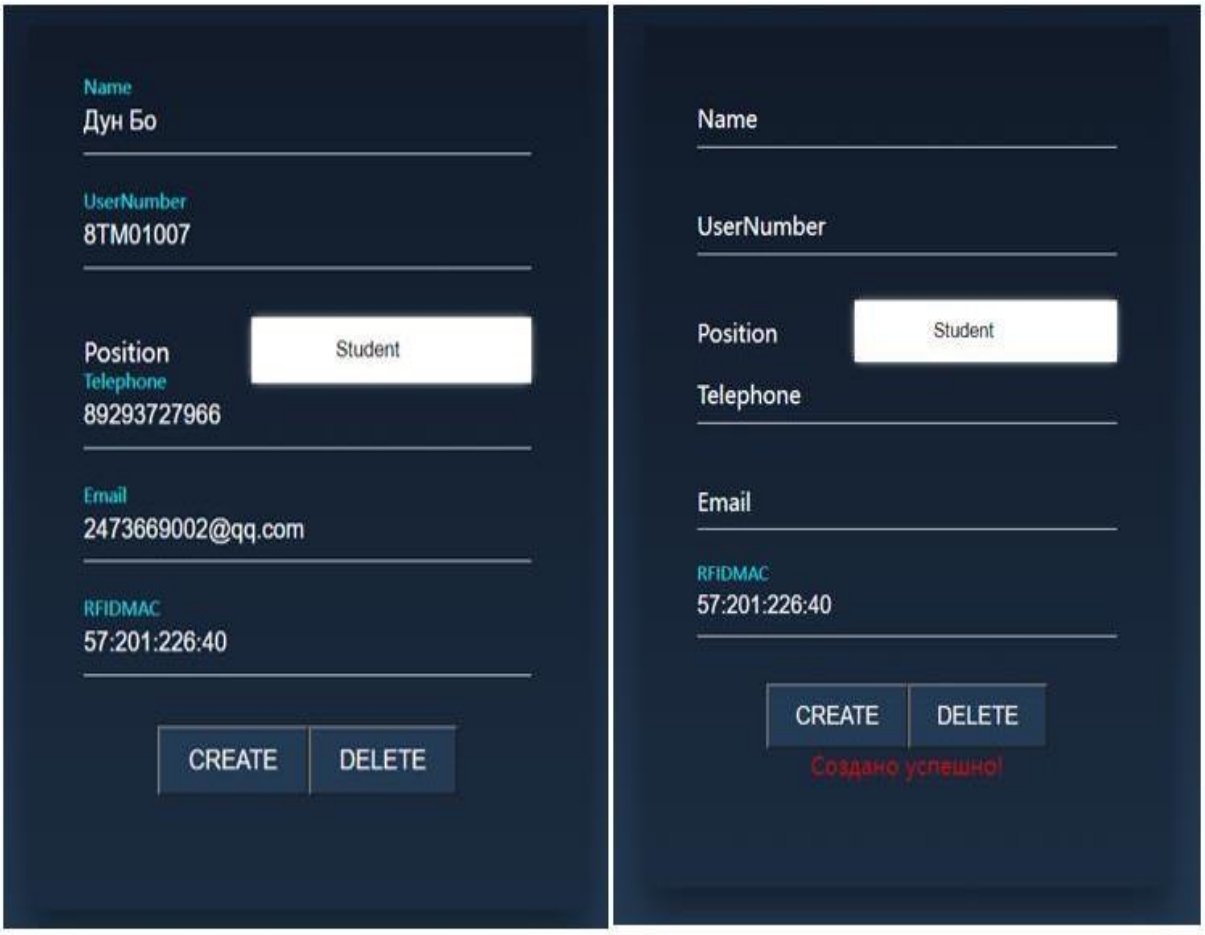

Рисунок 3.2 — Користувач успішно створено

Дані користувача в MYSQL показані на рисунку 3.3.

	userid	name	usernumber	authority	telephone	email	rfidmac	createdtime
编辑 复制 删除	12	Дун Бо	8TM01007	5	89293727966	2473669002@qq.com	57:201:226:40	2022-05-25 05:00:53

Рисунок 3.3 — Дані користувача в базі

При видаленні інформації вам потрібно лише надати інформацію про RFID-карту зчитувачу карток, і після натискання кнопки видалення відповідна інформація про користувача буде видалена.

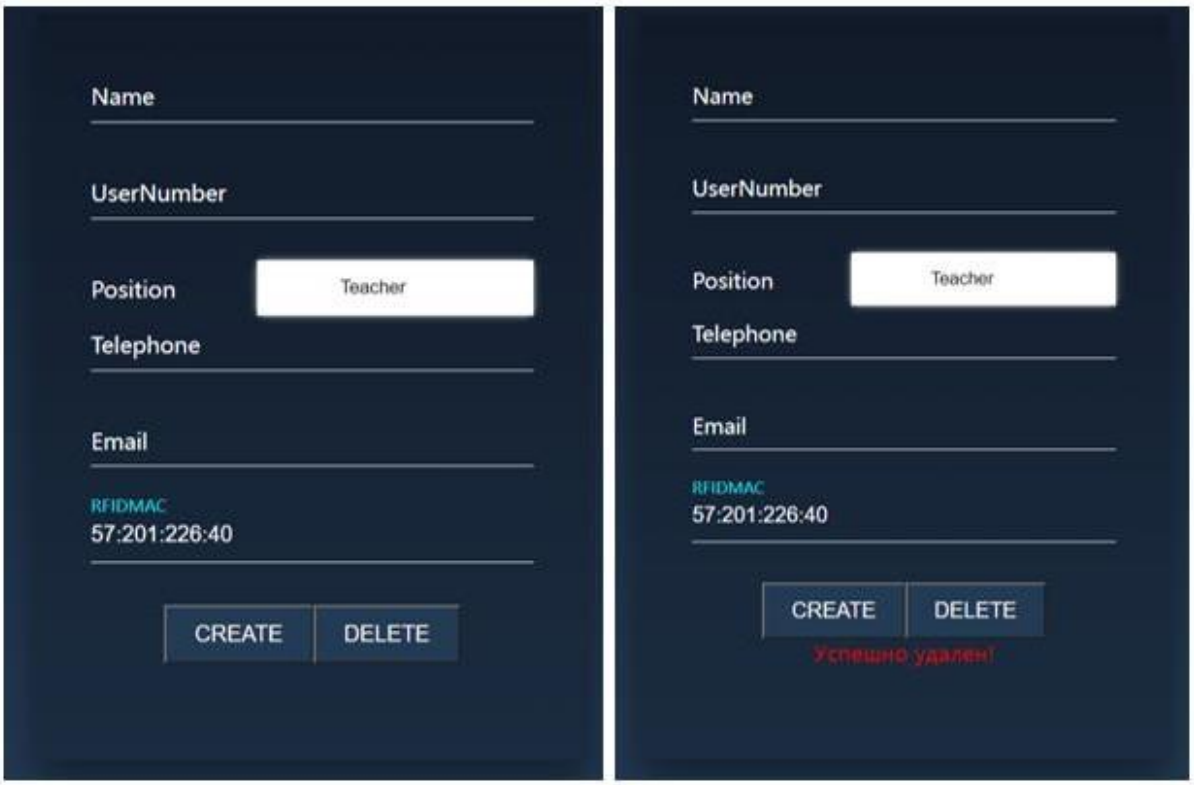

Рисунок 3.4 — Демонстрація функції видалення

3.2 ESP32 Мережа Wi-Fi Mesh

Для тесту ми використовували дві плати розробки ESP32 для формування невеликої мережі. Коли вони були підключені до джерела живлення, вони автоматично розпочинають процес виборів кореневого вузла. Цей вузол виконує роль основного координатора мережі, який забезпечує зв'язок з Інтернетом і керує маршрутизацією трафіку між вузлами.

Процес виборів кореневого вузла базується на алгоритмах, які враховують різні параметри, такі як потужність сигналу, рівень заряду батареї (якщо застосовується), і фізичне розташування вузлів. Це дозволяє забезпечити оптимальну продуктивність мережі. Після успішного підключення до мережі, кореневий вузол буде підключено до Інтернету, що дозволяє всім підключеним пристроям отримувати доступ до глобальної мережі.

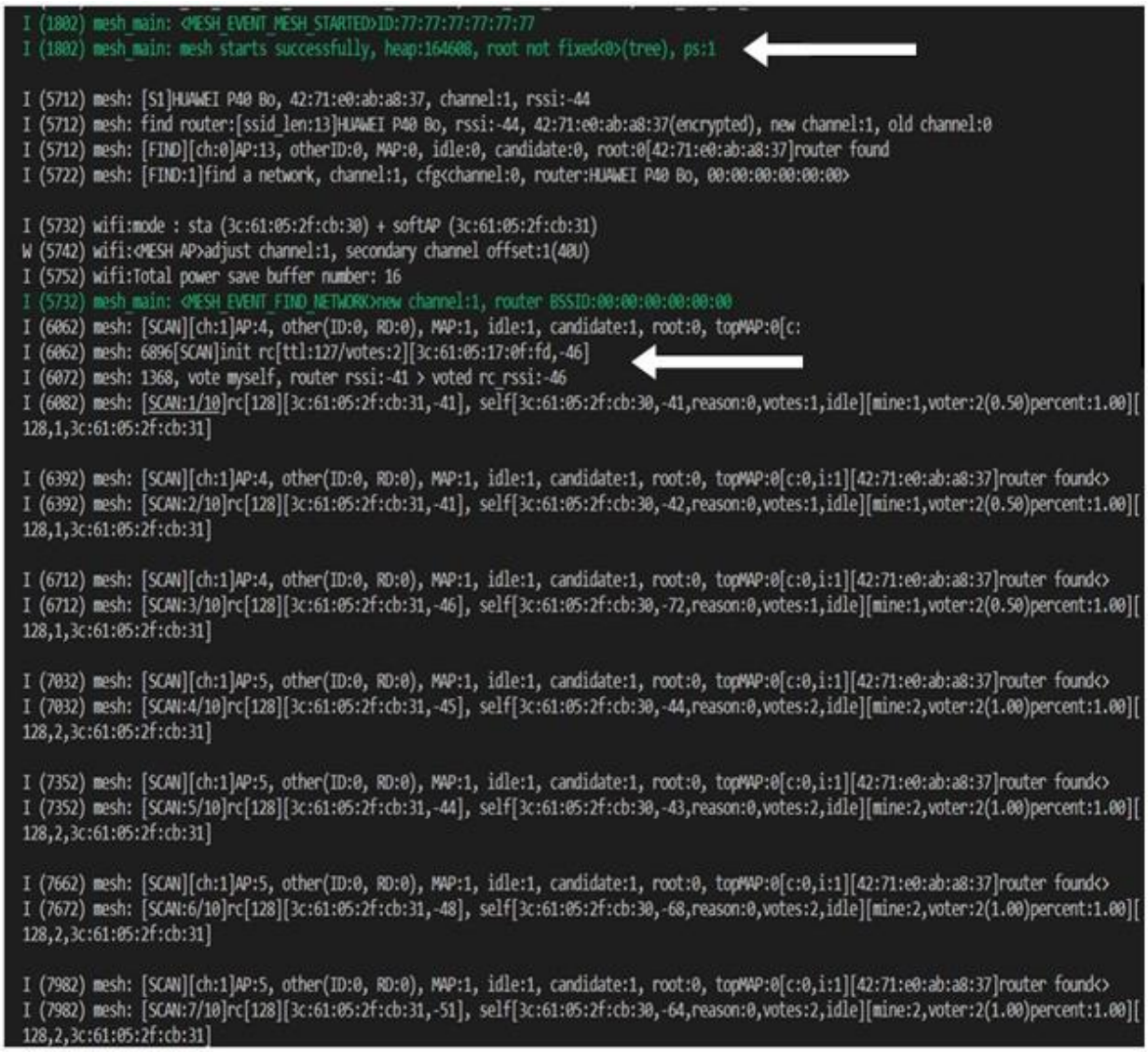

Рисунок 3.5 — Кореневий вузол виборів

На наступному етапі ми додали ще одну плату ESP32 до мережі. Ця нова плата автоматично підключилася до існуючої мережі, розширюючи її та покращуючи покриття. Кожен новий вузол, доданий до мережі, збільшує її загальну потужність і забезпечує більше маршрутів для передачі даних, що покращує надійність і швидкість мережі.

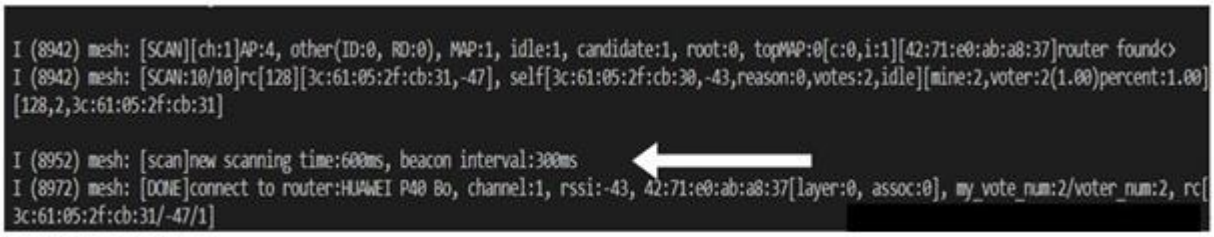

Рисунок 3.6 — Вибір кореневого вузла завершено

Мережа Wi-Fi Mesh автоматично адаптується до змін, забезпечуючи стабільне з'єднання для всіх підключених пристроїв. Це досягається за допомогою алгоритмів самовідновлення, які дозволяють мережі перебудовувати маршрути у випадку втрати зв'язку з одним з вузлів. Якщо один з вузлів виходить з ладу або втрачає з'єднання, інші вузли автоматично перебудують маршрути, щоб підтримати мережеве з'єднання.

Однією з переваг використання ESP32 у мережі Wi-Fi Mesh є можливість динамічного розподілу навантаження між вузлами. Це забезпечує високу ефективність і стабільність мережі навіть при збільшенні кількості підключених пристроїв. Кожен вузол може самостійно визначати оптимальні маршрути для передачі даних, враховуючи поточний стан мережі, навантаження на вузли та інші фактори.

Крім того, мережа Wi-Fi Mesh, побудована на базі ESP32, має високу ступінь безпеки. Використовуються сучасні протоколи шифрування даних, що забезпечує захист від несанкціонованого доступу і зломів. Це особливо важливо для використання в розумних будинках, промислових системах і інших середовищах, де безпека даних є критично важливою.

Ще однією важливою особливістю ESP32 є підтримка різноманітних інтерфейсів і протоколів зв'язку, таких як Bluetooth, SPI, I2C, UART, що робить ці плати ідеальними для інтеграції в різні проекти Інтернету речей (IoT). Це дозволяє створювати складні системи, де пристрої можуть взаємодіяти один з одним, обмінюватися даними і забезпечувати інтелектуальне управління різними аспектами середовища.

Використання ESP32 для побудови мережі Wi-Fi Mesh є ефективним рішенням для забезпечення стабільного і широкого покриття мережі в різних умовах. Ці плати забезпечують надійний і високопродуктивний зв'язок, що дозволяє реалізувати безліч інноваційних рішень у сфері домашньої автоматизації, промислових систем, розумного міста та інших областях.

3.3 Тестування функцій контролю доступу

Коли ми використовуємо карту RFID для доступу до пристрою, якщо пристрій є кореневим вузлом, результат відображається наступним чином. Коли ми отримуємо доступ до пристрою кореневого вузла, використовуючи RFID-карту, записану в системі, та незареєстровану RFID-картку, ми отримуємо результат:

```
write!
http://www.dongbtest.site/?tag=57:201:226:40&roomid=1
receive from 3c:61:05:2f:cb:30  request of http: http://www.dongbtest.site/?tag=57:201:226:40&roomid=1
read!
I (1955252) mesh_main: HTTP_EVENT_DISCONNECTED
I (1955262) mesh_main: HTTP GET Status = 200, content_length = 1
I (1955262) mesh_main: 31
I (1955262) mesh_main: we get the request is:1

I (1955262) mesh_main: HTTP_EVENT_DISCONNECTED
I (1959192) mesh_main: http://www.dongbtest.site/?tag=218:224:237:117&roomid=1
write!
http://www.dongbtest.site/?tag=218:224:237:117&roomid=1
receive from 3c:61:05:2f:cb:30  request of http: http://www.dongbtest.site/?tag=218:224:237:117&roomid=1
read!
W (1959642) wifi:<ba-add>idx:1 (ifx:0, 42:71:e0:ab:a8:37), tid:1, ssn:10, winSize:64
I (1960092) mesh_main: HTTP_EVENT_DISCONNECTED
I (1960092) mesh_main: HTTP GET Status = 200, content_length = 1
I (1960102) mesh_main: 30
I (1960102) mesh_main: we get the request is:0

I (1960102) mesh_main: HTTP_EVENT_DISCONNECTED
```

Рисунок 3.7 — Перевірка результатів влаштування кореневого вузла

Коли ми використовуємо RFID-карту, записану в системі, та незареєстровану RFID-карту для доступу до пристрою дочірнього вузла ми отримуємо результат.

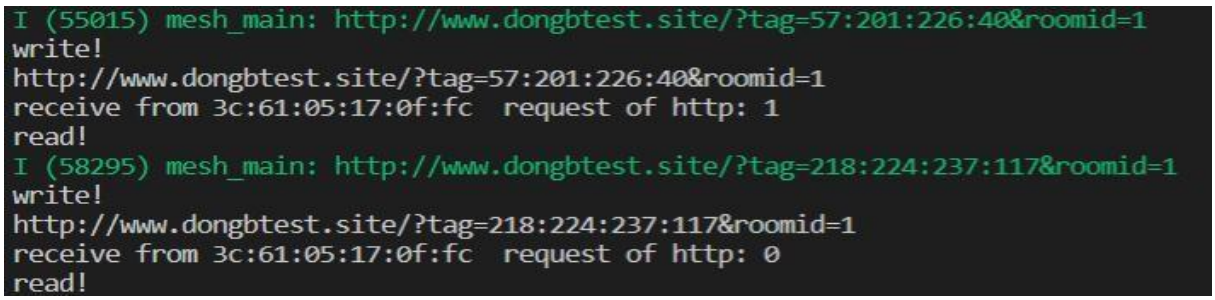

Рисунок 3.8 — Перевірка результатів влаштування дочірнього вузла

3.4 Самовідновлення пористих мереж

Після того, як ми відключимо живлення кореневого вузла, мережа повторно вибере кореневий вузол. У тесті процес повторного вибору кореневого вузла триватиме близько хвилини, і результат показаний рисунку 3.9.

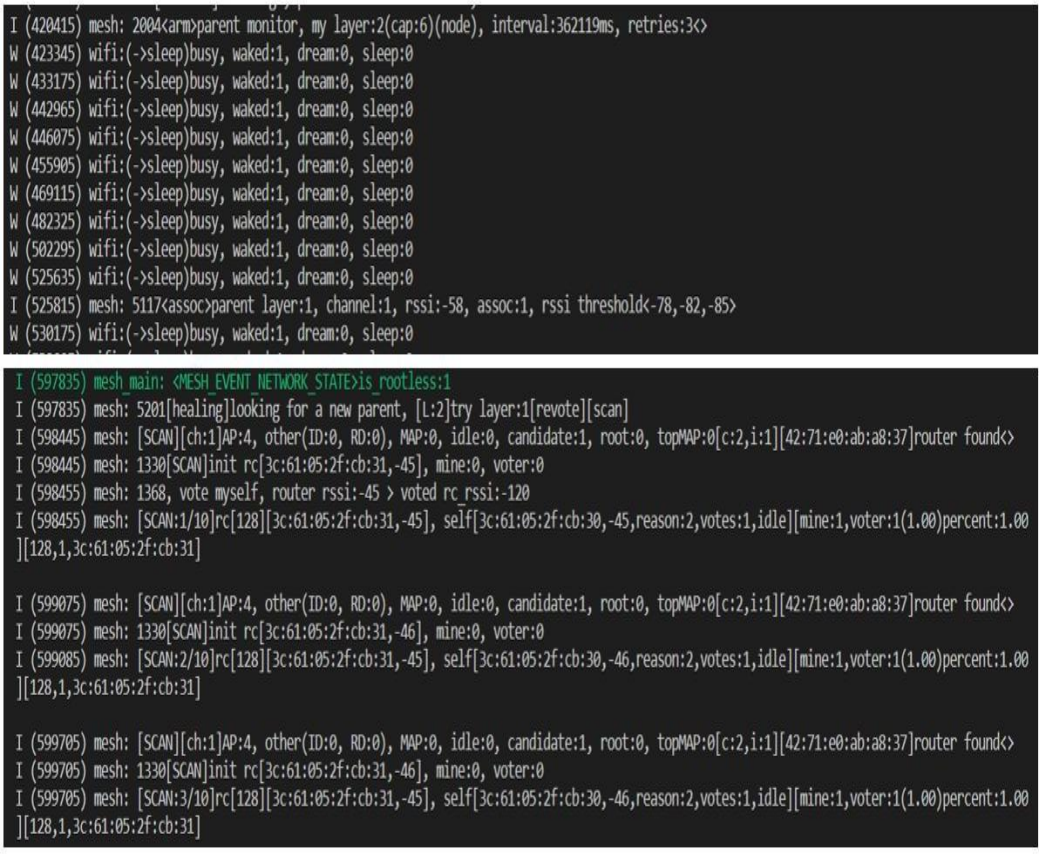

Рисунок 3.9 — Самовідновлення пористих мереж

Після самовідновлення пористої мережі дочірній вузол стає кореневим вузлом.

```
I (604075) mesh: [SCAN][ch:1]AP:4, other(ID:0, RD:0), MAP:0, idle:0, candidate:1, root:0, topMAP:0[c:2,i:1][42:71:e0:ab:a8:37]router found<>
I (604075) mesh: 1330[SCAN]init rc[3c:61:05:2f:cb:31,-45], mine:0, voter:0
I (604085) mesh: [SCAN:10/10]rc[128][3c:61:05:2f:cb:31,-44], self[3c:61:05:2f:cb:30,-45,reason:2,votes:1,idle][mine:1,voter:1(1.00)percent:1.0
0][128,1,3c:61:05:2f:cb:31]

I (604095) mesh: [DONE]connect to router:HUAWEI P40 Bo, channel:1, rssi:-62, 42:71:e0:ab:a8:37[layer:0, assoc:0], my_vote_num:1/voter_num:1, r
c[3c:61:05:2f:cb:31/-44/2]
I (604165) wifi:new:<1,1>, old:<1,0>, ap:<1,1>, sta:<1,0>, prof:1
I (605035) wifi:state: init -> auth (b0)
I (605055) wifi:state: auth -> assoc (0)
I (605065) wifi:state: assoc -> run (10)
I (605085) wifi:connected with HUAWEI P40 Bo, aid = 11, channel 1, BW20, bssid = 42:71:e0:ab:a8:37
I (605085) wifi:security: WPA2-PSK, phy: bgn, rssi: -45
I (605095) wifi:Set ps type: 0

I (605105) mesh: <nvs>write layer:1
I (605105) mesh: <flush>upstream packets, connections(max):6, waiting:0, upQ:0
I (605115) mesh_main: <MESH_EVENT_TODS_REACHABLE>state:0
I (605115) mesh: <flush>root
I (605115) mesh: [TXQ]<max:128>up(0, be:0), down(0, be:0), mgmt:0, xon(req:0, rsp:0), bcast:0, wnd(2, parent:3c:61:05:17:0f:fd)
I (605125) mesh: [RXQ]<max:128 = cfg:128 + extra:0>self:0, <max:128 = cfg:128 + extra:0>tods:0
I (605175) wifi:pm start, type: 0

I (605275) wifi:AP's beacon interval = 102400 us, DTIM period = 2
I (605705) esp_netif_handlers: sta ip: 192.168.43.163, mask: 255.255.255.0, gw: 192.168.43.1
I (605705) mesh_main: <IP_EVENT_STA_GOT_IP>IP:192.168.43.163
W (653175) wifi:<ba-add>idx:0 (ifx:0, 42:71:e0:ab:a8:37), tid:0, ssn:13, winSize:64
I (895605) mesh_main: http://www.dongbtest.site/?tag=57:201:226:40&roomid=1
write!
http://www.dongbtest.site/?tag=57:201:226:40&roomid=1
receive from 3c:61:05:2f:cb:30  request of http: http://www.dongbtest.site/?tag=57:201:226:40&roomid=1
read!
I (896645) mesh_main: HTTP_EVENT_DISCONNECTED
I (896645) mesh_main: HTTP GET Status = 200, content_length = 1
I (896655) mesh_main: 31
I (896655) mesh_main: we get the request is:1

I (896655) mesh_main: HTTP_EVENT_DISCONNECTED
```

Рисунок 3.10 — Самовідновлення пористих мереж

ВИСНОВКИ

У роботі розроблено розподілену мережеву для автоматичного управління доступом. Цю систему розділено на три частини для розробки відповідно. Система може добре працювати і може відповідати робочим вимогам контролю доступу. У процесі розробки глибоко вивчено використання мікроконтролера ESP32 та характеристики мережі ESP-WIFI-MESH на його основі, вивчено мережевий протокол HTTP.

Крім того, в процесі розробки управління користувачами системи контролю доступу:

— отримано можливості розробки веб-сторінок;

— освоєно можливість розробки веб-сторінок з використанням мов HTML+CSS та JavaScript.

Під час розгортання та використання сервера:

— вивчено функції та використання хмарного сервера;

— освоєно метод роботи та використання бази даних на основі MySQL;

— оглянуто та вивчено можливості мови PHP для обробки запитів доступу до веб-сторінки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кулаков Ю.А., Луцький Г.М. Локальні мережі. – Київ: Юніор, 1998. – 45 с.
2. Гамаюн І. П., Безменова О.П. Оцінювання міри схожості між об’єктам, що характеризуються кількісними і номінальними ознаками. – Харків : НТУ «ХПІ», 2013. – 19 с.
3. Тарнавський Ю.А., Кузьменко І.М. Організація комп’ютерних мереж: підручник. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 259 с.
4. Городецька О.С., Гикавий В.А., Онищук О.В. Комп’ютерні мережі: навчальний посібник. – Вінниця: ВНТУ, 2017. – 129 с.
5. Буров Є. В. Комп’ютерні мережі. – Львів: Бак, 2003. – 468 с.
6. Бірюков М.Л., Стеклов В.К., Костік Б.Я. Транспортні мережі телекомунікацій: системи мультиплексування. – Київ: Техніка, 2005. – 312 с.
7. Dong M., Ota K., Liu A., Guo M. Joint optimization of lifetime and transport delay under reliability constraint wireless sensor network. IEEE Trans Wirel Commun. – 2016. – V.27(1). – P.225–236.
8. Ren J., Zhang Y., Liu A., Zhang N. (2016) Dynamic channel access to improve energy efficiency in cognitive radio sensor networks. IEEE Trans Wirel Commun. – 2016. – V.15(5). – P.3143–3156.
9. Alicherry M., Bhatia R., Li L.E. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In: Proceedings of the 11th annual international conference on Mobile computing and networking. – 2005. – P.58–72.
10. Si W., Selvakennedy S., Zomaya A.Y. (2010) An overview of channel assignment methods for multi-radio multi-channel wireless mesh networks. J. Parallel. Distrib. Comput. – 2010. – V.70(5). – P. 505–524.

11. Pirzada A.A., Portmann M., Indulska J. Performance analysis of multi-radio aodv in hybrid wireless mesh networks. Comput. Commun.–2008. – V.31(5). – P. 885–895.

12. Mogaibel H.A., Othman M., Subramaniam S., Asilah Wati Abdul Hamid N. Ondemand channel reservation scheme for common traffic in wireless mesh networks. J. Net. Comput Appl. – 2012. – V.35(4). – P. 1329–1351. 34

13. Draves R., Padhye J., Zill B. Routing in multi-radio, multi-hop wireless mesh networks. In: Proceedings of the 10th annual international conference on Mobile computing and networking. ACM. – 2004. – P.114–128.

14. Chandra R., Bahl P. Multinet: Connecting to multiple ieee 802.11 networks using a single wireless card. In: INFO-COM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies . – 2004. – V.2. – P. 882–893.

15. Xie K., Wang X., Liu X. (2016) Cooperative routing with relay assignment in multiradio multi-hop wireless networks. IEEE Trans. Comput. – 2016. – V.24(2). –P.859– 872.

16. Xie K., Wang X., Liu X. Interference-aware cooperative communication in multiradio multi-channel wireless networks. IEEE Trans. Comput.– 2016. – V.65(5). – P. 1528–1542.

17. Gui J, Zhou K, Xiong N (2016) A cluster-based dual-adaptive topology control approach in wireless sensor networks. Sensors. – 2016. – V.16(10). – P. 1576.

18. Wang K., Yang K., Chen H. Computation diversity in emerging networking paradigms. IEEE Wirel Commun. – 2017. – V.24(1) . – P.88–94.

19. Zhang D., Chen Z., Awad M.K., Zhou H., Zhang N., Shen X. Utility-optimal resource management and allocation algotithm for energy harvesting congtive radio sensor networks. In: IEEE Journal on Selected Areas in Communications. – 2017. – V.34. – P.3552–3565.

ДОДАТОК А

Міністерство освіти та науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ ВНТУ
д.т.н., проф.
_____О. Д. Азаров
"23" квітня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської дипломної роботи

« Розподілена мережа для автоматичного управління доступом »

08-54.БДР.006.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Колесник І.С.

Студента групи КІ-22мс

_____ Ляховецький Д.С.

Вінниця 2024

1. Найменування та область застосування

Розподілена мережа для автоматичного управління доступом.

2. Основи для розробки

Необхідність реалізації розподіленої мережі для автоматичного управління доступом.

3. Мета та призначення розробки

Метою дипломної роботи є розробка розподіленої мережі для автоматичного управління доступом.

4. Етапи БДР та очікувані результати

Робота виконується у вісім етапів, що наведені в таблиці 4.1.

Таблиця 4.1 – Етапи виконання роботи

№ з/п	Назва етапів виконання бакалаврського проекту	Строк виконання етапів роботи	Примітка
1	Постановка задач проекту	27.02.24	
2	Огляд інформаційних джерел	06.03-20.03.24	
3	Огляд та аналіз розподілених мереж ESP-MESH	27.03-7.04.24	
4	Проектування автоматизованої мережі для автоматичного управління доступом	10.04-21.04.24	
5	Результати проектування та їх тестування	24.04-12.05.24	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	15.05-26.05.24	
7	Аналіз виконання проекту. Висновки. Додатки	29.05-09.06.24	
8	Перевірка якості виконання бакалаврського проекту та усунення недоліків	07.06.24	

5. Матеріали, що подаються до захисту БДР

Пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відзив наукового керівника, рецензія опонента, протоколи складання державних екзаменів, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДП діючим вимогам.

6. Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДП відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

7. Вимоги до оформлення БДР

При оформлюванні БДР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— документами на які посилаються у вище вказаних.

ДОДАТОК Б

Структура системи контролю доступу

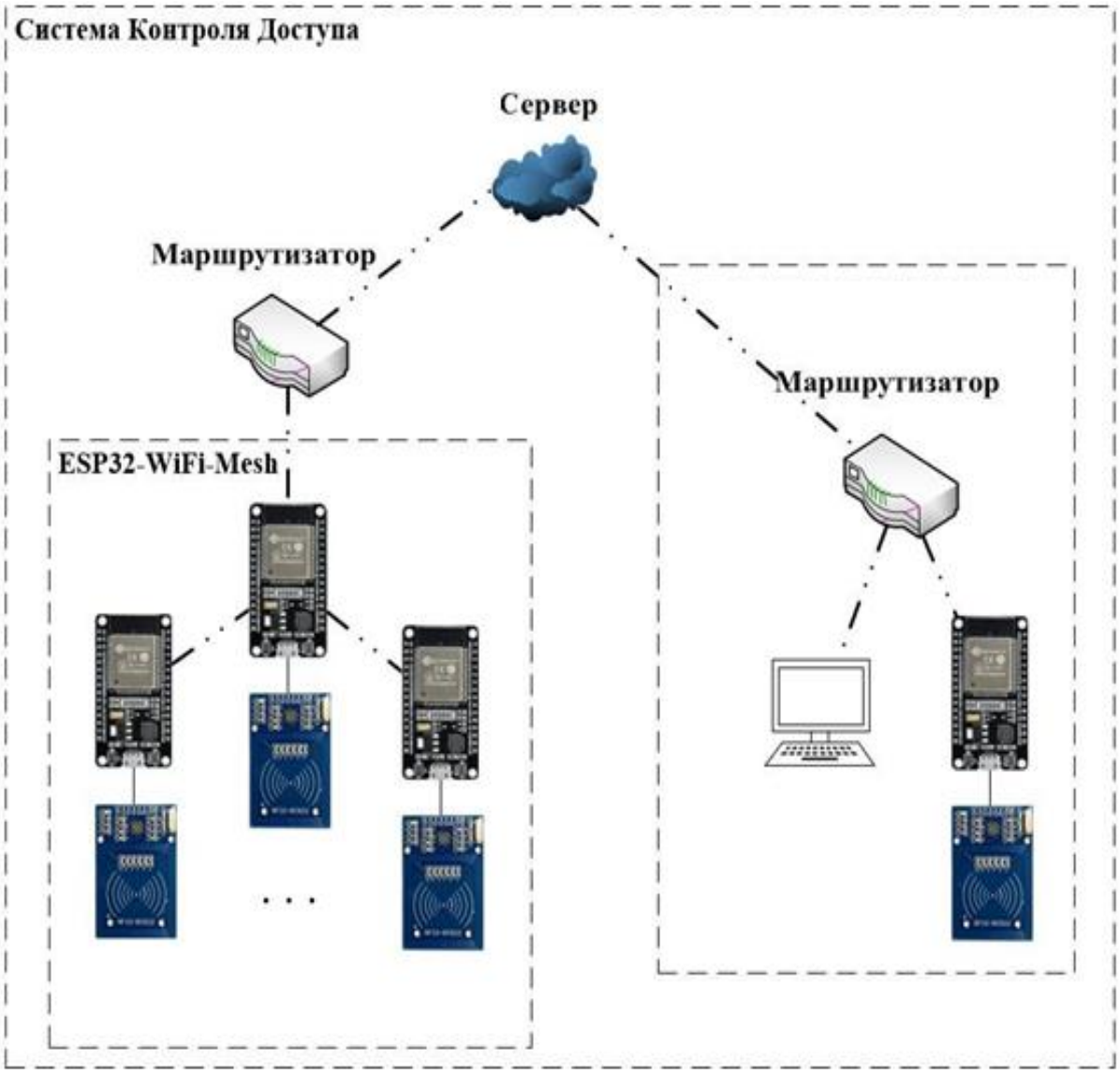

Рисунок Б1 — Структура системи контролю доступу

ДОДАТОК В

Структура бази даних

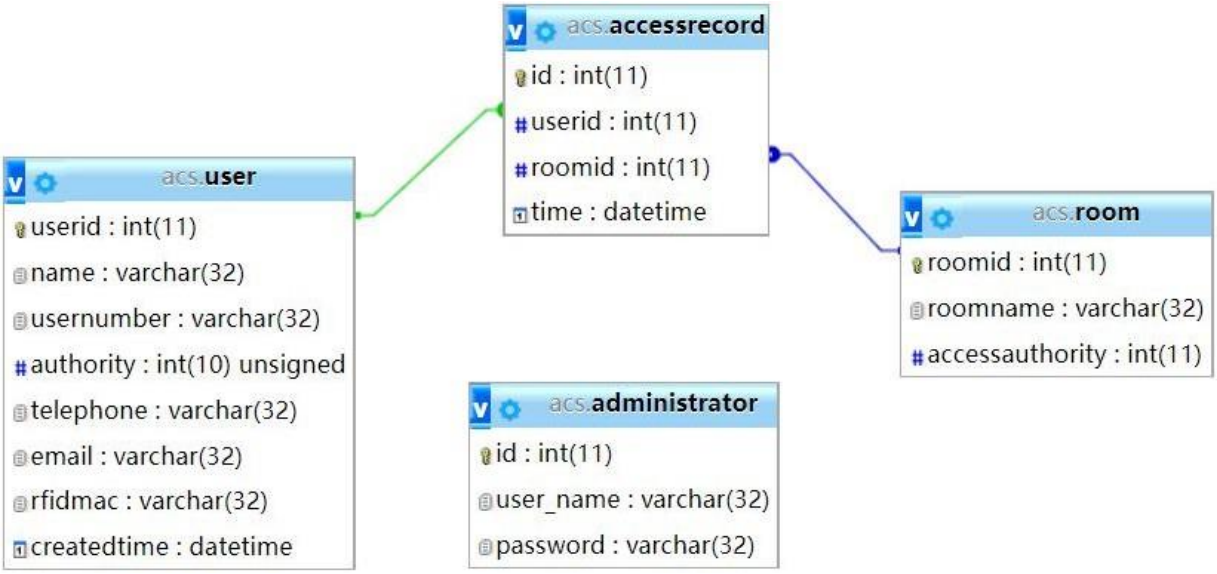

Рисунок В1 — Структура бази даних

ДОДАТОК Г

Мережа ESP-MESH, реалізована за допомогою інструмента ESP-IDF

```
#include"rc522.h"...
#define MAX_HTTP_RECV_BUFFER512
#define MAX_HTTP_OUTPUT_BUFFER2048
#define BOOL int
#define TRUE1
#define FALSE0
#define RX_SIZE (1500)
#define TX_SIZE (1460)
#define CONFIG_MESH_ROUTER_SSID"HUAWEI P40 Bo"#define CONFIG_MESH_ROUTER_PASSWD"12345678"
static const char*MESH_TAG ="mesh_main";
static const uint8_tMESH_ID[6] = {0x77,0x77,0x77,0x77,0x77,0x77};static uint8_ttx_buf[TX_SIZE] = {0, };
static uint8_trx_buf[RX_SIZE] = {0, };static boolis_running =true;
static boolis_mesh_connected =false;staticmesh_addr_t mesh_parent_addr;static intmesh_layer = -1;
staticesp_netif_t *netif_sta =NULL;const char* roomid ="1";
BOOL meshWork=FALSE; BOOL toSlef=FALSE;
structtagRFID
{
inttag[5];char*url;
};
structwriteData
{
uint8_t*addr;char* url;
};
```

```
esp_err_t _http_event_handler(esp_http_client_event_t *evt);static voidtask1(void* par);
static char* http_rest_with_url(char* toUrl);voidespwrite(char*mess,uint8_ttoAddr[6]);voidtag_handler(uint8_t* sn) {
if(meshWork) {
charurl[200] ="";
sprintf(url,"http://www.dongbtest.site/?tag=%d:%d:%d:%d&roomid=%s", sn [0], sn [1], sn [2], sn [3], roomid);if(esp_mesh_is_root()){
toSlef=TRUE;
}
espwrite(url,NULL);
}
}
voidespwrite(char*mess,uint8_ttoAddr[6])
{
esp_err_t err;
mesh_addr_t route_table[50];introute_table_size =0; mesh_data_t data;
data.data = tx_buf; data.size =sizeof(tx_buf); data.proto = MESH_PROTO_BIN; data.tos = MESH_TOS_P2P;
esp_mesh_get_routing_table((mesh_addr_t *) &route_table,
50*6, &route_table_size); memcpy(tx_buf, (uint8_t*)mess, strlen(mess)+1);if(!esp_mesh_is_root() || toAddr==NULL) {
err = esp_mesh_send(NULL, &data,0,NULL,0);
}
else if(toAddr!=NULL) {
err = esp_mesh_send(toAddr, &data, MESH_DATA_P2P,NULL,0);
}
}
voidespread(void* arg)
```

```
{
intrecv_count =0; esp_err_t err; mesh_addr_t from;intsend_count =0; mesh_data_t data;intflag =0; data.data = rx_buf; data.size = RX_SIZE; is_running =true;
while(is_running) {data.size = RX_SIZE;
err = esp_mesh_recv(&from, &data, portMAX_DELAY, &flag,NULL,0);if(esp_mesh_is_root() && err==ESP_OK){
structwriteData d1={&from.addr,(char*)data.data}; xTaskCreate(task1,"task",10*1024, &d1,5,NULL);
}
}
vTaskDelete(NULL);
}
esp_err_t esp_mesh_comm_p2p_start(void)...
//Mesh event handler
voidmesh_event_handler(void*arg, esp_event_base_t event_base,
int32_tevent_id,void*event_data)...
//ip event handler
voidip_event_handler(void*arg, esp_event_base_t event_base,
int32_tevent_id,void*event_data)...
voidapp_main(void)
{
ESP_ERROR_CHECK(nvs_flash_init());
/* tcpip initialization */ESP_ERROR_CHECK(esp_netif_init());
/* event initialization */ESP_ERROR_CHECK(esp_event_loop_create_default());
/* create network interfaces for Mesh
ignored */
```

```
ESP_ERROR_CHECK(esp_netif_create_default_wifi_mesh_netifs(&netif_
sta, NULL));
/* wifi initialization */
wifi_init_config_t config = WIFI_INIT_CONFIG_DEFAULT();
ESP_ERROR_CHECK(esp_wifi_init(&config));
ESP_ERROR_CHECK(esp_event_handler_register(IP_EVENT,
IP_EVENT_STA_GOT_IP, &ip_event_handler,NULL));
ESP_ERROR_CHECK(esp_wifi_set_storage(WIFI_STORAGE_FLASH));
ESP_ERROR_CHECK(esp_wifi_start());
/* mesh initialization */ESP_ERROR_CHECK(esp_mesh_init());
ESP_ERROR_CHECK(esp_event_handler_register(MESH_EVENT,
ESP_EVENT_ANY_ID, &mesh_event_handler,NULL));
/* set mesh topology
*/ESP_ERROR_CHECK(esp_mesh_set_topology(CONFIG_MESH_TOPOLOGY
));
/* set mesh max layer підтверджується топологією
*/ESP_ERROR_CHECK(esp_mesh_set_max_layer(CONFIG_MESH_MAX_LA
YER));ESP_ERROR_CHECK(esp_mesh_set_vote_percentage(1));
ESP_ERROR_CHECK(esp_mesh_set_xon_qsize(128));
constrc522_start_args_t start_args = {
.miso_io =19,
.mosi_io =23,
 .sck_io=  18,
.sda_io=  21,
.callback = &tag_handler,
};
rc522_start(start_args);#ifdef CONFIG_MESH_ENABLE_PS
/* Enable mesh PS функція
*/ESP_ERROR_CHECK(esp_mesh_enable_ps());
```

```
/* better to increase the associate expired time, if a small duty cycle is set.
*/ESP_ERROR_CHECK(esp_mesh_set_ap_assoc_expire(60));
/* better to increase the announce interval to avoid too much management
traffic, if a small duty cycle is
set. */

ESP_ERROR_CHECK(esp_mesh_set_announce_interval(600,3300));#else
/* Disable mesh PS функція
*/ESP_ERROR_CHECK(esp_mesh_disable_ps());
ESP_ERROR_CHECK(esp_mesh_set_ap_assoc_expire(10));
#endif
mesh_cfg_t cfg = MESH_INIT_CONFIG_DEFAULT();
/* mesh ID */
memcpy((uint8_t*) &cfg.mesh_id, MESH_ID,6);
/* router */
cfg.channel = CONFIG_MESH_CHANNEL;
cfg.router.ssid_len = strlen (CONFIG_MESH_ROUTER_SSID);
memcpy((uint8_t*) &cfg.router.ssid, CONFIG_MESH_ROUTER_SSID,
cfg.router.ssid_len); memcpy((uint8_t*) &cfg.router.password,
CONFIG_MESH_ROUTER_PASSWD,
strlen(CONFIG_MESH_ROUTER_PASSWD));
/* mesh softAP
*/ESP_ERROR_CHECK(esp_mesh_set_ap_authmode(CONFIG_MESH_AP_AU
THMODE)); cfg.mesh_ap.max_connection =
CONFIG_MESH_AP_CONNECTIONS; cfg.mesh_ap.nonmesh_max_connection
= CONFIG_MESH_NON_MESH_AP_CONNECTIONS; memcpy((uint8_t*)
&cfg.mesh_ap.password, CONFIG_MESH_AP_PASSWD,
strlen(CONFIG_MESH_AP_PASSWD));
ESP_ERROR_CHECK(esp_mesh_set_config(&cfg));
```

/* mesh start */ESP_ERROR_CHECK(esp_mesh_start());

#ifdef CONFIG_MESH_ENABLE_PS

/* Set device active duty cycle. (default:10, MESH_PS_DEVICE_DUTY_REQUEST) */ESP_ERROR_CHECK(esp_mesh_set_active_duty_cycle(CONFIG_MESH_PS_DEV_DUTY, CONFIG_MESH_PS_DEV_DUTY_TYPE));

/* Set network active duty cycle. (default:10, -1, MESH_PS_NETWORK_DUTY_APPLIED_ENTIRE) */

ESP_ERROR_CHECK(esp_mesh_set_network_duty_cycle(CONFIG_MESH_PS_NWK_DUTY,CONFIG_MESH_PS_NWK_DUTY_DURATION, CONFIG_MESH_PS_NWK_DUTY_RULE));

#endif

ESP_LOGI(MESH_TAG,

"meshstartssuccessfully,heap:%d,%s<%d>%s,

ps:%d\n",esp_get_minimum_free_heap_size(),

esp_mesh_is_root_fixed() ?"root fixed":"root not fixed",

esp_mesh_get_topology(),esp_mesh_get_topology() ?"(chain)":"(tree)", esp_mesh_is_ps_enabled());

}

esp_err_t _http_event_handler(esp_http_client_event_t *evt)...static char* http_rest_with_url(char* toUrl)

{

charlocal_response_buffer[MAX_HTTP_OUTPUT_BUFFER] = {0};

/**

* NOTE:Всі параметри параметрів для http_client повинні бути specied either in URL or as host and path

параметри.

* Якщо host and path parameters not set, query parameter will be ignored. In such cases,

```
*       query parameter повинен бути specified в URL.
*       Якщо URL-адреси є добре як host і path parametrs є визначеними, значення host і path буде вважатися.
*/
esp_http_client_config_t config = {
.method=HTTP_METHOD_GET,
.url = toUrl,
.event_handler = _http_event_handler,
.user_data =local_response_buffer,  // Pass address of local buffer to get response
};
esp_http_client_handle_t client = esp_http_client_init(&config); esp_err_t err = esp_http_client_perform(client);
if(err == ESP_OK) {
ESP_LOGI(MESH_TAG,"HTTP GET Status = %d, content_length = %d", esp_http_client_get_status_code(client), esp_http_client_get_content_length(client));
}else{
ESP_LOGE(MESH_TAG,"HTTP GET request failed: %s", esp_err_to_name(err));
}
char* result = (char*)local_response_buffer; esp_http_client_cleanup(client);
returnresult;
}
static voidtask1(void* par)
{
structwriteData *d; d=(structwriteData*)par;char* result;
result=http_rest_with_url((*d).url);if(toSlef) {
```

```
toSlef=FALSE;
}
else
{
espwrite(result,(*d).addr);
}
vTaskDelete(NULL);
}
```

ДОДАТОК В

Обробник доступу PHP

```
<?php
//Встановіть відповідні заголовки HTTP header('Access-Control-Allow-Origin:*'); header("Cache-Control: no-cache, private"); header("Connection: Keep-Alive"); header("Content-length: 1");
// Дозвіл на доступ до бази даних
$ dbhost = "localhost";
$dbuser = "***";
$dbpass = "***";
$dbname = 'acs';
$conn = mysqli_connect($dbhost, $dbuser, $dbpass);
//Помилка підключення до бази даних if(! $ Conn) {
echo 2; die();
}
mysqli_query($conn, "set names utf8"); mysqli_select_db($conn, $dbname);
/*Дані, отримані методом POST, отримані із системи управління користувачами, а дані, отримані методом GET, отримані від HTTP-клієнта мережі WiFi-Mesh ESP32.
*/
if( $_SERVER['REQUEST_METHOD'] == 'POST'){
if($_POST["mode"]=="login"){
$admin=$_POST["adm"];
$password=$_POST["pwd"];
$sql="SELECT user_name,password FROM administrator WHERE user_name='{$admin}' and password='{$password}'";
```

```
$result = mysqli_query($conn, $sql); if (mysqli_num_rows($result) > 0){
echo 1;
}else{
echo 0;
}
mysqli_close($conn);
}else if($_POST["mode"]=="create"){
 /*
create це режим реєстрації
create Інформація про нову карту може бути записана
Дані, додані до бази даних:name id telephone email position rfidmac
*/
$sql="SELECT userid FROM user WHERE rfidmac='{$_POST["rfidmac"]}'";
$result = mysqli_query($conn, $sql); if (mysqli_num_rows($result) > 0){
echo 0;
}else{
$ authority; if($_POST["position"]=="teacher"){
$ authority = 9;
}else{
$ authority = 5;
}
$sql="INSERT INTO user (name,usernumber,authority,telephone,email,rfidmac) VALUES
('{$_POST["name"]}','{$_POST["usernumber"]}','{$authority}','{$_POST["telephone"]}','{$_POST[" email"]}','{$_POST["rfidmac"]}')";
```

```
if(mysqli_query($conn, $sql)){
//Вставити дані успішно echo 1;
}else{
//Не вдалося вставити дані echo 0;
}
}
mysqli_close($conn);
}else if($_POST["mode"]=="delete"){
//режим видалення
$sql="DELETE FROM user WHERE rfidmac='{$_POST["rfidmac"]}"';
if(mysqli_query($conn, $sql)){
//Дані успішно видалені echo 1;
}else{
//Не вдалося видалити дані echo 0;
}
}
}else if( $_SERVER['REQUEST_METHOD'] == 'GET'){
$tag = $_GET['tag'];

$roomid = $_GET['roomid'];
$sql = "SELECT userid ,authority FROM user WHERE rfidmac='{$tag}"';
$ retval = mysqli_query ($ conn, $ sql); if(! $retval ){
echo 2; mysqli_close($conn); die();
}
if(mysqli_num_rows($retval)<=0){echo 0;
mysqli_close($conn); die();
}
```

```
//Визначити, чи може дозвіл відвідувача отримати доступ до кімнати
$row = mysqli_fetch_array($retval, MYSQLI_ASSOC);
$authorityUser = $row['authority'];
$userid = $row['userid'];
$sql = "SELECT accessauthority FROM room WHERE roomid='{$roomid}'";
$ retval = mysqli_query ($ conn, $ sql); if(! $retval ){
echo 2; mysqli_close($conn); die();
}
if(mysqli_num_rows($retval)<=0){echo 0;
mysqli_close($conn); die();
}
$row = mysqli_fetch_array($retval, MYSQLI_ASSOC);
$authorityRoom = $row['accessauthority'];

if($authorityRoom<=$authorityUser){
//Додати запис доступу echo 1;
$sql = "INSERT INTO accessrecord (userid,roomid) VALUES ('{$userid}','{$roomid}')";
$ retval = mysqli_query ($ conn, $ sql); if(! $retval ){
//не вдалося додати echo 2;
}
```

ДОДАТОК Д

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розподілена мережа для автоматичного управління доступом

Тип роботи: Бакалаврська дипломна робота

(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck
Оригінальність 91,5% Схожість 8,5%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Ляховецький Д.С.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Колесник І.С. (підпис)