МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Безпека комп'ютерних систем і мереж»

Група: 4КБ-02

Дипломний проект

здобувача освіти денної форми навчання
КБ.02.17.000.ДП

Рибакова
В'ячеслава Олеговича

м. Одеса
2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Безпека комп'ютерних систем і мереж»**

Група: **4КБ-02**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту на тему:

Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю

Проєктний матеріал складається з пояснювальної записки на 78 сторінках та графічного (презентаційного) матеріалу на 11 аркушах (слайдах)

Дипломник _____ (Рибаков В.О.)

Керівник _____ (Залапін О.І.)

Консультанти:

з економічного розділу_____ (Канський М.Ю.)

з розділу охорони праці та техніки безпеки_____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Краснокутська К.Г.)

Захист «20» червня 2025 р. Протокол ЕК № 1

Оцінка ЕК 5 (відмінно) / 95 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Безпека комп'ютерних систем і мереж»*

ЗАТВЕРДЖУЮ:

Заст. дир. НВР *Беркань І.В.*

"19" 05 2025 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Рибакову В'ячеславу Олеговичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю*

затверджена наказом по коледжу від "14" листопада 2024 р. № 246

2. Термін здачі закінченого проекту

3. Вихідні данні до проекту *1. Розробити електричну-принципову схему для плати-приймача та передавача системи авторизації користувача з використанням модулів ESP8266 та* NRF24L01 *для обміну даними; 2. Розробити апаратну частину із двох плат – приймач та передавач; 3. Розробити друковану плату пристрою, виконати травлення, пайку; 4. Розробити програмний код для приймача та передавача на мові Arduino C. 5. Реалізувати програмне забезпечення web-сервера на Python (у середовищі VSCode), та web-інтрефейсу з кикористанням HTML/CSS/JavaScript, що забезпечує збереження UID, перевірку наявності UID на сервері, та керування UID. 7. Забезпечити відправлення даних між приймачем і web-сервером.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Проектування електрично-принципових схеми; Збірка та монтаж апаратної частини; Розробка програмного забезпечення для плати-приймача; Розробка програмного забезпечення для плати-передавача; Реалізація серверної частини (зберегання, керування UID). Розробка web-інтерфейсу для керування; Проведення тестування всіх модулів системи; Аналіз отриманих результатів; Економічний розділ; Розділ охорони праці і техніки безпеки

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Модель загальної архітектури системи авторизації; Модулі системи передавача; Принципово-електрична схема передавача; Модулі системи приймача; Принципова схема приймача; Демонстрація виготовлених пристроїв; Алгоритм реєстрації UID; Алгоритм авторизації UID; Алгоритм видалення UID; Розробка web-інтерфейсу.

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Залапін О.І.*		
Економічний розділ	*Канський М.Ю.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання _____

Керівник *Залапін О.І.* _____
(підпис)

Завдання прийняв до виконання *Рибаков В.О.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Огляд технологій авторизації користувачів*	*20.10.2024*	виконав
2	*Аналіз вивченого матеріалу та опис вимог до системи*	*12.01.2025*	виконав
3	*Розробка електричної принципової схеми пристрою*	*10.03.2025*	виконав
4	*Виготовлення друкованої плати*	*19.03.2025*	виконав
5	*Виготовлення апаратної частини*	*23.03.2025*	виконав
6	*Розробка програмного коду для приймача та передавача*	*30.03.2025*	виконав
7	*Розробка програмного забезпечення web-сервера*	*09.04.2025*	виконав
8	*Розробка web-інтерфейсу*	*14.04.2025*	виконав
9	*Тестування роботи системи*	*17.04.2025*	виконав
10	*Перевірка функцій приймача та передавача*	*20.04.2025*	виконав
11	*Аналіз результатів тестування*	*02.05.2025*	виконав
12	*Оформлення пояснювальної записки*	*12.05.2025*	виконав
13	*Виконання економічних розрахунків*	*22.05.2025*	виконав
14	*Розробка заходів з охорони праці*	*30.05.2025*	виконав

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КБ 01.17 000.00 ДП	Дипломний проект		
А4			КБ 01.17 000.00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КБ 02. 17 000. 00 ДП		
Розробив		Рибаков В.О.		16.06.	Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю	Літ.	Аркуш 4 / Аркушів 78
Перевірив		Залапін О.І.		16.06.		Н Д П	
Н. контр.		Петрашова В.І.		16.06.25		ВСП ОТФК ОНТУ гр. 4КБ-02	
Затверд.		Кривченко Ю.В.		16.06			

ЗМІСТ

ВСТУП

Сьогодні все більше аспектів нашого життя залежать від цифрових технологій. Те, що раніше здавалося неможливим, зараз є звичайною справою: електронні замки, інтелектуальні домофони та системи контролю доступу, що не потребують ключів – достатньо сучасного гаджета у кишені. Це стимулює пошук нових, зручніших та водночас безпечних методів організації ідентифікації та контролю доступу до різних об'єктів.

Одним із рішень для побудови таких систем є платформа Arduino. Вона дозволяє швидко створювати прототипи пристроїв. Простота підключення зовнішніх модулів робить Arduino ідеальною основою для реалізації інтелектуальних систем авторизації.

Система авторизації - це комплекс технічних і програмних рішень, метою яких є забезпечення доступу користувача до певного об'єкта чи ресурсу лише у тому випадку, якщо його особа підтверджена і він має відповідні права. Така система дозволяє захистити фізичний або цифровий простір від несанкціонованого доступу, забезпечуючи контроль і порядок.

Один з найефективніших варіантів – модуль ESP8266. Це невеликий мікроконтроллер з Wi-Fi, який дозволяє легко інтегрувати приймач у мережу. Проте іноді лише Wi-Fi недостатньо: потрібна енергоефективність, гнучкість та можливість обміну даними на коротких відстанях між пристроями. У таких ситуаціях доцільним є використання радіомодулів NRF, які забезпечують надійний зв’язок з мінімальним споживанням енергії.

Важливим елементом сучасних систем авторизації є наявність зручного та функціонального web-інтерфейсу. Саме завдяки веб-доступу адміністратор має можливість взаємодіяти з системою: переглядати журнал подій, відслідковувати спроби входу, додавати або видаляти користувачів, змінювати права доступу та налаштовувати різні параметри без потреби у складному програмному забезпеченні. Усе це здійснюється прямо через звичайний браузер, що значно спрощує процес адміністрування.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Розробка апаратної частини системи авторизації

Створення апаратної частини пристрою є одним із ключових етапів будь якого проекту, адже саме вона визначає здатність системи повністю виконувати свої функції та забезпечує основу для подальшої інтеграції програмного забезпечення. На етапі розробки апаратної складової відбувається перехід від теоретичної моделі до реального фізичного виробу, який має відповідати технічному завданню, вимогам до продуктивності, енергоефективності, надійності та зручності експлуатації.

Для успішної реалізації задуму необхідно на початку проаналізувати вимоги до пристрою, ознайомитися з існуючими рішеннями, оцінити доцільність тієї чи іншої апаратної реалізації з точки зору доступності компонентів та подальшого обслуговування. Розглядалася також можливість майбутнього розширення функціональності для цього на етапі проектування важливо врахувати залишкові ресурси, наявність резервних каналів зв’язку, достатню кількість портів введення виведення.

Розробка апаратної частини зазвичай включає кілька послідовних етапів. Один із перших кроків є формування концепції системи та визначення її структурної схеми. Тут визначаються основні функціональні блоки, типи мікроконтролерів, модулів, інтерфейсів і джерел живлення.

Далі здійснюється деталізація внутрішньої організації пристрою, підбір відповідних комплектуючих, розробка електрично-принципової схем з урахуванням усіх особливостей роботи компонентів.

Важливо, що розробка апаратної частини складається не лише з теоретичного проектування, а й із практичних операцій із фізичними матеріалами: виготовлення друкованої плати, монтажу, пайки елементів та перевірки працездатності зібраної конструкції. На кожному з етапів критично важливо дотримуватися технологічної дисципліни з обробки текстоліту, правильного розташування елементів, щоб домогтися стабільного та довговічного пристрою.

1.1.1 Проектування плати-передавача

Під час створення плати-передавача система авторизації, насамперед, орієнтувалася на простоту, надійність і універсальність майбутнього пристрою. На цій стадії важливо було не тільки правильно підібрати потрібні електронні компоненти, але й забезпечити їхню взаємодію у межах однієї компактної й зручної для розведення плати. Завдяки цьому вдалося добитися максимальної стабільності пристрою, зручності обслуговування та повторюваності конструкції при виробництв

Ключовою частиною в конструкції передавача обраний перевірений контролер з підтримкою ESP8266-12F і радіомодуль NRF24L01. Саме такий тандем забезпечує найкраще співвідношення функціональності і ціни, у поєднанні із доступністю на ринку, простотою схеми та інтуїтивною логікою підключення.

Особливість проекту полягає у тому, що саме плата-передавач є головним вузлом, який активно передає дані по радіоканалу до приймача. Під час проектування важливо було обрати саме такі компоненти, які дають змогу досягнути максимальної сумісності з майбутнім програмним забезпеченням, а всі рішення приймалися з урахуванням подальшої масштабованості та модернізації системи. Через високу популярність цих модулей серед ентузіастів, можна легко знайти необхідні бібліотеки або рекомендації по цим модулям.

ESP8266-12F - це основа передавача. Цей модуль є просунутою версією модуля ESP8266-12 з узгодженим вихідним каскадом та антеною. Виділяється гнучкими портами GPIO, простотою включення у будь-які схеми, підтримкою УАРТ, SPI, I2C, а також режимів глибокого сну для енергозбереження. В цьому проекті було розглянуто використання інтерфейсу підключення SPI для модулів та ESP8266-12F та NRF24L01. Контролер та радіомодуль працюють виключно при напрузі 3.3V, в обв’язці обов’язково застосовується малошумний стабілізатор (наприклад, AMS1117-3.3 або його аналоги), а також додатковий конденсатор 16V 47uf, який буде захищати пристрій від пікових навантажень на лінії живлення.

Для плати-передавача знадобиться вбудована та оперативна пам’ять контролеру ESP8266-12F(рисунок 1.1).

Рисунок 1.1 Зовнішній вигляд контролеру ESP-8266-12F

Стандартна розводка контактів ESP8266-12F дає змогу підключати як різноманітні цифрові компоненти, так і програмувати плату завдяки окремим BOOT та RESET-контактам. Це значно спрощує працю під час тестування й оновлення прошивки.

Таблиця 1.1 Основні характеристики модулю ESP8266

Процесор	**Tensilica L106 32-bit, 80/160 МГц**
Оперативна пам’ять	80 КБ
Flash пам’ять	4 МБ (32 Мбит)
Живлення	3.3В
Кількість GPIO	12
Wi-Fi	IEEE 802.11 b/g/n до 72.2 Мбіт/с
Інтерфейси	UART, SPI, I2C, I2S
Розміри	24,5 x 16 мм
Робоча частота	2,4 ГГц

Другим важливим модулем на платі-передавачі є NRF24L01 (рисунок 1.2). Він відповідає за обмін даними між пристроями на невеликих відстанях. Його інтерфейс SPI дозволяє легко й надійно під'єднати модуль до ESP8266 та керувати передачею інформації із мінімальними затримками.

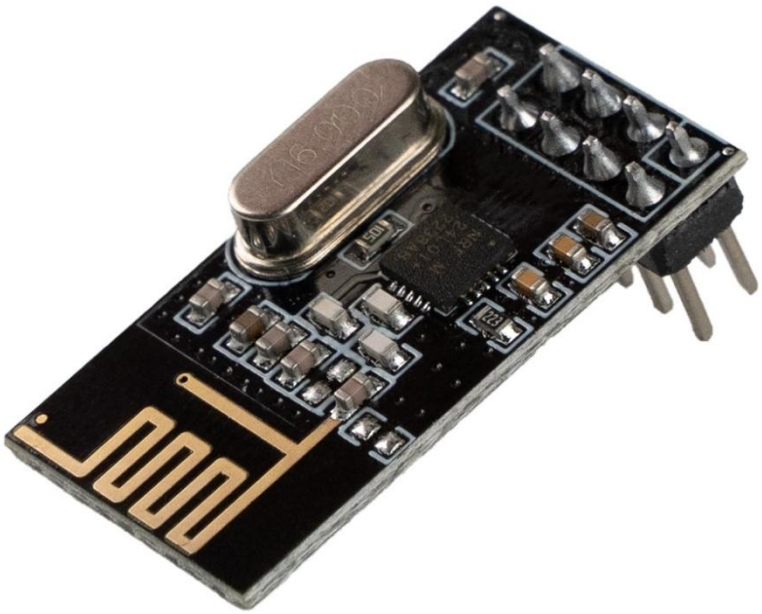


Рисунок 1.2 Зовнішній вигляд модуля NRF24L01

Для забезпечення стабільної роботи радіомодуля у схемі обов'язково додається додатковий конденсатор на 47 мкФ між VCC та GND. Це необхідно для зменшення ризику збоїв модуля при короткочасних стрибках навантаження, що особливо актуально для живлення від мобільних джерел (акумуляторів чи батарейок).

Таблиця 1.2 Основні технічні характеристики NRF24L01

Робоча напруга	3.3В
Інтерфейс	SPI
Дальність зв’язку	До 30м
Швидкість передачі	250 kbps, 1 Mbps и 2 Mbps

Усі підключення та компоненти для плати-передавача були ретельно продумані та промодельовані у сучасній середовищі розведення друкованих плат KiCad. Ця програма дає змогу розмістити компоненти оптимально для подальшого монтажу, забезпечити мінімальні довжини критичних ліній живлення та сигналу, звести перешкоди до мінімуму і пришвидшити етап виробництва.

В кінцевій принципово-електричній схемі інтегровані всі основні вузли: ESP8266-12F, NRF24L01, живлення, стабілізатори, елементи фільтрації, керуючі кнопки RESET та BOOT (рисунок 1.3).

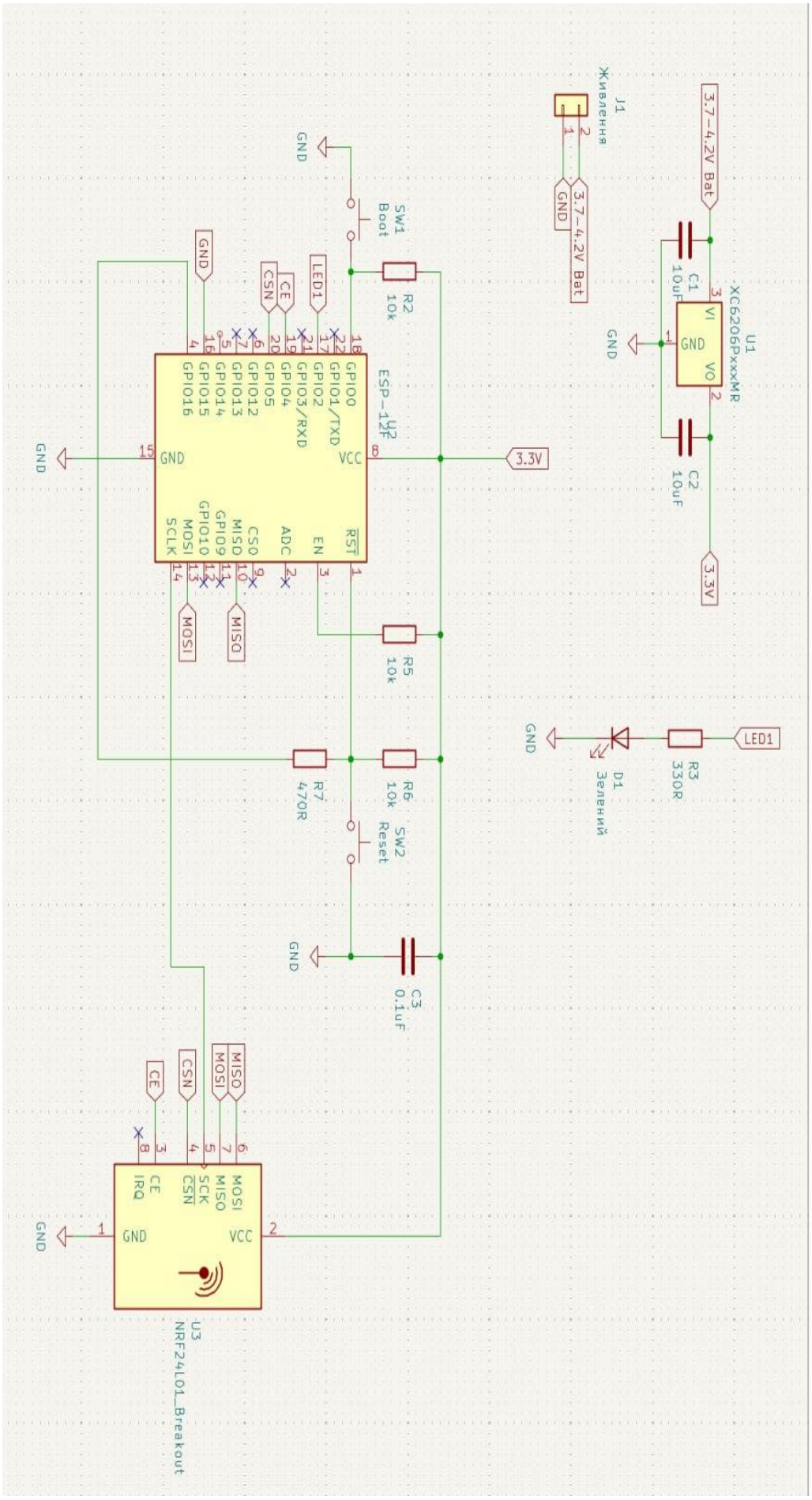


Рисунок 1.3. Принципово-електрична схема плати-передавача

1.1.2 Виготовлення та збірка плати-передавача

На першому етапі для виготовлення плати було обрано односторонній текстоліт, покритий шаром міді (рисунок 1.4). Від правильного вибору цього матеріалу залежить не лише якість майбутнього виробу, а й простота виконання усіх наступних технологічних операцій. Необхідно підібрати товщину текстоліту та якість мідного покриття: для компактних пристроїв оптимально використовувати плату завтовшки 1-1,5 мм.

Рисунок 1.4 Односторонній текстоліт, покритий шаром міді

Перед розміткою лист було очищено в відповідній частині за розмірами від пилу та слідів окислення за допомогою дрібнозернистої шкірки для подальшого якісного нанесення малюнка доріжок та покращення адгезії тонера. Далі робиться попереднє креслення контурів майбутньої плати на поверхні текстоліту, використовуючи олівець чи маркер. Важливо дотримуватися розмірів, що продиктовані схемотехнікою та розмірами корпусу пристрою. Заготовка після розмітки вирізається ножицями по металу або лобзиком. Важливо перевірити рівність країв: це сприяє коректному компонуванню пристрою в корпусі та запобігає утворенню задирок, які можуть пошкодити руки чи інші компоненти.

Наступним етапом для перенесу схеми провідників використовується лазерний принтер та спеціальний жовтий термотрансферний папір. Схема друкується на термопереносному папері. Використовувати необхідно саме

лазерний принтер, адже його тонер легко переноситься на мідь під дією високої температури. Для якісного малюнка варто обирати максимальну насиченість чорного кольору й переглядати, аби всі доріжки були чіткими, без переривань або пробілів.

Після друку папір необхідно обережно обрізати і підготувати до наступного переносу на текстоліт (рисунок 1.5). Якщо залишилися дрібні розриви на доріжках - їх можна підправити тонким маркером.

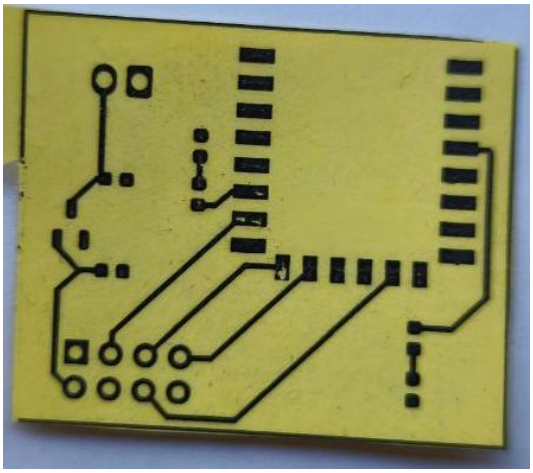


Рисунок 1.5 Підготовлений термопереносний папір з розміткою

Наступний етап - перенесення заготовленого малюнка на мідну поверхню текстоліту. Перш за все, мідну поверхню текстоліту необхідно ретельно знежирити та очистити - наприклад, спиртом або уайт-спірітом, а також дрібнозернистим наждачним папером до блиску, щоб тонер рівномірно прилипнув до міді та не було пропусків у доріжках. Далі вирізаний малюнок акуратно прикладається стороною з надрукованим тонером до міді. Важливо щоб шаблон ніде не ворушився, бо від цього залежить точність всіх доріжок.

Далі необхідно використовувати нагрівальний прилад, який необхідно розігрівати до 190 °C. Слід починати нагрівати шаблон через папір, рівномірно притискаючи всю поверхню (рисунок 1.6). Оптимальний час нагріву складав приблизно 5 хвилин: за цей час тонер практично повністю переходить з паперу на мідь, утворюючи точний малюнок майбутніх доріжок. Під час нагріву не слід поспішати: краще притискати праску поступово, затримуючись на кожній ділянці по кілька секунд, щоб забезпечити рівномірний розподіл температури.

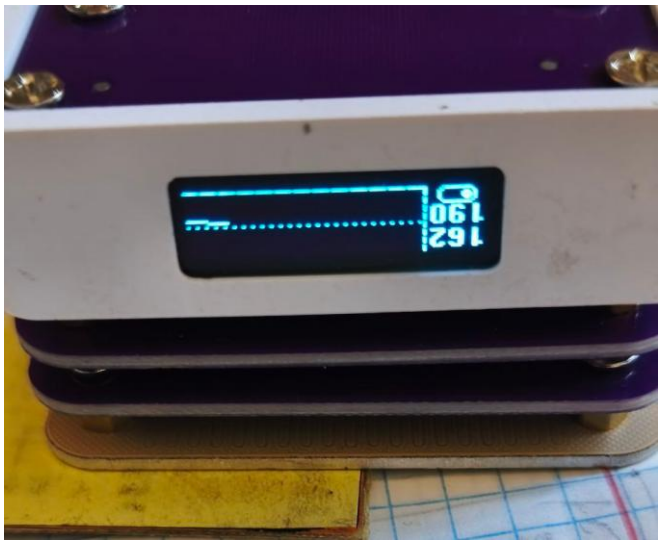


Рисунок 1.6 Прогрівання текстоліту з термопереносним папіром

Після завершення нагріву заготовки, потрібно дати їй повністю охолонути, а потім акуратно занурити у теплу воду на декілька хвилин, щоб папір розмок. Потім папір обережно скочується: залишиться лише шар тонеру, який точно повторить всі доріжки схеми (рисунок 1.7).

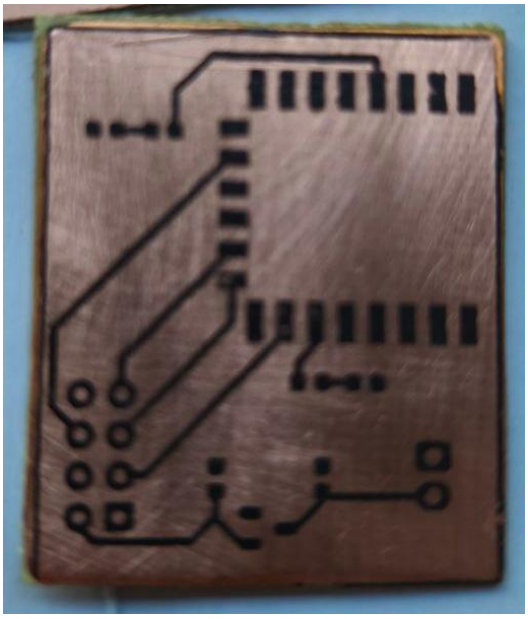


Рисунок 1.7 Готовий шаблон з шаром тонеру на доріжках

Закріплений малюнок дозволяє перейти до етапу травлення. Підготовлена плата поміщається у ванночку з розчином Залізо III Хлорид (Рисунок 1.8). В спеціальну ємність насипаємо хлорне залізо та додаємо воду розігрітою до температури 50-60°C. В процесі травлення незахищена мідь розчиняється, залишається лише малюнок провідників.

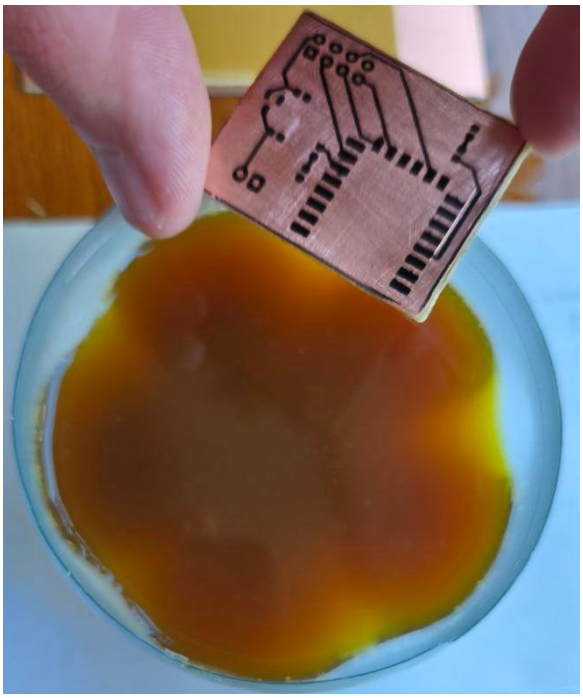


Рисунок 1.8 Травлення плати у розчині хлорного заліза

Для контролю швидкості травлення необхідно періодично діставати плату пінцетом й оцінювати ступінь видалення міді на відкритих ділянках. Після завершення цього етапу залишки лаку або маркера змивалися ацетоном чи спиртом. Очищена і висушена плата готова до подальших дій (рисунок 1.9)

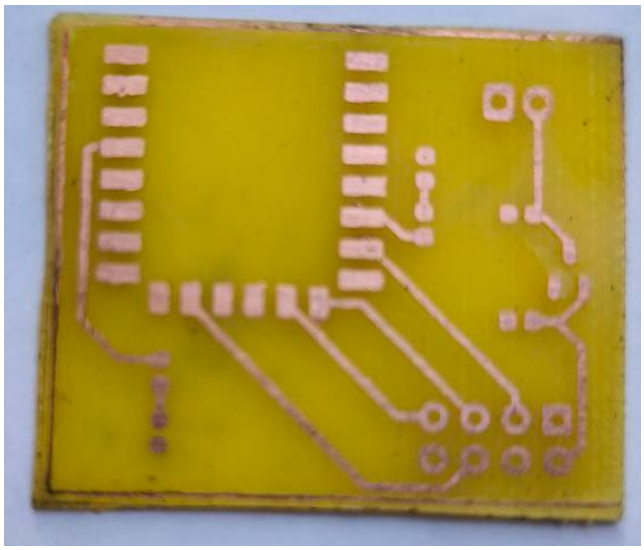


Рисунок 1.9 Витравлена плата з залишками мідного шару на доріжках

З метою підвищення провідності та захисту мідних доріжок від окислення проводилося лудіння розчином флюсу та припою. Це суттєво спрощує процес пайки та продовжує термін служби плати. Заключний етап - безпосередньо встановлення всіх компонентів згідно з розробленою електричною схемою та їх

пайка. Починати треба із монтажу найменших та малогабаритних елементів, поступово переходячи до більших (рисунок 1.10). Паяння виконувалося за допомогою електропаяльника середньої потужності з тонким жалом, що дозволяє уникнути замикань і забезпечити акуратність з’єднань.

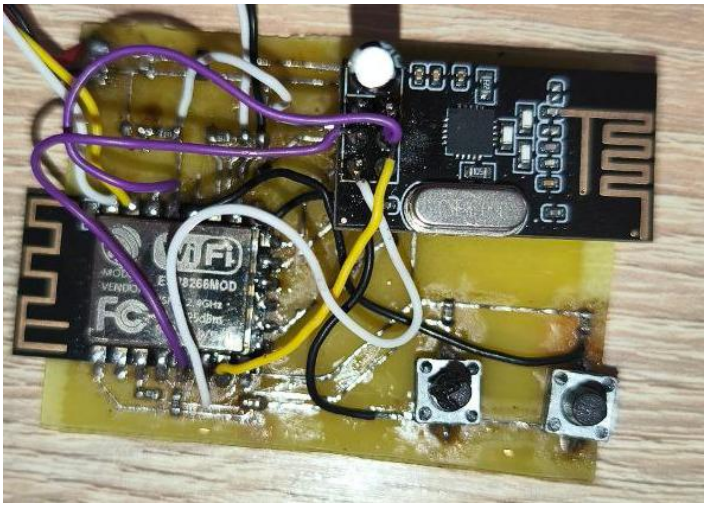


Рисунок 1.10. Зібрана плата-передавача

1.1.3 Проектування та виготовлення плати-приймача

Під час розробки приймача для нашої системи авторизації першочергова увага приділялася правильному вибору основної апаратної платформи, що повинна була забезпечити не лише високу стабільність роботи, а й простоту обслуговування та гнучкість у майбутньому розширенні системи. На відміну від передавача, де використовувався ESP8266-12F, у цій частині проекту основним модулем обрано сучасну плату розробника - ESP-8266-12E NodeMCU V3 (рисунок 1.11).

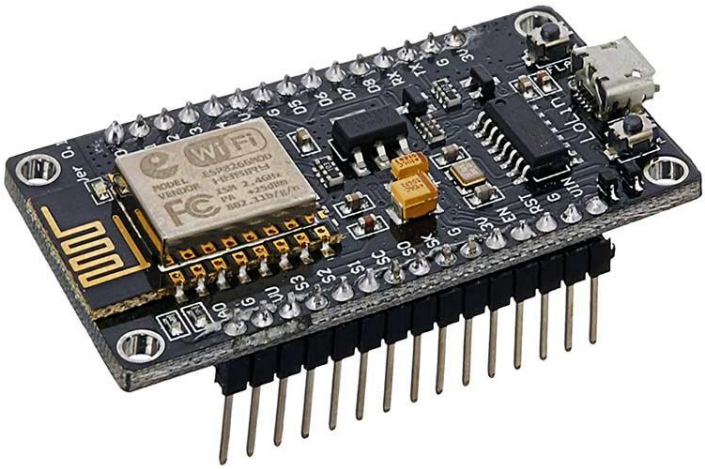


Рисунок 1.11. Плата розробника ESP-8266-12E NodeMCU V3

Під час розробки приймача для нашої системи авторизації першочергова увага приділялася правильному вибору основної апаратної платформи, що повинна була забезпечити не лише високу стабільність роботи, а й простоту обслуговування та гнучкість у майбутньому розширенні системи. На відміну від передавача, де використовувався ESP8266-12F, у цій частині проекту основним модулем обрано сучасну плату розробника - ESP-8266-12E NodeMCU V3.

На платі вже реалізовані USB-інтерфейс для підключення до комп'ютера, стабілізатор живлення, а також усі виводи чіпа виведені на гребінки зі стандартним кроком 2.54 мм. Завдяки цьому її зручно використовувати з макетними платами без потреби в пайці. Окрім того, вона вже містить прошивку NodeMCU, що дає змогу програмувати її мовою Lua або через середовище Arduino IDE. Схематична структура цієї плати значно полегшує не тільки реалізацію прошивки, а й подальшу експлуатацію, оскільки всі процедури Reset/Boot вже автоматизовані і достатньо одну кнопку для скидання або перепрошивки.

Для радіозв'язку із передавачем використовується все той же радіомодуль NRF24L01. Також для максимально стабільної роботи NRF24L01 на платі-приймачі також встановлено електролітичний конденсатор 47 мкФ безпосередньо між VCC і GND живленням nrf-модуля, як зображено на схемі (рисунок 1.12).

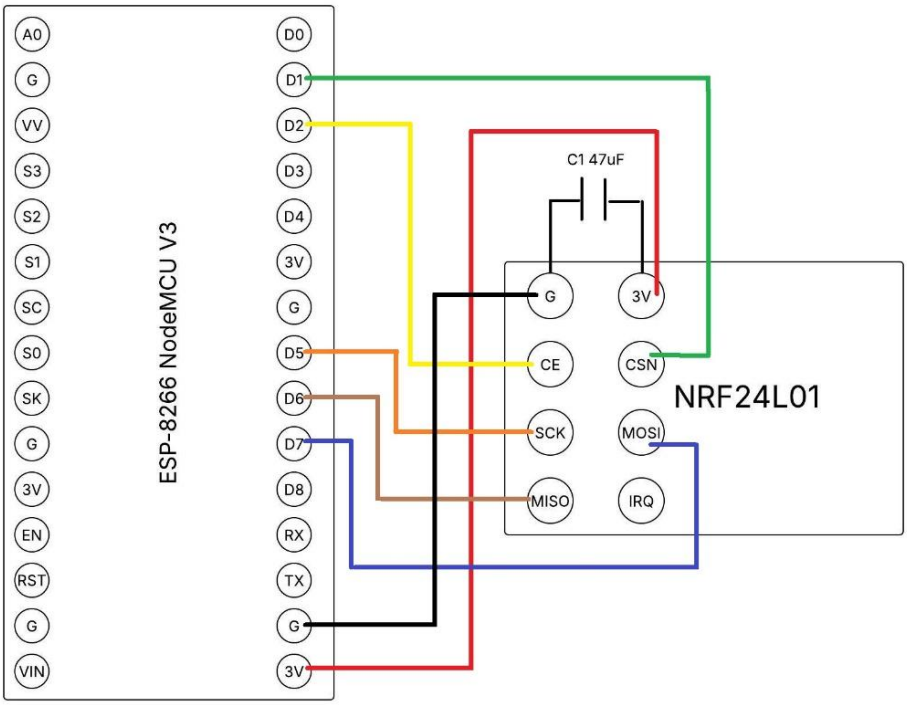


Рисунок 1.12. Принципова схема підключення плати-приймача

Згідно зі схемою розпіновки, акуратно лудимо виводи та проводимо монтаж дротів або фіксуємо модулі через роз’єм або безпосередньо на плату. Для пайки використовуємо тонке жало паяльника і якісний флюс - так можна уникнути випадкових замикань між сусідніми доріжками чи пінів. Важливо перевірити контакти, по яких ідуть основні сигнали SPI між NodeMCU та NRF24L01: MOSI, MISO, SCK, CSN, CE. Всі вони повинні бути чітко підключені згідно зі схемо. Коли всі елементи на платі встановлені, та пайка завершена, ретельно оглядаємо результат під лупою чи збільшувальним склом. Так виглядає готова плата-приймач (рисунок 1.13).

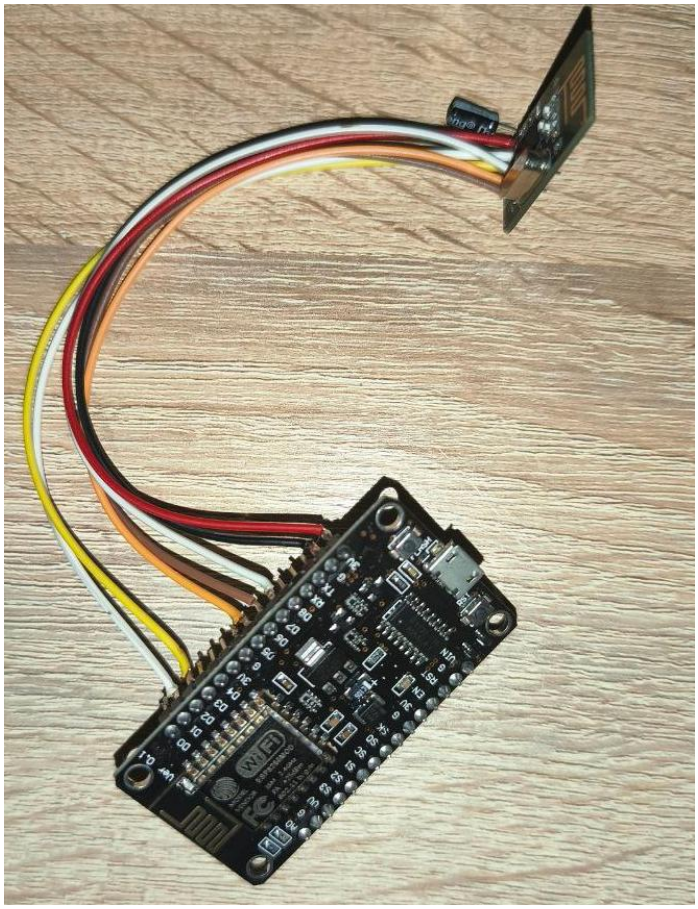


Рисунок 1.13 Готова плата-приймач

1.2 Програмна реалізація пристроїв користувача

Після того як апаратна складова системи була спроектована та зібрана, постає новий комплекс завдань - пов’язаних із програмною реалізацією пристроїв. Вона має ключове значення для коректної та надійної роботи всієї системи авторизації. Адже навіть ідеально виготовлене апаратне забезпечення без

правильного програмного супроводу не зможе повністю реалізувати свій потенціал. Програма визначає логіку взаємодії різних модулів, контролює обробку даних, забезпечує своєчасний обмін інформацією та реагування на дії користувача.

У цьому розділі розглядаються загальні підходи до написання програмного забезпечення для обох основних компонентів: плати-передавача та плати-приймача. Окремо буде висвітлено особливості розробки для кожного з них, оскільки вони мають свої специфічні завдання й унікальну роль у структурі системи.

1.2.1 Розробка програмного забезпечення для плати-передавача

Сьогодні бездротові технології стали невід'ємною складовою практично будь-якої системи автоматизації, від простих розумних будинків і контролю доступу до промислових IoT-рішень і навчальних стендів. Майже скрізь виникає потреба у пристроях, здатних автономно передавати ідентифікатори або службові дані по повітрю з мінімальним струмоспоживанням і максимальною надійністю. Саме таку задачу - створення передавача для бездротової передачі унікального ідентифікатора (UID) нам і довелося вирішити.

Передавач можна використовувати у значній кількості різних сфер: Системи контролю доступу та обліку робочого часу; Автоматизація обліку відвідуваності, фіксації присутності у будь-яких приміщеннях; IoT-системи для інвентаризації чи визначення унікальних пристроїв у мережі; Навчальні проекти та дослідницькі зразки для вивчення роботи радіо та ідентифікаційних систем. У цьому спектрі передавач UID - це свого роду цифровий бейдж, який регулярно або за подією (наприклад, натиском кнопки) повідомляє центральній системі, хто зараз є у зоні дії радіоканалу.

Плата-передавач у цій системі виконує ключову роль - вона є мобільним джерелом ідентифікаційної інформації, що дозволяє безпечно та швидко передавати унікальний ідентифікатор (UID) іншій стороні, а саме - платі-приймачу. По суті, це електронний ключ, який може діяти як бейдж співробітника,

маркер предмету чи мобільний токен для контролю доступу. В основі роботи лежить ідея забезпечити безперервний або подієвий обмін UID через радіоканал, використовуючи протокол зв’язку nRF24L01, що гарантує низьке енергоспоживання, високу надійність та хорошу дальність дії навіть у складних умовах.

Сам алгоритм роботи плат зводиться до того, що мікроконтролер (ESP8266 з підключеним модулем nRF24L01) періодично формує та надсилає по радіоканалу фіксований UID, закладений у коді прошивки. Передача здійснюється у циклі, з невеликою затримкою, аби не спричинити перевантаження ефіру і забезпечити коректне приймання зі сторони приймача. В такій архітектурі передача UID може відбуватись незалежно від зовнішніх тригерів; втім, у разі потреби, логіку можна легко змінити, щоб відсилання відбувалось, скажімо, натисканням кнопки чи зовнішнім сигналом.

Окремої уваги заслуговує питання апаратного з’єднання компонентів. nRF24L01 - це SPI-периферія, а тому для її коректного підключення до ESP8266 необхідно виділити чітко визначені піни для SPI-шини, серед яких CE та CSN виконують керуючу функцію. У приведеному прикладі за CE вибрано GPIO4, а CSN підключено до GPIO5, що зазначено на старті програми відповідними #define. Саме ці рядки коду чітко задають, куди підключати фізичні контакти nRF-модуля, уникнути плутанини при монтажі та забезпечити стабільність у серійному виробництві чи тестуванні.

Ще на етапі ініціалізації коду, після підключення Serial для зневадження, встановлюється основна конфігурація nRF24L01 через функції radio.begin, radio.openWritingPipe та radio.setPALevel. Важливим є відкриття каналу передачі даних з уніфікованою адресою - це саме той ідентифікаційний ефірний канал, через який листується тільки потрібний (свій) приймач. Значення цього поля жорстко задається в масиві address, щоб уникнути перехресних завад між кількома незалежними системами поблизу.

Основне навантаження у циклі роботи бере на себе функція radio.write, вона якраз і відповідає за фактичний викид UID у радіоефір. Код виконує надсилання

відповідного масиву символів та одразу дає підтвердження у Serial для подальшого моніторингу. Щоб уникнути надмірної передачі та дати достатній інтервал для фіксації на приймачі, після кожної спроби накладається затримка на 10 секунд. Такий підхід не лише економить енергію, якщо пристрій живиться від батарейки, а й робить роботу системи більш передбачуваною при одночасній роботі кількох передавачів.

Нижче приведена блок схема алгоритму роботи прошивки для плати-передавача (рисунок 1.14):

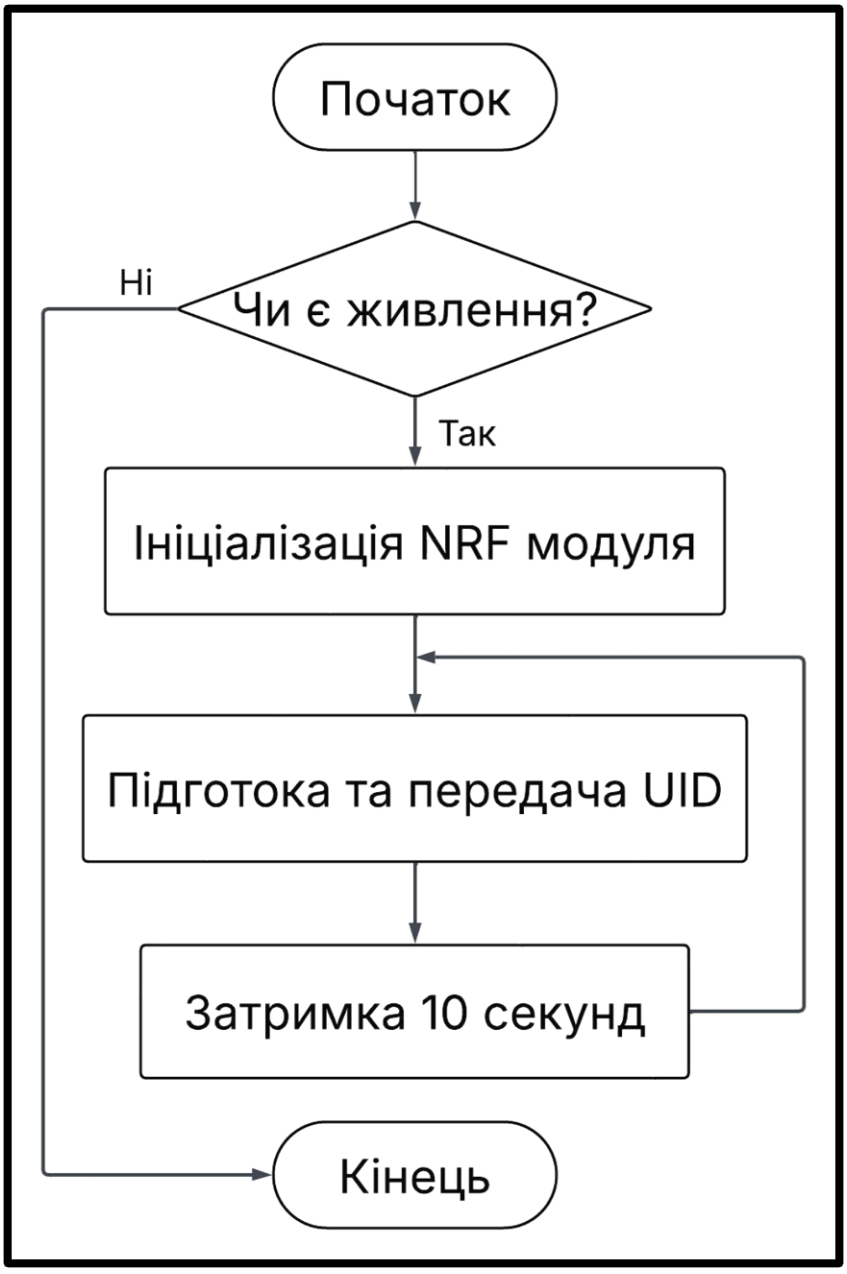


Рисунок 1.14 Схема послідовності роботи програмного коду плати-передавача

Програмне забезпечення плати-передавача поєднує у собі як простий, так і дуже стійкий механізм бездротового обміну ідентифікаторами UID. Логіка чітко розмежована: фізичне підключення кожної ноги визначено у коді, налаштовано канал для зв'язку, параметри радіоканалу налаштовані для мінімізації колізій, а періодичність передачі UID обирається з урахуванням практичних потреб і безпеки.

1.2.2 Розробка програмного забезпечення для плати-приймача

Частиною запропонованої системи бездротової ідентифікації є плата-приймач, яка здійснює прийом сигналів із передавача, обробку отриманої інформації й її подальшу передачу на сервер. Сама ідея використання приймача полягає в тому, щоб забезпечити надійну і максимально автоматизовану зустріч усіх переданих UID у конкретній точці доступу, наприклад, у місці входу на територію, складу або офісу. Приймач стає свого роду посередником між мобільними передавачами та центральною системою контролю.

Принцип роботи пристрою максимально автоматизований: головним його завданням є цілодобово відстежувати радіоефір, сприймати сигнали UID, що надходять із одного чи кількох передавачів, та гарантовано доставляти ці UID до бекенд-системи через мережу WiFi. Сам механізм реакції на сигнал досить гнучкий: надходження UID миттєво запускає алгоритм передачі інформації далі, щоб система могла віддати результат авторизації чи виконати якісь дії у реальному часі.

Головна технічна особливість приймача - вдала синергія двох засобів комунікації. З одного боку, модуль nRF24L01 відповідає за прийом і розпізнавання UID по радіоканалу у виділеному частотному діапазоні, а з іншого мікроконтролер на ESP8266 дозволяє оперативно відправити отримані дані на сервер через WiFi за допомогою сучасного HTTP API. Такий підхід поєднує простоту розгортання з високою гнучкістю: плата може працювати у будь-якій мережі, де є WiFi, і так само легко перебудовуватись на іншу інфраструктуру.

Як і на передавачі, на етапі ініціалізації важливо правильно налаштувати підключення модулів. У коді видно, що SPI-піни налагоджено через CE на GPIO4

і CSN на GPIO5 - саме такі значення прописано в #define CE_PIN та CSN_PIN на самому початку програми. Це забезпечує гарантовану коректність підключення апаратної частини до ESP8266, мінімізуючи проблеми сумісності між платами у випадку повторення чи масштабування проекту. Використання ідентичних адресних налаштувань модуля nRF24L01, що і на передавачі (даний приклад address 00001), забезпечує передачу UID тільки між своїми пристроями, скорочуючи ймовірність збоїв при наявності декількох бездротових систем поблизу.

Весь механізм зв’язку та обробки поділений на кілька логічних етапів. Після запуску і підключення до WiFi, про що у серійному моніторі сигналізується фраза WiFi підключено., система негайно переходить до режиму прослуховування радіоефіру. Це встановлюється викликами radio.begin, radio.openReadingPipe і radio.startListening у такий спосіб модуль постійно готується до прийому нових UID. В ибір PALevel у режимі LOW, оптимальний для внутрішніх приміщень, коли не потрібна надвелика дальність, а головне мінімізація енергоспоживання та завад.

Процес прийому UID організовано через нескінченний цикл, у якому постійно перевіряється, чи з’явились нові дані radio.available(). Якщо був отриманий сигнал, то UID негайно зчитується в буфер символів, а сама подія фіксується у порті Serial, що надзвичайно зручно для діагностики і відлагодження на етапі тестування і запуску системи.

Подальший етап відправка UID на сервер через HTTP POST. Тут ESP8266 використовує сучасний синтаксис бібліотеки ESP8266HTTPClient, формуючи компактне JSON повідомлення й надсилаючи його за вказаною адресою сервера. Кожна спроба передачі логуються, а відповіді від серверу виводяться у монітор, що дозволяє не лише контролювати стабільність зв’язку, а й негайно реагувати у випадку збоїв чи непередбачених ситуацій із мережею. Якщо на момент чергової спроби Wi- Fi з якихось причин у мережі не присутній, програма спочатку сповістить у port Serial про цю проблему, не намагаючись марно надсилати дані, а просто чекатиме покращення зв'язку.

Для запобігання надмірного навантаження на сервер чи бездротовий інтерфейс, після кожної передачі даних застосовується затримка. Це дозволяє уникнути спаму із повторюваних UID у ситуаціях, коли передавач перебуває довго у зоні дії приймача, й гарантує стабільність роботи системи навіть при великій кількості паралельних зчитувань.

Платформа ESP8266 NodeMCU V3, яку використано в основі пристрою, спроектована так, щоб можна було швидко змінювати параметри роботи через прошивку, підключати додаткові модулі або впроваджувати нові сценарії взаємодії з сервером. Наприклад, у міру розширення системи і збільшення кількості точок контролю, можна організувати мережу з кількох приймачів, які спільно працюють з одним сервером.

У програмному коді передбачена можливість журналювання та діагностики всіх подій. Серійний вивід кожного етапу, від отримання UID до результату HTTP-запиту, допомагає розробнику і адміністратору бачити, як поводиться система в реальних умовах. Це корисно на етапі тестування й підтримки, коли оперативний відгук дозволяє швидко виявити й усунути неполадки. Наприклад, при зникнені сигналу Wi-Fi, якщо приймач отримує невірний UID, або якщо сервер тимчасово недоступний.

У режимі очікування й прийому сигналів споживання енергії пристроєм мінімальне, що дозволяє утримувати постійний контроль при мінімальних витратах, а також розташовувати приймачі у важкодоступних місцях або навіть використовувати резервне живлення в автономних системах. Багаторазові перевірки радіоканалу та перевірка статусу Wi-Fi гарантують, що жоден важливий UID не буде втрачений, навіть у разі короткочасних збоїв мережі.

Формат передачі даних легко піддається зміні за необхідності можна налаштувати відправку додаткової інформації, прописати унікальні ідентифікатори самого приймача чи навіть додати штамп часу, підвищуючи рівень деталізації обліку даних. Відкритість програмного коду весь цей функціонал може бути легко розширений.

Блок-схема алгоритму демонструє повну роботу приймача (рисунок 1.15):

Рисунок 1.15 БСА роботи плати-передавача

1.3 Проектування та реалізація web-сервера та інтерфейсу

Сучасна система контролю доступу чи ідентифікації нерозривно пов’язана із використанням web-технологій, які забезпечують зручну та ефективну організацію обміну даними, обліку та візуалізації інформації для користувачів і адміністраторів. Саме серверна частина виступає центральною ланкою в усій архітектурі: вона приймає сигнали від апаратної частини, обробляє дані і забезпечує їх збереження чи подальшу обробку. Разом із тим, web-інтерфейс виконує ключову роль обличчя для системи, надаючи змогу в реальному часі спостерігати за подіями, здійснювати контроль та керування певними процесами, а також отримувати аналітику чи звіти.

У цьому розділі розглядається питання проектування та реалізації web-сервера і користувацького інтерфейсу. Основна увага приділятиметься загальним принципам побудови серверної логіки та створенню зручного інтерфейсу для взаємодії з системою, що разом формують єдину інформаційну платформу для подальшої автоматизації, обліку й контролю.

1.3.1 Реалізація серверної частини

В умовах сучасної автоматизації та цифровізації більшість систем ідентифікації й контролю доступу не можуть обійтися без надійної серверної частини, яка виконує одразу кілька критично важливих функцій. Саме веб-сервер стає ядром, що поєднує в собі апаратні пристрої - приймачі й передавачі із зручною й зрозумілою для людини інфраструктурою, забезпечуючи головний канал для обробки, зберігання та подальшої верифікації отриманої інформації. Роль сервера полягає у тому, щоб кожен унікальний ідентифікатор, надісланий з пристрою прийому, не лише потрапляв у єдину точку обліку, а й міг бути перевірений або ж зареєстрований і використаний у майбутніх сценаріях авторизації.

Реалізація серверної частини у цьому проекті базується на сучасному, легкому фреймворку Flask, обраному завдяки простоті його інтеграції, можливості гнучко налаштовувати маршрути та швидко розгортати потрібні

програмні ендпоінти. Flask відмінно підходить для прототипування та розробки систем, де важливо підтримувати просту логіку, але водночас залишати простір для розширення або впровадження нових сценаріїв у майбутньому.

Бібліотека Flask - це своєрідна точка входу до світу створення сучасних web-додатків на Python. Flask часто обирають розробники, коли потрібно оперативно та просто підняти власний сервер або API для обміну даними між пристроями. Перевагою цього фреймворку є його мінімалізм і легкість: Flask не нав'язує жорстких структур і дозволяє буквально за кілька рядків коду отримати робочий web-сервер з усіма базовими можливостями.

Щоб почати працювати з Flask, досить виконати стандартну установку за допомогою Python-пакетного менеджера pip - команда pip install flask за лічені секунди завантажує всі потрібні залежності й дозволяє одразу імпортувати бібліотеку у свій проект. Далі все надзвичайно інтуїтивно: визначається роутинг, описуються функції, які відповідають на HTTP-запити, і буквально в два рядки запускається сервер, який слухає визначений порт. У Flask легко додавати авторизацію, обробляти JSON, реалізовувати складні логічні сценарії або навіть оформлювати повноцінний веб-інтерфейс із динамічними сторінками.

Якщо розглядати принцип роботи серверної частини більш детально, то дуже важливою є зовнішня взаємодія між приймачем і сервером на рівні HTTP-запитів. Приймач, отримавши новий UID від передавача, надсилає POST-запит на визначену адресу http://<IP-сервера>:5000/add_uid. Тут, у спеціальному ендпоінті /add_uid, сервер приймає дані у форматі JSON, витягає UID та фіксує його як останньо отриманий. Запис відбувається безпосередньо у файл на сервері - це простий, але надійний спосіб організувати локальне короткострокове зберігання інформації, особливо у невеликих системах або на етапі розробки.

Реалізація збереження UID у файлі виглядає логічною, оскільки дозволяє суттєво спростити перевірку останньої операції та залишає можливість легкого резервного копіювання критично важливих ідентифікаторів. Для подальшої інтеграції розроблено окремий механізм перевірки та авторизації UID: сервер містить ендпоінти для збереження UID як зареєстрованого, порівняння вхідних та

вже зареєстрованих UID, а також видалення даних у разі потреби. Це дозволяє будувати просту, зрозумілу й надійну логіку авторизації, яка легко масштабується залежно від побудови майбутнього веб-інтерфейсу.

Весь цей функціонал прописаний у відповідних функціях, які відповідають за різні аспекти обробки UID. Наприклад, функція add_uid не просто приймає дані, вона одразу ж проводить валідацію, переконується, що UID надано, і тільки після цього фіксує його у файл. Це важливо з точки зору захисту від помилкових запитів і гарантує цілісність даних, які надходять безпосередньо з пристрою прийому.

Всі запити від пристрою відбуваються через сучасні HTTP POST-запити із вмістом типу JSON, завдяки чому система є дружньою до інтеграції з іншими компонентами й дозволяє розширювати функціонал у майбутньому, наприклад, додавати нові параметри або ідентифікатори без суттєвих змін у базовій архітектурі.

Сервер, побудований на Flask, дуже зручний з точки зору моніторингу - усі події та маніпуляції з UID фіксуються як у консолі, так і у відповідних файлах. Такий підхід полегшує етапи тестування, дозволяє в реальному часі бачити стан системи та контролювати взаємодію між різними її компонентами.

Гнучкість самого проекту - використання бібліотеки Flask залишає простір для побудови масштабованих систем. Коли кількість підключених приймачів або користувачів зростатиме, сервер можна легко перенести на більш потужну платформу або в хмару, реалізувати зберігання UID у реальній базі даних, розгорнути захищений канал зв’язку через HTTPS, а в разі необхідності інтегрувати систему зі сторонніми сервісами чи мобільними додатками. Тобто навіть базове програмне рішення є міцною основою для подальшої модернізації та розвитку.

Серверна частина будується так, щоб кожен запит оброблявся швидко, а відповідь була зрозумілою і дружньою як для людини, так і для підключеного пристрою.

Простота розгортання й експлуатації такого сервера дозволяє адміністраторам швидко вводити нові UID у базу, а розробникам - без зайвих

складнощів оновлювати код і впроваджувати новий функціонал без ризику порушити вже стабільну роботу основних процесів.

1.3.2 Розробка web-інтерфейсу користувача

Веб-інтерфейс користувача у контексті системи контролю UID виконує надзвичайно важливу функцію: він перетворює всю невидиму роботу апаратної та серверної частини у просту, зрозумілу та візуально зручну взаємодію для адміністратора чи кінцевого користувача. Завдяки інтерфейсу стає можливим не лише моніторинг поточного стану системи в реальному часі, а й активне керування ключовими процесами - додавання, авторизація чи видалення UID відбуваються буквально в кілька кліків, без потреби заглиблюватися у технічні деталі чи команди на сервері.

Принцип роботи веб-інтерфейсу будується на тісній інтеграції з серверною частиною через HTTP-запити. Кожна дія користувача, наприклад натискання кнопки Додати UID чи Авторизація, автоматично ініціює запити до відповідних маршрутів Flask-сервера. Сам інтерфейс розроблено з використанням стандартних технологій веб-розробки: HTML для структури контенту, CSS для створення чистого та сучасного вигляду, і JavaScript, який забезпечує динаміку, реактивність та зв’язок із сервером.

У базі цього веб-інтерфейсу лежить асинхронна, жива взаємодія із сервером через fetch-запити. Наприклад, при завантаженні сторінки JavaScript одразу звертається до ендпоінтів /uid та /last_uid для отримання актуальної інформації про зареєстрований UID та останній UID, який щойно надійшов від приймача. Такою парою асинхронних звернень оновлюється й відображається фактичний стан системи - користувач бачить, який саме UID наразі зареєстровано та чи очікується надходження нового від апаратної частини. Операція saveUID() реалізує POST-запит до /save_uid, і одразу після відповіді сервера на сторінці відображається результат - користувач не лише бачить повідомлення про успіх чи помилку, а й одразу отримує оновлену інформацію про всі UID.

Завдяки функції updateUIDs() дані про UID оновлюються автоматично кожні кілька секунд, тому стан системи завжди актуальний без необхідності

постійно перезавантажувати сторінку. Такий підхід максимально наближує веб-інтерфейс до живого спілкування з апаратурою: якщо приймач передав новий UID, це майже миттєво з’являється на екрані оператора.

Код JavaScript ретельно продуманий з погляду як функціональності, так і зрозумілості для майбутньої модифікації. Наприклад, у функції authorizeUID() після натискання кнопки відправляється запит на сервер, і відповідь кольором і текстом сигналізує про результат спроби авторизації: зелений колір означає успішну відповідність UID, а червоний про невдачу. Це дозволяє адміністратору одразу бачити підсумок без аналізу логів чи технічних повідомлень.

Використання JavaScript для створення fetch-запитів робить інтерфейс універсальним - він працюватиме практично на будь-якому сучасному браузері і не потребує складного налаштування сторонніх бібліотек чи фреймворків. Усе оформлення реалізовано через прості стилі CSS, що забезпечує чистий і професійний вигляд без перевантаження зайвими деталями.

Через гарну організація веб-інтерфейсу не лише підкреслює високий рівень інтеграції між програмними й апаратними компонентами системи, але й забезпечує майже ідеальну простоту користування для людини різного рівня підготовки. Усі ключові операції, від додавання до авторизації та видалення UID виконуються інтуїтивно та зрозуміло. Гнучкість будови дозволяє за потреби розширити можливості інтерфейсу: наприклад, додати ведення журналу подій або більш складне налаштування прав доступу.

Важливо звернути увагу на аспект відгуку та гнучкості інтерфейсу при використанні на різних пристроях. Хоча базова реалізація створена для роботи на комп’ютері, структура HTML і підхід до стилізації дозволяють легко адаптувати сторінку для перегляду із планшетів і навіть смартфонів. Це актуально для сучасних систем, адже багато адміністраторів або користувачів здійснюють контроль чи моніторинг на ходу. Подібна мобільність відкриває більше варіантів використання системи у реальних умовах - наприклад, при виїзній перевірці або швидкому реагуванні на подію просто із телефону, без підключення до стаціонарного ПК.

Можливість доповнення інтерфейсу елементами зворотного зв’язку, що підвищує комфорт користування. Йдеться не лише про короткі текстові повідомлення, а й про автоматичне візуальне виділення активних статусів, відображення часу останнього оновлення або навіть звукові сигнали при виконанні важливих дій.

Логіка взаємодії між сторінкою та сервером побудована так, щоб залишатися максимально простою для налаштувань та оновлень у майбутньому. Наприклад, для розширення функціоналу достатньо додати новий ендпоінт на сервері й дописати відповідний блок у JavaScript - весь підхід залишається зрозумілим навіть для початківця. Такий web-інтерфейс є чудовою платформою для експериментів: можна легко вбудовувати статистику по UID, фільтри, розширені журнали подій або інтеграцію із зовнішніми сервісами.

Важливим етапом створення зручного web-інтерфейсу є розташування та оформлення функціональних кнопок. Вони знаходяться в самому центрі й мають зручний проміжок одна від одної, що позитивно впливає на ергономіку роботи: випадкові помилки при натисканні майже виключені, а необхідна дія завжди під рукою. Кожна кнопка має простий і однозначний підпис - Додати UID, Авторизація, Видалити UID, для того, щоб будь-який користувач може інтуїтивно зрозуміти, що й для чого призначено, без потреби заглядати в документацію чи запитувати допомоги.

Задля чіткого керування веб-серверу було створено заголовок Зареєстрований UID, яке одразу дає зрозуміти, що саме на цій сторінці можна управляти і контролювати основний елемент системи - унікальний ідентифікатор користувача або пристрою. Ще нижче, відображається поточний стан: наприклад, якщо жоден UID ще не був доданий чи збережений у систему, користувач бачить повідомлення UID не додано.

В веб-інтерфейсі реалізовано лише мінімально потрібний функціонал (рисунок 1.16). Завдяки цьому інтерфейс не перевантажений, абсолютно все зроблено на користь простоти та швидкості роботи. Користувачу не потрібно відволікатися на непотрібні налаштування, зайві повідомлення чи займатися

пошуком потрібної інформації серед великої кількості тексту. Всі повідомлення максимально короткі, а реакція системи миттєва: будь-які зміни відразу ж відображаються у центральному блоці сторінки. Наприклад, після успішного додавання UID він з’явиться замість попереднього повідомлення про його відсутність.

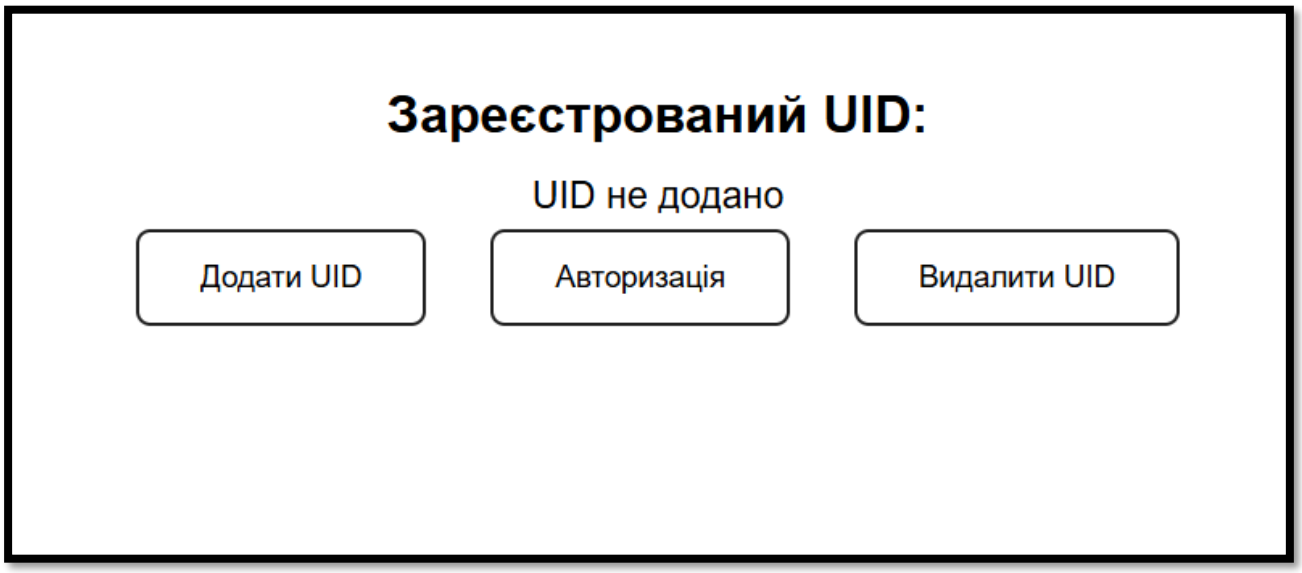


Рисунок 1.16 Web-інтерфейс для системи авторизації

1.4. Підготовка апаратного забезпечення до прошивки

Перед тим як розпочати практичну частину програмування пристроїв, важливо підготувати апаратну та програмну складові таким чином, щоб робота із платами відбувалась швидко, безпечно й злагоджено. На цьому етапі проект переходить із теоретичних напрацювань до реальних дій із живим обладнанням, і саме якість підготовки часто визначає, наскільки весь подальший процес буде стабільним та передбачуваним.

У рамках даної системи використовується встановлена Arduino IDE, зручне середовище для розробки, налаштування та прошивки мікроконтролерів. Основне завдання цього етапу - налаштувати середовище під вибрані плати, встановити відповідні бібліотеки, які забезпечуватимуть коректну роботу із UID, модулями зв’язку та іншими підключеними компонентами, а також підготувати самі плати до процесу прошивки. На цьому кроці проводиться проста перевірка підключення

плати до комп'ютера, обирається потрібний COM-порт, завантажуються необхідні драйвери й виконується базова адаптація програмного коду під конкретну модель приймача чи передавача.

Щоб уникнути несподіваних помилок під час заливання прошивки, забезпечити нормальну комунікацію між платою й комп'ютером, а також мати впевненість, що в подальшому і приймач, і передавач будуть працювати саме так, як цього вимагає ідея проекту. Цей розділ слугує містком між теорією і практикою, допомагає пройти всі початкові етапи без складнощів і зайвих хвилювань.

1.4.1 Підготовка середовища Arduino IDE.

Перш ніж розпочати роботу із самими платами, потрібно правильно налаштувати програмне середовище, у якому відбуватиметься програмиування і тестування пристроїв. На цьому етапі все починається із встановлення Arduino IDE (рисунок 1.17):

Рисунок 1.17 Інтерфейс програми Arduino IDE

Arduino IDE - найпопулярнішої й найбільш дружньої до користувача платформи для роботи з мікроконтролерами. Саме через неї ми надалі будемо створювати, компілювати й заливати прошивку як на плату приймача, так і на передавач. Сама процедура інсталяції досить стандартна: програму можна

завантажити з офіційного сайту arduino.cc, після чого встановити у декілька кліків як звичайний додаток.

Однак просто стандартної підтримки Arduino буває замало, якщо проект використовує плати на основі контролерів ESP8266. Щоб середовище побачило такі плати й могло робити з ними все потрібне, необхідно додатково додати їхню підтримку через Менеджер плат, без якого будь-які дії із цим типом обладнання будуть просто неможливі.

Після запуску Arduino IDE слід перейти у розділ Файл, далі Налаштування (Preferences). У вікні налаштувань знаходиться спеціальне поле Додаткові URL-адреси для менеджера плат (Additional Boards Manager URLs), куди потрібно вставити посилання на бібліотеку від розробників ESP8266, щоб середовище могло завантажити всі потрібні пакети для компіляції, програмування та обслуговування плат на базі ESP8266.

Без цього менеджер плат не буде знати про існування таких пристроїв, а користувач просто не зможе їх обрати в списку підтримуваних моделей. Також необхыдно стежити за оновленням бібліотек гарантує стабільність і сумісність коду з останніми версіями середовища.

Після додавання посилання потрібно відкрити Менеджер плат через меню Інструменти, далі Плита, та Менеджер плат... і знайти у списку серед ESP8266 версію by ESP8266 Community.

Після натискання кнопки Встановити кілька хвилин триває автозавантаження файлів, і вже після цього середовище Arduino IDE зможе працювати із нашою апаратурою: з’явиться можливість вибирати конкретну модель ESP8266 при налаштуванні проекту, а також компілювати та заливати на неї програмний код.

Важливий етап, адже без нього вся система програмування була б недоступною. Саме завдяки підтримці ESP8266, яка підключається через Arduino IDE, проект отримує гнучкість та можливість використовувати сучасні можливості розумних модулів для швидкої двосторонньої комунікації, автоматизації і передачі даних по мережі.

1.4.2 Підготовка плати передавача до прошивки

Перед початком прошивки важливо впевнитися, що ми працюємо саме з тією версією плати, яка відповідає вимогам нашого проекту. У світі пристроїв на базі ESP8266 існує багато різних ревізій і моделей, тому перед усіма подальшими діями потрібно обов'язково подивитися, яка мікросхема-конвертер використовується на нашій платі. У нашому випадку це CH341 - один із найбільш поширених контролерів для перетворення USB в UART. Оскільки на ринку можна зустріти й інші варіанти, наприклад CH340G чи CP2102 або iFT232, необхідно завжди перевіряти офіційну документацію до плати, щоб уникнути будь-яких плутанин із драйверами й сумісністю.

Для мікросхеми CH341 потрібно встановити відповідний драйвер на комп'ютер, оскільки без драйвера операційна система просто не побачить підключений пристрій, і зв'язок між Arduino IDE та платою буде неможливим. Драйвер можна завантажити з офіційного сайту виробника або взяти з перевірених джерел, після чого стандартно встановити відповідно до інструкцій. Перевірити правильну роботу драйвера можна через системну утіліту Диспетчер пристроїв - при під'єднанні плати повинен з'явитись розділ Порти (COM та LPT), в якому буде наш пристрій та у дужках буде відображатись номер COM порту, до якого приєднаний пристрій. Це важливо запам'ятати, оскільки від вибору COM-порту залежить можливість прошивання плати.

Далі, коли драйвер уже встановлений, відкриваємо Arduino IDE. Перш ніж безпосередньо приступати до написання або заливки прошивки, потрібно підключити всі необхідні бібліотеки для роботи з нашими модулями.

Оскільки у цьому проекті ми використовуємо бездротовий модуль NRF24, а це досить специфічний пристрій, до коду необхідно додати відповідну бібліотеку. Для цього у Arduino IDE обираємо пункт меню Скетч, далі Include Library, наступним Add .ZIP Library... (українською - Додати бібліотеку (ZIP)). Вказуємо шлях до архіву з бібліотекою, який попередньо треба завантажити та зберегти на комп'ютері, це дає змогу використовувати всі функції й методи, що потрібні для

коректного користування модулем, передачі та прийому даних бездротовим шляхом.

Переходимо у розділ Інструменти, далі Плита, та серед запропонованих варіантів обираємо ESP8266, далі Generic ESP8266 Module. Це забезпечує правильне формування прошивки під архітектуру саме нашої плати, адаптує всі необхідні налаштування компілятора й забезпечує сумісність із апаратною частиною проекту.

Після цього треба вибрати порт, до якого підключена плата. У меню Інструменти, далі Порт, обираємо відповідний COM-порт, що відображає підключену плату (його номер можна глянути у диспетчері пристроїв Windows). Якщо порт не відображається - отже драйвер встановлено некоректно або встановлено неактуальну версію драйвера, який не підтримую саме цю ревізію модуля, також необхідно переконатись, що кабель не має несправностей.

Встановлюємо правильну швидкість обміну даними (baud rate). У нашому випадку виставляємо рейт 115200 - це оптимальне значення для більшості плат ESP8266, яке дозволяє максимально швидко передавати дані й заливати скетч без помилок передачі.

Далі переходимо до більш тонких налаштувань:

- Upload Speed: 115200 - визначає швидкість завантаження прошивки, чим вища, тим швидше триває процес.
- Debug port: Disabled - відключає додатковий відлагоджувальний порт, якщо він не потрібен для моніторингу.
- Flash Size: 4MB (FS:2MB OTA:~1019KB) - налаштовує розподіл пам'яті мікроконтролера. Важливо виставити саме такі параметри, щоб залишалося достатньо місця й для самої програми, і для OTA-оновлень (оновлення по повітрю).
- C++ Exceptions: Disabled - вимикає виключення C++ для економії пам’яті.
- lwIP Variant: v2 Lower Memory - версія стеку TCP/IP із нижчим використанням оперативної пам’яті, що підвищує стабільність роботи на пристроях із малим обсягом RAM.

- Builtin Led: 2 - визначає, до якого піну підключений стандартний індикатор (корисно для індикації статусів у скетчі).
- Debug Level: None - немає виводу додаткової відлагоджувальної інформації, не навантажує систему.
- MMU: 32KB cache + 32KB IRAM (balanced) - оптимальне співвідношення кешу й оперативної пам’яті.
- Non-32-Bit Access: Use pgm_read macros for IRAM/PROGMEM - спеціальна оптимізація для роботи з пам’яттю.
- SSL Support: All SSL ciphers (most compatible) - забезпечує підтримку захищених з’єднань, якщо вони раптом будуть потрібні.
- Stack Protection: Disabled - параметр, який рекомендується залишати вимкненим на цьому етапі.

Ці налаштування дозволяють максимально стабільно й ефективно працювати з готовою платою-передавачем, уникати зайвих помилок у процесі прошивки і зводять до мінімуму проблеми сумісності.

Перед самою прошивкою потрібно ввести плату в режим завантаження. Для цього дії виконуються у чіткій послідовності:

1. Натискаємо і утримуємо кнопку Reset на платі.
2. Не відпускаючи Reset, натискаємо кнопку Boot і тримаємо обидві близько двох секунд.
3. Відпускаємо спочатку Reset, а потім - кнопку Boot. Завдяки цим діям плата переходить у special boot mode, в якому вона готова приймати нову прошивку.

Тепер можна перейти до заливки скетча. У Arduino IDE для цього досить натиснути кнопку у вигляді стрілки (праворуч вгорі). Слід дочекатися завершення процесу компіляції та запису.

Неможна від’єднувати плату від комп’ютера під час завантаження, а після успішного завершення перевірити, чи стартувала прошивка та чи плата функціонує, як очікувалось. Після натискання на кноку для прошивки плати, у терміналі Output в Arduino IDE з’явиться строка Connecting…..

При правильному підключенні плата виведе в термінал деякі свої параметри, та почнеться процесс прошивки, який буде супроводжуватись записами Writing at…… з процентом завантаження. Після завершення прошивки з'явиться відповідний надпис (рисунок 1.18). Якщо під час прошивки з'явились помилки, необхідно перевірити, чи встановлено вірний COM-порт, чи підключені необхідні бібліотеки, перевірити чи встановлено вірний тип плати, та спробувати замінити кабель або програматор.

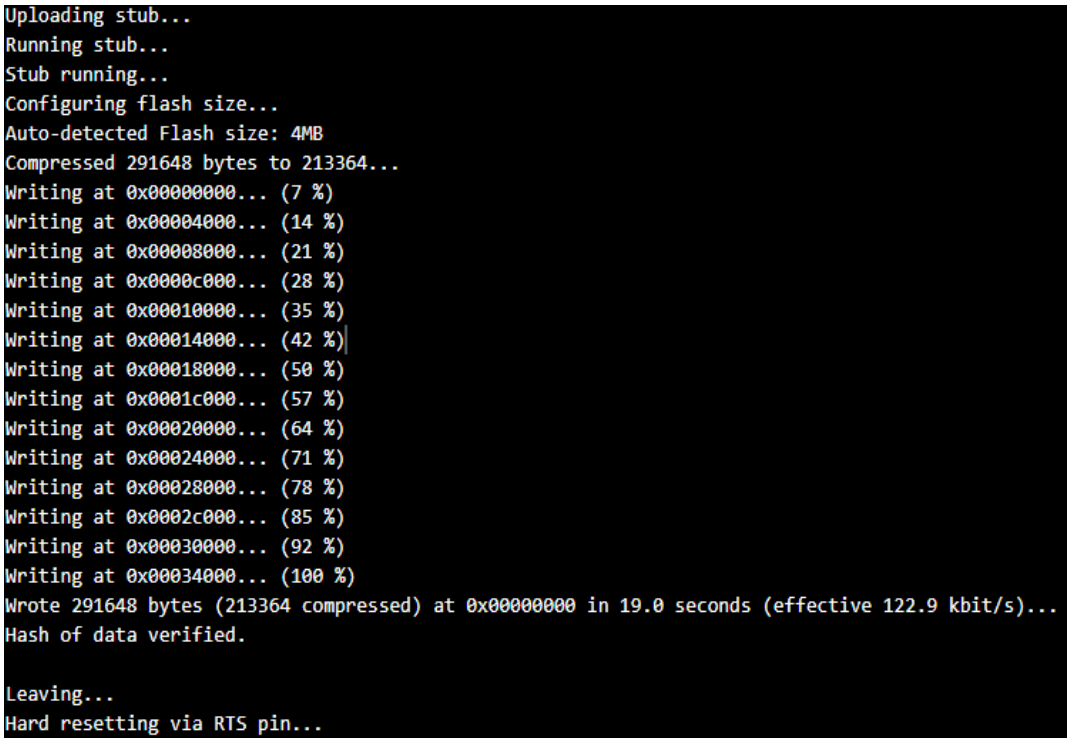


Рисунок 1.18 Процес вдалого завантаження прошивки на плату-передавач

1.4.3 Підготовка плати приймача до прошивки

Перед тим як розпочати прошивку приймача, потрібно бути впевненим, що ми маємо справу саме з тією апаратною версією плати, яка використовується у проекті. Дуже часто через декілька ревізій пристроїв однакові на вигляд плати можуть відрізнятися типом мікросхеми для конвертації USB-UART. В цьому випадку це IFT232 - популярний, надійний чіп, який забезпечує стабільний обмін даними між комп’ютером і платою на етапі прошивки.

Якщо тип мікросхеми не співпаде із зазначеним у документації, система не зможе розпізнати пристрій, або можуть виникати раптові обриви з’єднання під час

налаштування, тому обов’язково треба візуально або через документацію перевірити назву чіпа на платі.

Зробимо інсталяцію драйверів для нашої мікросхеми. Для iFT232 найбільш стабільним рішенням буде встановлення драйвера CDM2123620, який можна завантажити з офіційного сайту виробника або перевірених ресурсів. Без встановлення драйвера комп’ютер просто не побачить використовуєму плату, а процес прошивки буде неможливим. Інсталяція проходить у стандартному режимі, за інструкціями, і після перезавантаження система вже готова до роботи з пристроєм.

Коли під’єднання плати і драйвери вже налаштовані, переходимо в Arduino IDE для програмування. Оскільки для комунікації у нашому проекті використовується бездротовий модуль NRF24, обов’язково потрібно додати відповідну бібліотеку до проекту. Це робиться через меню Скетч, далі Include Library, наступним Add .ZIP Library.... Далі вказуємо шлях до завантаженої на комп’ютер бібліотеки у вигляді ZIP-архіву - вона навчить мікроконтролер взаємодіяти з NRF24, передавати та отримувати дані по радіо.

Після додавання бібліотеки переходимо в меню Інструменти, далі Плата, де серед варіантів обираємо ESP8266, а там вже NodeMCU 1.0 (ESP-12E Module). Вона відображає правильну апаратну конфігурацію для нашого приймача, що гарантує сумісність усіх бібліотек та коду з реальною схемою пристрою.

Після цього необхідно обрати COM-порт, до якого підключено плату (меню Інструменти, далі Порт). Якщо драйвер встановлений вірно, порт з’явиться у списку автоматично, але найкращим варіантом буде перевірити обраний COM-порт. Його вибір надзвичайно важливий: якщо порт обрано неправильно, жодна програма просто не зможе знайти плату для заливки прошивки. Швидкість передачі виставляємо на рівні 115200 бод. Це універсальне, рекомендоване значення для швидкого й стабільного програмування ESP8266, що дозволяє уникнути зависань та помилок у процесі завантаження коду.

Щоб завантажити скетч, приймач необхідно ввести у режим програмування флешинг.

Послідовність дій така ж, як для передавача:

1. Натискаємо й утримуємо кнопку Reset.
2. Не відпускаючи Reset, натискаємо Boot і тримаємо їх обидві приблизно 2 секунди.
3. Відпускаємо спочатку Reset, потім Boot.

Після цього переконуємося, що все під'єднано правильно, виходимо на головний екран Arduino IDE та натискаємо кнопку прошивки (Upload) - це кнопка у вигляді значок-стрілочка зверху. Під час заливки скетча не відключаємо плату від комп'ютера. Весь процес займає кілька хвилин.

Після завершення система повідомить про успішне завантаження. У разі помилки треба перевірити налаштування порту, тип плати і стан драйверів. Якщо все зроблено правильно - плату можна перезавантажити, і вона відразу почне виконувати нову прошивку.

1.5 Тестування та аналіз роботи системи

Тестування та аналіз роботи системи - це важлива частина усього процесу розробки. Саме цей етап дозволяє переконатися, що створене рішення не лише виглядає логічно на папері, а й безвідмовно виконує усі свої функції у реальних умовах експлуатації.

Після завершення розробки, сбірки, програмування та інтеграції веб-інтерфейсу виникає потреба детально перевірити, наскільки коректно система взаємодіє з підключеними пристроями, як пристрої взаємодіють між собою, як сервер обробляє UID, забезпечує стабільність збереження та авторизації даних, реагує на різні сценарії роботи.

В цьому розділі зосереджено увагу на кількох ключових аспектах: Оглянути ще раз плати на наявність дефектів, неякісної пайки, тощо. Тестування плат передавача й примача, тестування серверної частини та клієнтського інтерфейсу, які результати були отримані під час симуляції різних сценаріїв використання, а також що вдалося зробити щодо зручності, надійності й ефективності всієї системи в цілому.

1.5.1 Проведення тестування всіх модулів системи

Відразу після завершення прошивки нашого передатчика - модулю на основі ESP8266-12F з радіомодулем NRF24 настав час провести його тестування, щоб впевнитися у повній працездатності й відповідності очікуваному функціоналу.

Плата-передавач у цій системі відповідалає за регулярну й надійну передачу UID по радіоканалу без будь-яких зовнішніх взаємодій з подібними носіями. UID у цій архітектурі фіксований та «зашитий» у коді (наприклад, RS5656FT), а завдання передавача - автоматично надсилати цей ідентифікатор платі-приймачу з певною періодичністю.

Перед початком тестування ще раз перевіряємо правильність підключення модулів та цілісність живлення. Особливо звертаємо увагу на пінові з'єднання між ESP8266 і nRF24, це важливо для стабільної роботи радіоканалу. Коли все готово, переходимо до запуску пристрою.

Після підключення живлення запускаємо програму Arduino IDE та обираємо файл з прошивкою для плати-передавача, оскільки там вже є усі необхідні налаштування, тільки необхідно перевірити встановлений COM-порт. Відкриваємо серіальний монітор, він знаходиться в правому верхньому куту, виглядає як лупа з крапкою. Згідно коду, пристрій кожні 10 секунд передає фіксований UID через радіоканал і паралельно виводить відповідне повідомлення в Serial Monitor (Рисунок 1.19):

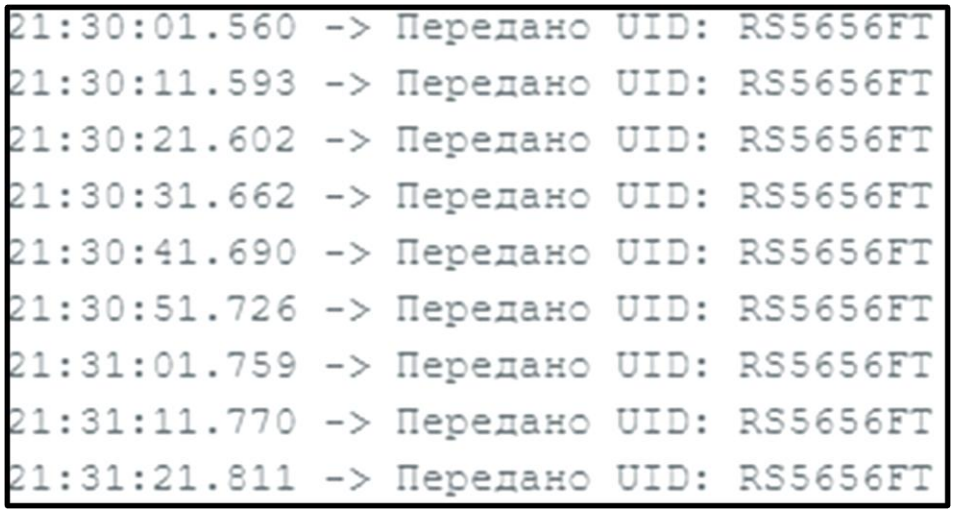


Рисунок 1.19 Передавач відправляє UID кожні 10 секунд по радіоканалу

Особливо зручно, що UID зашитий у коді та не змінюється випадковим чином. Це дає змогу виявити навіть незначні збої: якщо приймач отримує неіснуючий або пошкоджений UID - джерело проблеми буде одразу помітно.

Для повноцінного тестування бажано використати приймач із відповідною бібліотекою, та налаштованим каналом, який зможе підтвердити отримання кожного UID, але навіть перші тести на самому передатчику дозволяють переконатися, що цикл передачі й логіка роботи відповідає задумці.

Після успішного тестування передавача переходимо до наступного важливого етапу - перевірки і аналізу роботи приймача. Він виступає тією контактною точкою, яка з'єднує бездротовий модуль передачі UID із серверною частиною, даючи можливість усій системі функціонувати як єдиний цілісний комплекс.

Головна задача приймача - постійно слухати радіоканал у режимі очікування і миттєво реагувати, щойно на канал потрапляє новий UID. Пристрій не генерує запити самостійно й нічого не ініціює з нуля, а перебуває у пасивному режимі, за рахунок чого споживає мінімум ресурсів і знижує ймовірність будь-яких помилок через незаплановане дублювання або передавання інформації.

Як тільки приймач ловить пакет із UID (наприклад, наш RS5656FT), він одразу ж активізується: фіксує отриманий ідентифікатор та формує HTTP-запит для передачі цих даних на web-сервер. Сам факт відправлення інформації, а також відповідь від сервера, контролюються через програму Arduino IDE в Serial Monitor. Це дає змогу швидко локалізувати будь-яку проблему, якщо раптом система поводиться не так, як очікувалося.

В процесі тестування було приділено увагу аналізу можливих відповідей від веб-сервера, які дозволяють зрозуміти, чи вдалося приймачу успішно завершити завдання, або ж потрібно повторити спробу чи втрутитися вручну.

HTTP -1 - цей статус має особливе значення, адже він сигналізує про повну відсутність зв'язку із сервером. Причина може бути у вимкненому сервері, збої у мережевих налаштуваннях, неправильній адресі або ж просто втраті доступу до Інтернету. Саме така відповідь одразу виділяється на всіх індикаторах і часто

супроводжується окремим попередженням користувача через Serial Monitor в Arduino IDE, щоб він міг оперативно перевірити стан мережі чи серверної частини.

HTTP 400 - це статус помилкового запиту. Як правило, він означає, що сервер отримав дані у неправильному форматі, був некоректно сформований сам HTTP-запит. Причина може ховатися у помилках програмування, неправильних заголовках, або якщо є помилки з розпіновкою плати-приймача, відсутності обов’язкових параметрів або інших проблемах зі структурою переданих даних. У випадку отримання такої відповіді застосовується механізм повторної спроби.

HTTP 200 - це найбажаніший варіант розвитку подій, адже такий статус підтверджує, що запит прийнятий, оброблений і сервер успішно отримав UID. Жодних додаткових дій із боку користувача або приймача вже не потрібно, і система може перейти в режим очікування наступного UID.

Після кожної такої взаємодії приймач автоматично повертається в стан прослуховування - всі подальші UID прийматимуться й передаватимуться на сервер за аналогічною схемою. Для прикладу тестування приймач буде запущений, буде приймати по радіоканалу UID, але між повторним відправленням буде підключений web-сервер, задля того, щоб продемонструвати роботу з різними сценаріями, такими як UID зчитано, але немає зв'язку з web-сервером та UID зчитано, але зв'язок з web-сервером є. Це можна переглянути через Serial Monitor в Arduino IDE (рисунок 1.20):

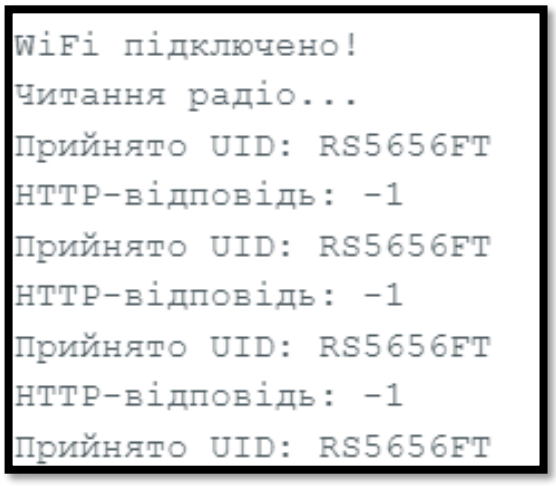


Рисунок 1.20 Демонстрація роботи приймача

Наступним кроком буде тестування web-серверу, який створений з використанням фреймворку Flask. Саме він керує потоком UID, що надходять із приймача, і відповідає за коректність та прозорість всіх операцій зі збереженням, перевіркою та видаленням даних. Його робота проста, але водночас надійна та наочно демонструється в логах серверу.

Після запуску сервер відразу стає на прослуховування усіх вхідних запитів та очікує, коли ж із плати-приймача прийде новий UID(Рисунок 1.21). Коли значення з'являється, воно одразу потрапляє у файл last_uid. Кожна важлива подія фіксується максимально інформативно: наприклад, якщо прийшов UID - у консолі видно, що одержано UID від пристрою та сам UID-код (наприклад RS5656FT).

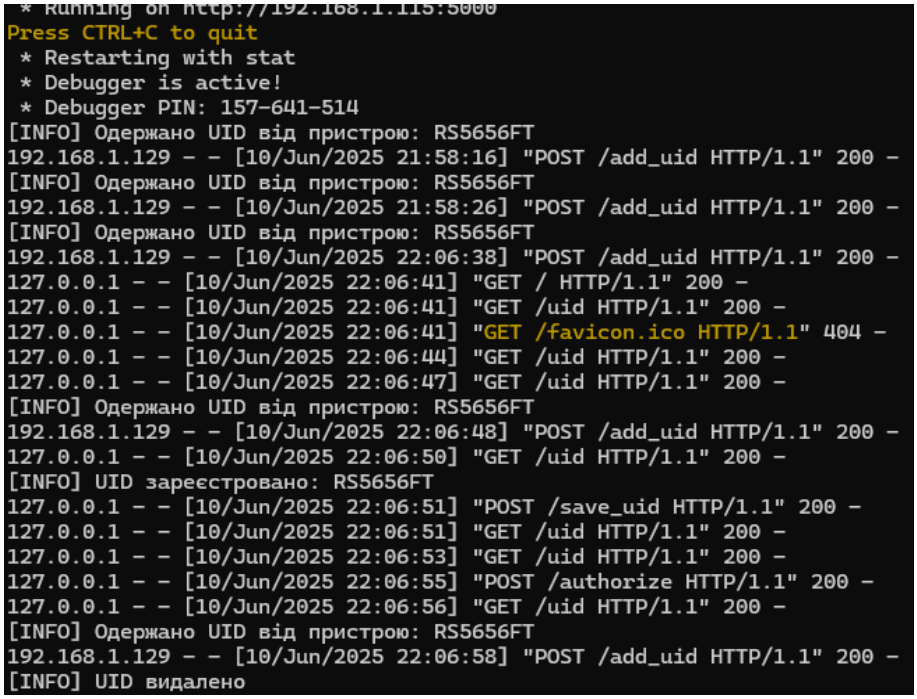


Рисунок 1.21 Логи web-серверу під час тестування з різними функціями

Після запуску сервер відразу стає на прослуховування усіх вхідних запитів UID. Коли значення з'являється, воно одразу потрапляє у файл last_uid. Кожна важлива подія фіксується максимально інформативно: наприклад, якщо прийшов UID - у консолі видно, що одержано UID від пристрою та сам UID-код (наприклад RS5656FT).

Під час тестування якраз і перевірялися різні сценарії: спочатку додавали новий UID через POST /add_uid - і сервер показував отриманий код, якщо UID треба зберегти як основний відправлявся POST-запит на /save_uid (рисунок 1.22)

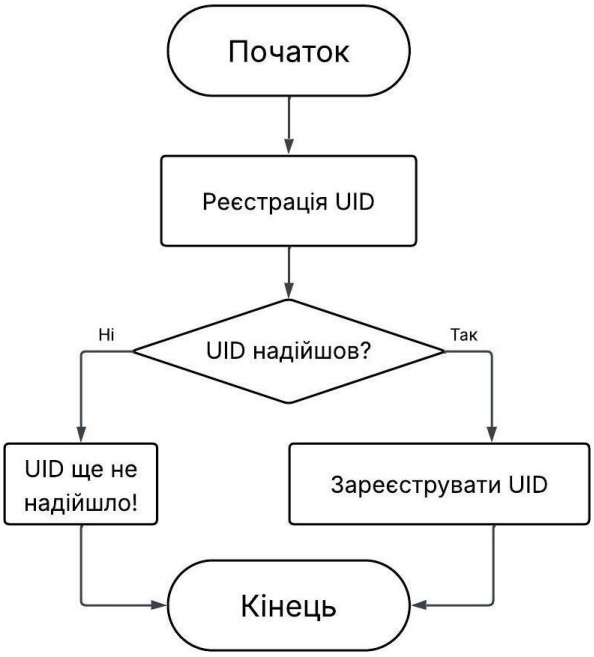


Рисунок 1.22 БСА системи реєстрації UID на сервері

У логах після цього з'являлось повідомлення, що UID зареєстровано. Для того щоб перевірити, чи проходить авторизація, надсилали запит на /authorize: тут сервер порівнює збережений UID із останнім отриманим та повертає відповідь про успішність чи невдачу цього співставлення (рисунок 1.23):

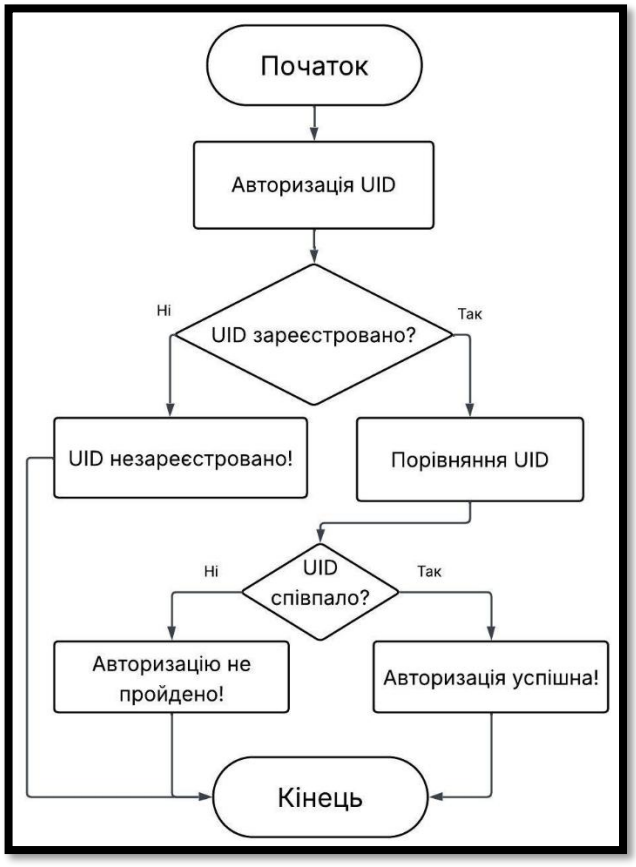


Рисунок 1.23 БСА функції авторизації web-серверу

Окрім базових операцій, важливий етап - видалення UID (рисунок 1.24). Після команди POST /delete_uid, система не просто очищає файл із UID, а й виразно повідомляє про це у логах (наприклад, [INFO] UID видалено) для уникання плутанини при подальшій реєстрації нових ідентифікаторів і засвідчує чесність даних.

Рисунок 1.24 БСА функції видалення зареєстрованого UID

Сервер не губиться і у випадках неправильних запитів - наприклад, неправильно вказаний маршрут типу GET /favicon.ico, для якого немає фізичного файлу, отримує чесну відповідь 404. Інші ж запити, приймаються та обробляються із відповіддю 200, що підтверджує: сервер стабільний, немає збоїв в обробці основних сценаріїв.

Веб-інтерфейс серверу - це зручна панель керування, що дозволяє в реальному часі взаємодіяти із сервером та UID, які надходять від пристрою-приймача. Після запуску головна сторінка одразу ж показує, який UID наразі зареєстрований у системі. Якщо UID ще не додано, то буде надпис «UID не додано», і це дає зрозуміти - система очікує на реєстрацію нового UID. Всі кнопки

розташовані компактно й інтуїтивно зрозумілі: «Додати UID», «Авторизація», «Видалити UID».

Найбільш показовий момент тестування - це, звісно, перевірка функції авторизації UID користувача. В стандартному режимі, коли передавач має той самий UID, що й зареєстрований, необхідно натиснути кнопку «Авторизація» - і з’являється зелене повідомлення: «Авторизація успішна!». Це означає, що все працює штатно, веб-сервер перевірив UID котрий йому надійшов, UID співпав, доступ дозволено.

Під час тестування саме інтерфейсу веб-сервера можна перевірити функцію порівняння посупаючого UID з зареєстрованим на сервері. Для цього в режимі реального часу було перепрошито передавач, де в коді прошивки було змінено UID новий, для прикладу, AS1234TT, тоді як зареєстрованим у системі лишався старий UID (RS5656FT). Саме тут проявилась зручність web-інтерфейсу. Після натискання кнопки «Авторизація» система одразу відреагувала на неспівпадіння UID та вивела червоним повідомленням: «Авторизація не пройшла!» (рисунок 1.25):

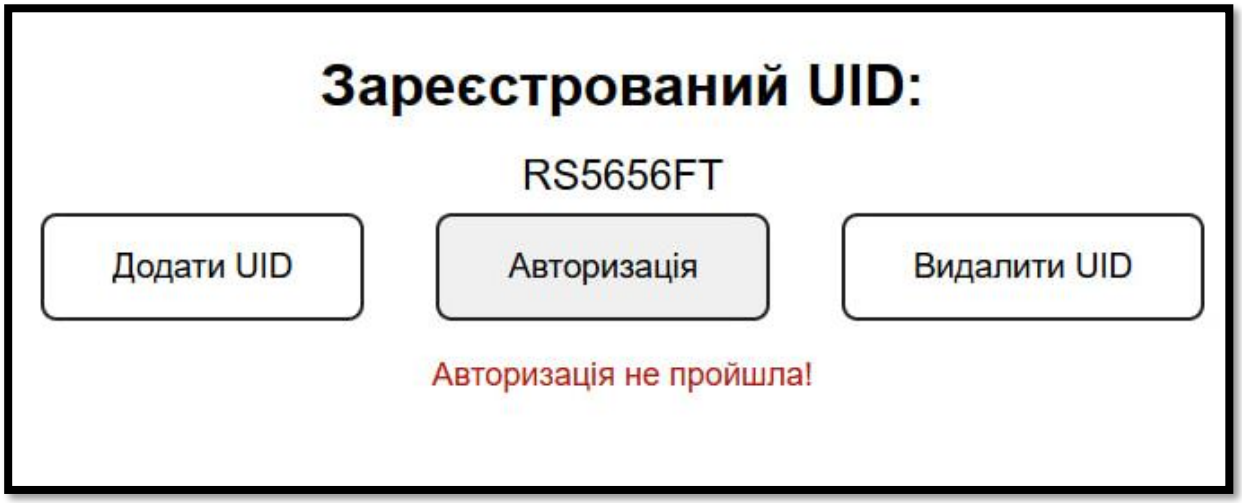


Рисунок 1.25 Демонстрація авторизації з незареєстрованим UID та відображення зареєстрованого UID

Це повідомлення говорить про те, що сервер ретельно порівнює поточний UID з тим, що вже внесений як довірений, і не дозволяє доступу «чужим» ідентифікаторам. Завдяки помітному кольору одразу ясно, що трапилася помилка, і можна не турбуватися про безпеку.

Система також одразу дає зворотний зв’язок при виконанні інших дій. Коли натискається «Додати UID», відбувається запис останнього отриманого UID у базу, і спливає синє інформативне повідомлення: «UID додано!». Якщо натиснути «Видалити UID», і система коректно очистила UID з пам’яті, ти відобразиться текстове підтвердження: «UID видалено!», також синім кольором. Тобто завжди розуміло, що відбувається і чи був запит оброблений.

Важливо, що для всіх повідомлень використовується короткий, разючий текст, а колір змінюється залежно від результату дії - зелений для успіху, синій для інформаційних попереджень, червоний для помилок. Наприклад, якщо UID ще не був надісланий з плати, при спробі «Додати UID» можна отримати повідомлення: «Ще не надійшов UID від пристрою!», що означає - спочатку треба ініціювати надсилання з передавача.

Для того, щоб ще наочніше побачити, як виглядає структура цієї системи авторизації, прикладена функціональна схема (рисунок 1.26). Вона чітко ілюструє послідовність передачі даних та ролі кожного модуля:

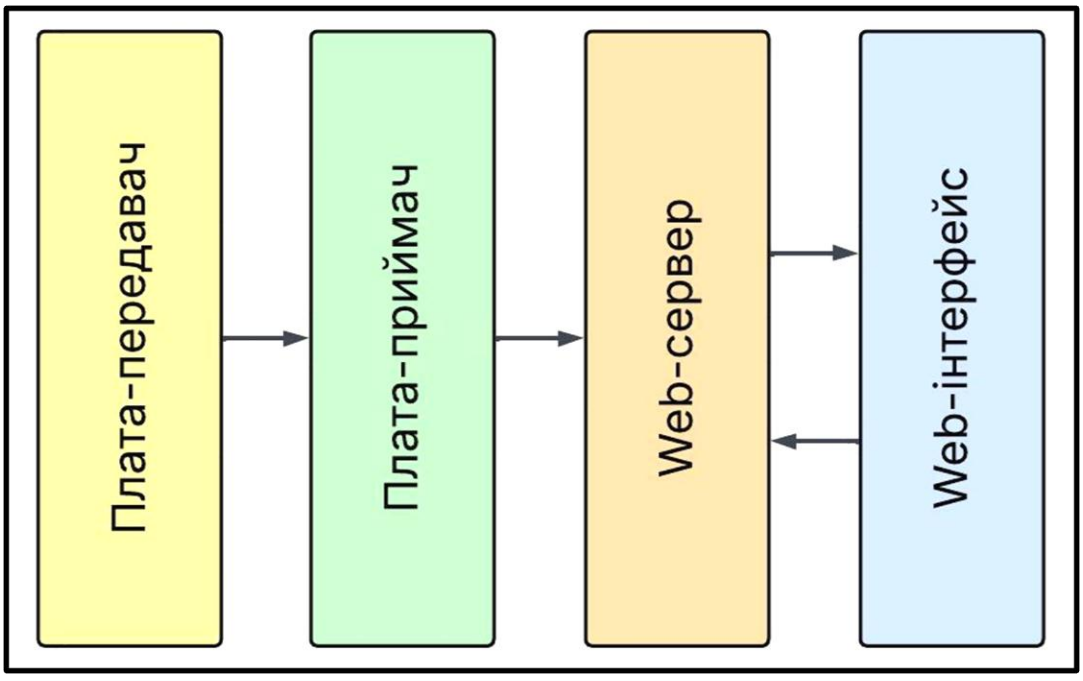


Рисунок 1.26 Функціональна схема роботи системи авторизації

На схемі видно, що сигнал спочатку формується на платі-передавачі, потім передається до плати-приймача, яка далі комунікує з web-сервером. Web-сервер, у свою чергу, забезпечує роботу із UID та дозволяє взаємодіяти з користувачем

через web-інтерфейс: той у реальному часі відображає стан системи та надає усі потрібні засоби керування.

1.5.2 Аналіз отриманих результатів

Після проведення комплексного тестування всієї системи авторизації, можна впевнено констатувати – розроблена система повністю відповідає поставленим завданням і працює саме так, як було задумано. Весь комплекс перевірок проводився у кілька етапів: починаючи з роботи базової зв'язки плата-передавач та плата-приймач, і закінчуючи інтеграцією web-сервера та web-інтерфейсу. В процесі тестування особливий акцент робився на тому, як різні компоненти взаємодіють між собою у реалістичних сценаріях.

Результати випробувань показали, що кожен модуль системи виконує свою функцію без збоїв. Завдяки продуманому механізму передачі й перевірки UID, система впевнено захищає від несанкціонованого доступу. При зміні UID на передавачі, web-сервер і web-інтерфейс моментально реагують і не допускають чужий UID до системи, про що користувач чітко і зрозуміло дізнається через повідомлення на екрані. Реалізація логіки миттєвого виводу повідомлень про статус операцій дозволила не лише підвищити зручність, але й зробити діагностику максимально наочною навіть для людей без технічної освіти.

Під час тестування використовувалися й різні ситуації: хибні UID, спроби видалення, та повторна авторизація. У кожному випадку система давала передбачуваний і контрольований результат. Наприклад, коли UID співпадав із зареєстрованим - авторизація проходила успішно, а коли був змінений - система чітко сигналізувала про це, не дозволяючи доступу.

Структура програмної частини залишалася гнучкою та зрозумілою при розширенні - веб-сервер на Flask виявився чудовою платформою для швидкого внесення змін та оперативного тестування нових сценаріїв. Також під час експериментів жодного разу не виникло ситуацій з непередбачуваними поведінками чи системними помилками. Це ще раз підтвердило правильність

обраної архітектури та ретельність відпрацювання усіх типових і навіть нетипових сценаріїв.

Веб-інтерфейс із простим, але сучасним дизайном, добре інтегрувався у загальний цикл керування та контролю. Він дав змогу бачити стан системи буквально в кілька кліків, одразу отримувати зворотний зв’язок. Отримані результати однозначно показують, що обрана модель взаємодії пристроїв та програмного забезпечення не просто працює, а й задовольняє всім вимогам: від стабільності до простоти використання для кінцевого користувача.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

Проєкт присвячено розробці системи авторизації користувача на web-сервері з використанням радіомодулів NRF, що забезпечує зручний і надійний контроль доступу. Система складається з двох частин: передавача (на електронному ключі), який формує та передає унікальний UID, і приймача на точці контролю, що приймає цей сигнал та перевіряє UID через web-сервер.

2.1 Розрахунок собівартості пристроїв

У цьому розділі проводиться оцінка вартості створеного пристрою. Розрахунок здійснюється укрупненим методом — визначається загальна вартість придбаних комплектуючих на основі специфікації. Детальний перелік компонентів і схема розрахунку наведені у таблиці 2.1.

Таблиця 2.1 Розрахунок відомості покупних комплектуючих елементів

Найменування, тип, модель	**Од.вим**	**Норма витрат на виріб**	**Ціна, грн.**	**Вартість комплектуючих**
Плата NodeMCU V3 ESP8266 (iFT232-s16)	шт.	1	168	168
Радіомодуль NRF24L01	шт.	2	58	116
Конденсатор 47uF 16V	шт.	2	1	2
модуль ESP8266 (ESP-12F)	шт.	1	78	78
Стабілізатор AMS1117 3V	шт.	1	12	12
Кнопка 6x6x9.5мм	шт.	2	1	2
Загальна вартість покупних комплектуючих елементів				378
Транспортні витрати (10%)				37.8
Всього ($В_{пк}$)				415.8

Калькуляцію планової собівартості розробленого виробу розраховуємо з використанням методу питомих ваг і структури собівартості аналогічної продукції.

Тому що, проектований виріб відноситься до радіоелектронної апаратури, то:

Питома вага матеріалу → $\alpha_м = 20\%$;

Питома вага покупних виробів →$\alpha_{пк} = 62\%$

Питома вага основної заробітної плати → $\alpha_{озп}$ =18%

Таблиця 2.2 Калькуляція планової собівартості

Найменування статті витрат	Значення статті, грн.	Розрахунок
1. Сировина і матеріал	134.13	Вм = $\alpha_м$ * $В_{пк}/\alpha_{пк}$ Вм= 20 * 415.8 / 62 = 134.13
2. Комплектуючі вироби і покупні напівфабрикати	415.8	Впк =см.табл.2.1
3. Основна заробітна плата	120.72	Воз= $\alpha_{озп}$ * $В_{пк}/\alpha_{пк}$ Воз= 18 * 415.8 / 62 = 120.72
4. Додаткова заробітна плата	48.29	Вдз = 0,4 * Воз Вдз = 0.4 * 120.72 = 48.29
5. Відрахування до єдиного соцфонду	37.18	Вєс = (Воз + Вдз) * 0.22 Вєс= (120.72 + 48.29) * 0.22 = 37.18
6. Загально-виробничі витрати	181.08	Взаг.вир= (1,2…1,5)* Воз Взаг.вир = 1.5 * 120.72 = 181.08
7. Виробнича собівартість	767.2	Свир = Вм+ Впк+ Воз+ Вдз+ Вєс +Взаг.вир Свир = 134.13 + 415.8 + 120.72 + 48.29 + 37.18 + 181.08 = 767.2
8. Адміністративні витрати	36.22	Ва = Воз * 0,3 Ва= 120.72 * 0.3 = 36.22
9. Витрати на збут	15.34	Взб = Свир * 0,02 Взб= 767.2 * 0.02 = 15.34
10.Інші операційні витрати	7.67	Воп = Свир *0,01 Воп= 767.2 * 0.01 = 7.67
Повна собівартість	826.43	Спов.= Свир+ Ва+ Взб+ Воп Спов= 767.2 + 36.22 + 15.34 + 7.67 = 826.43

Розмір планового прибутку, що включається в ціну, визначаємо по формулі:

$$П= (Спов* ρ)/100\% = (826.43 * 17) / 100 = 140.49 \text{ грн}$$

де

ρ-планова рентабельність продукції (10%...30%)

Оптову ціну виробу визначаємо по формулі:

$$Ц_о = Спов + П = 826.43 + 140.49 = 966.92 \text{ грн}$$

Ціну реалізації виробу встановлюємо з урахуванням ПДВ:

$$Цр = Цо + Пз = 966.92 + 193.38 = 1160.3 \text{ грн}$$

де Пз - податкове зобов'язання з ПДВ:

$$Пз = Цо * 0,2 = 966.92 * 0.2 = 193.38 \text{ грн}$$

Звідси:

$$Цр = 1160.3 \text{ грн}$$

2.2 Прогноз обсягів продаж даного виробу

Повна собівартість виготовлення одного виробу за поточний рік подана в таблиці 2.2. На основі цих даних розглядається прогноз обсягів продажу на другому етапі життєвого циклу виробу — «Виробництво», з розподілом по роках протягом чотирьох років. Це дозволяє оцінити потенціал виходу виробу на ринок і динаміку реалізації. Основні етапи й зони промислового випуску виробу детально показані на Рисунку 2.1:

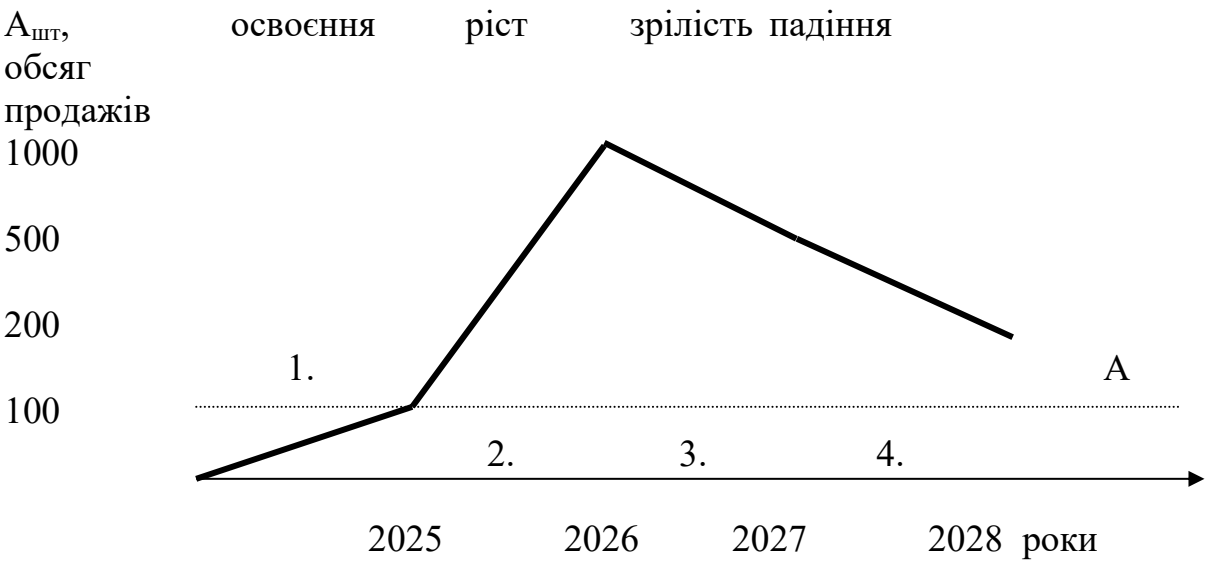


Рисунок 2.1 Характерні зони промислового випуску виробу

2.2 Розрахунок ціни продукту та розрахунок прибутку

У 2025 році планується продати 100 штук виробу під замовлення. У наступному році очікується зростання продажів, тому витрати на виробництво розраховуємо за формулою:

$$C_{пов\ i+1} = C_{пов i} \left(\frac{A_i}{A_{i=1}}\right)^{0.23},$$

де

A_i – обсяг продажів (виробництва) у 1 рік розрахункового періоду, шт.

A_{i+1} – обсяг продажів (I+1)-ом року, шт.;

0,23 – показник ступеня, який характеризує вплив росту обсягів виробництва на собівартість продукції.

Звідси випливає, що

$$Спов_{2026} = 826.43 * (100 / 1000)^{0.23} = 486.64 \text{ грн}$$

Якщо обсяги виробництва не зростають, тобто у разі, коли обсяг продажів у наступному році залишається незмінним або зменшується, витрати на виробництво встановлюються на тому ж рівні, що й у попередньому році.

$$Спов_{2027} = Спов_{2028} = 486.64 \text{ грн}$$

Плановий прибуток, який буде включено в оптову ціну підприємства на наступний рік при зростанні обсягу продажів, розраховується за формулою:

$$П_{i+1} = C_{пi+1} * \frac{\rho}{100} = 486.64 * (17/100) = 82.73 \text{ грн}$$

Звідси:

$П_{2026} = П_{2027} = П_{2028} = 82.73$ грн

Оптову ціну підприємства в наступні роки розрахункового періоду визначаємо по формулі:

$$Ц_{O_{i+1}} = C_{n_{i+1}} + П_{i+1} = 486.64 + 82.73 = 569.37 \text{ грн}$$

Звідси:

$Ц_{2026} = Ц_{2027} = Ц_{2028} = 569.37$ грн

Податкове зобов'язання визначається по формулі:

$$Пз_{i+1} = Ц_{O_{i+1}} * 0.2 = 569.37 * 0.2 = 113.87 \text{ грн}$$

Звідси:

$Пз_{2026} = Пз_{2027} = Пз_{2028} = 113.87$ грн

$Пз_{2026} = Пз_{2027} = Пз_{2028} = 113.87$ грн

Ціну реалізації одиниці продукції в наступні роки визначаємо по формулі:

$$Ц_{P_{i+1}} = Ц_{O_{i+1}} + Пз_{i+1} = 569.37 + 113.87 = 683.24 \text{ грн}$$

Звідси:

$Ц_{р2026} = Ц_{р2027} = Ц_{р2028} = 683.24$ грн

Вартісну оцінку результатів за розрахунковий період (Рт) визначаємо по формулі:

$$P_T = \sum_{i=t_p}^{t_k} A_i * Ц_{P_i} * \alpha_i$$

Де t_p, t_k – відповідно розрахунковий і кінцевий рік розрахункового періоду;

A_i – обсяг продажів у і-тім році, грн.;

α_i – коефіцієнт, що включає фактор часу, тобто коефіцієнт приведення різночасних витрат і результатів до розрахункового року.

Коефіцієнт α_I визначаємо по формулі:

$$\alpha_i = \left|1 + E_H\right|^{t_p - t_i}$$

Де $E_н$ – норматив ефективності капітальних вкладень, $E_н = 0,1$;

t_p – розрахунковий рік розрахункового періоду;

t_i – і-ій рік розрахункового періоду, витрати і результати якого приводяться до розрахункового року.

Вартісну оцінку за розрахунковий період визначаємо по формі, приведеної в таблиці 2.3

Таблиця 2.3 Розрахунок вартісної оцінки результатів

Найменування показника	Позначення	Розрахунок виробничого періоду				всього
		1-й	2-й	3-й	4-й	
Обсяг продажів, шт	A_i	100	1000	500	200	-
Ціна реалізації, грн.	$Ц_{рі}$	1160.3	683.24	683.24	683.24	-
Вартісна оцінка результатів, грн.	$A_i * Ц_{рі}$	116030	683240	341620	136648	-
Коефіцієнт, що враховує фактор часу	α_i	0,98	0,86	0,75	0,69	-
Вартісна оцінка результатів з урахуванням фактора часу, грн.	$A_i * Ц_{рі} * \alpha_i$	113709.4	587586.4	256215	94287.12	1051797.92

Виробництво дає змогу одержати дохід за 4 роки 1051.797 тис. грн.

3 РОЗДІЛ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

Охорона праці є невід'ємною складовою будь-якого інженерного чи технічного проекту, і особливого значення вона набуває при створенні систем контролю та моніторингу доступу. У випадку реалізації проекта «Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю», питання безпеки мають бути пріоритетними на всіх етапах роботи - від початкового проектування до практичного тестування. Важливо забезпечити умови, які мінімізують можливі ризики для здоров'я розробника, сприяють комфортній роботі з електронними компонентами та програмним забезпеченням, а також відповідають стандартам охорони праці. Дотримання цих вимог створює надійне, безпечне і продуктивне робоче середовище.

3.1 Аналіз шкідливих та небезпечних факторів, які впливають на розробника

У процесі розробки системи авторизації користувача, яка включає апаратну частину, такі як друковані плати та їх виготовлення, web-сервер та веб-інтерфейс, розробник стикається з низкою факторів, що можуть впливати на його безпеку, здоров'я та загальний комфорт. За класифікацією ГОСТ 12.0.003-74, такі фактори можна розподілити на кілька груп.

Фізичні фактори. Робота з паяльним обладнанням, живленням для електронних компонентів і випробуванням макетів передбачає ризик ураження електричним струмом, особливо при неправильному заземленні або недотриманні правил безпеки. Тривале сидіння за комп'ютером у незручній позі, недостатнє освітлення робочого місця та відсутність вентиляції також можуть негативно впливати на самопочуття.

Хімічні фактори. Робота з деякими матеріалами під час складання плат (флюсами, розчинниками або чистячими рідинами) може призвести до подразнення шкіри або дихальних шляхів при тривалому контакті або недостатній вентиляції.

Психофізіологічні фактори. Розробка таких систем вимагає високої концентрації уваги, аналітичного мислення та часто супроводжується розумовим перевантаженням, особливо під час відлагодження коду або пошуку помилок у схемі. Крім того, монотонна робота з великими обсягами інформації може викликати емоційне виснаження.

Біологічні фактори в умовах офісної або лабораторної роботи, як правило, не мають суттєвого впливу, однак за відсутності належного прибирання чи провітрювання приміщення можуть виникати умови для розвитку мікроорганізмів, що вражають слизові оболонки.

3.2 Вимоги з гігієни до приміщення

Раціональне облаштування робочого місця під час реалізації дипломного проєкту на тему «Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю» є важливою складовою загальної системи охорони праці. Організація робочого простору повинна забезпечувати не лише фізичну безпеку розробника, а й створювати умови для стабільної, продуктивної та безперебійної роботи протягом усього періоду виконання завдань.

Головними аспектами, які безпосередньо впливають на ефективність роботи, є мікроклімат приміщення, достатній рівень природного та штучного освітлення, якісна вентиляція, а також відповідність ергономічним нормам - зокрема, зручне розміщення обладнання, меблів та доступ до робочих інструментів

3.2.1 Вимоги до приміщення

Робоче приміщення, у якому здійснюється розробка системи авторизації користувача на web-сервері з використанням nrf-модуля, розташоване в окремому технічному кабінеті, площею не менше 6 м². Висота стелі становить 3 метри, що забезпечує необхідний мінімум повітряного об’єму - не менш як 18 м³ для однієї особи. Завдяки наявності вентиляції та додаткового припливу повітря, цей показник фактично збільшено до понад 20 м³, що відповідає нормативам ДСанПіН 3.3.6.042-99 щодо параметрів мікроклімату в робочому середовищі. Інтер’єр

приміщення витримано в нейтральних, комфортних зору тонах: стіни мають світло-сіре матове покриття, стеля білого кольору, а підлога вкрита світлим лінолеумом із антистатичними властивостями для запобігання накопиченню статичної електрики при роботі з електронними компонентами.

3.2.2 Вимоги до шуму в приміщенні

Робоче приміщення, у якому здійснюється розробка, спроектоване таким чином, щоб мінімізувати вплив шумового навантаження. Зокрема, встановлені сучасні вікна з двокамерними склопакетами, які значно знижують рівень зовнішнього шуму з вулиці, створюючи більш спокійну атмосферу для зосередженої роботи.

Обладнання, яке використовується в рамках проекту, зокрема плата-передавач та плата-приймач, а також комп'ютер з розгорнутим web-сервером, функціонують практично безшумно. Це дозволяє уникати зайвого звукового фону, який міг би відволікати або спричиняти втому протягом тривалого часу.

Упродовж дня рівень шуму в приміщенні коливається в межах 38–53 дБ, що цілком відповідає чинним санітарним нормам (ДСанПіН) і не створює дискомфорту для виконавця. У вечірній та нічний час рівень шуму знижується до 30–35 дБ, що забезпечує додатковий акустичний комфорт під час роботи у пізній час або при необхідності нічного моніторингу системи.

3.2.3 Вимоги до освітлення в приміщенні

Приміщення обладнане великим вікном площею близько 2 м², яке виходить на східну сторону. Завдяки такому розташуванню в перші години після сходу сонця забезпечується достатній рівень природного освітлення, що сприяє зниженню навантаження на зір та створює сприятливі умови для зосередженої роботи над проектом. Світлопропускна здатність віконного скла становить не менше 50%, що відповідає вимогам ДБН В.2.5-28:2018 і гарантує ефективне використання природного світла.

У разі недостатнього природного освітлення, особливо в похмурі дні або під час роботи у вечірній та нічний час, передбачено якісне штучне освітлення.

Робоче місце обладнане сучасним настільним LED-світильником потужністю 14 Вт із нейтральною температурою світла (приблизно 4000K), який забезпечує чітке освітлення без мерехтіння. Крім того, на стелі встановлений LED-світильник загального освітлення з інтенсивністю не менше 800 лм, що дозволяє підтримувати комфортний рівень освітлення на робочій поверхні в межах 400–600 лк.

Освітлення в приміщенні адаптується залежно від часу доби: вдень може використовуватися лише природне світло або в комбінації зі штучним для досягнення оптимального рівня освітлення. У вечірній час основним джерелом стає штучне світло, яке підтримує яскравість у межах 300–500 лк - достатньо для безпечної та зручної роботи з компонентами системи авторизації, включаючи мікроконтролери, дисплеї, зчитувачі та інші елементи.

3.2.4 Вимоги з електробезпеки

У робочому приміщенні передбачено сучасну електробезпеку: встановлено трьохконтактні розетки із заземленням, що мінімізує ризики у разі стрибків напруги чи несправностей обладнання. Електролінія окремо захищена автоматичним вимикачем, розміщеним у легкодоступному місці, тож у разі потреби живлення можна швидко вимкнути.

Для підключення електронних пристроїв - як базового комп’ютера, так і модулів системи авторизації - використовуються лише сертифіковані кабелі та адаптери живлення, що мають вбудований захист від короткого замикання. Всі кабелі прокладені з дотриманням технічних вимог: без натягу, перегинів, або перехрещень з джерелами тепла. Усі елементи електросистеми чітко промарковані, а біля розеток і щитка розміщено попереджувальні позначки про наявність електричної напруги.

3.2.5 Вимоги до мікроклімату

Для забезпечення комфортних умов праці у приміщенні створено стабільний та керований мікроклімат. Температурний режим підтримується в межах +20…+24°C, що відповідає рекомендованим значенням для тривалого

перебування людини. У холодну пору року тепло забезпечується центральною системою опалення, яка дозволяє прогрівати простір до заданого рівня. Влітку ж, під час підвищення температури зовнішнього повітря, застосовується побутовий кондиціонер, що допомагає підтримувати прохолоду в приміщені.

Вологість у приміщенні регулюється в межах близько 50%, що є оптимальним не лише для збереження самопочуття, а й для надійної роботи електронного обладнання. За необхідності додатково використовуються зволожувачі повітря, особливо в опалювальний сезон, коли повітря стає надто сухим.

Контроль температури, вологості та руху повітря здійснюється за допомогою цифрового термогігрометра, який знаходится у приміщенні. Усі параметри відповідають санітарним нормам, визначеним у ДСанПіН 3.3.6.042-99, для приміщень із використанням комп'ютерної техніки й електроніки.

3.3 Пожежна безпека

Робоче приміщення облаштоване з дотриманням усіх вимог пожежної безпеки, що діють на сьогодні. Для оздоблення використано матеріали з підвищеною стійкістю до займання, а покриття підлоги має антистатичні властивості, що дозволяє зменшити ймовірність іскріння або накопичення електростатичного заряду.

Усе електричне обладнання підключене через автоматичні вимикачі та якісні мережеві фільтри із захистом від перенапруги, що надійно оберігає систему від коротких замикань, перевантажень та перегріву.

У разі надзвичайної ситуації у приміщенні легко доступний вуглекислотний вогнегасник, який придатний для гасіння електроніки без шкоди для пристроїв. Також на видному місці розміщено інструкцію з дій у разі пожежі та план евакуації. Робочий простір організовано так, щоб завжди був вільний прохід до виходу - нічого не блокує шлях, навіть при повному завантаженні робочої зони.

ВИСНОВКИ

У цій дипломній роботі були проаналізовані, розроблені та протестовані компоненти сучасної системи авторизації на базі мікроконтролерів із радіомодулями NRF24 та web-сервером. Метою проекту стало створення зручної, універсальної та безпечної системи, яка забезпечує ідентифікацію користувача і гнучке керування доступом через передачу й перевірку унікального ідентифікатора (UID).

У ході виконання роботи проведений аналіз сучасних технологій і вибір оптимальної архітектури, що відповідає завданню проекту. Апаратна частина побудована на контролерах ESP8266 у поєднанні з NRF24, що дозволило реалізувати радіообмін UID між пристроями. Алгоритми взаємодії пристроїв забезпечують автоматичну роботу: передавач періодично надсилає UID, а приймач приймає дані і пересилає їх на сервер для подальшої обробки.

Програмна логіка авторизації реалізована на web-сервері, де організовано перевірку UID, а також зручний інтерфейс для керування списком ідентифікаторів. Інтерфейс розроблено максимально простим і зрозумілим для користувачів з різним рівнем підготовки у процесі експлуатації.

Позитивними сторонами розробки є актуальність використаних рішень, дальність зв’язку по радіоканалу, зручний інтерфейс для керування.

До недоліків можна віднести не захищеність від цілеспрямованих атак на канал зв’язку, оскільки на даному етапі не було впроваджено додаткових протоколів шифрування чи аутентифікації даних.

Для подальшого вдосконалення системи доцільно впровадити додаткові алгоритми шифрування UID при передачі, розширення можливостей веб-інтерфейсу та додавання ролей користувачів, а також інтеграцію батареї у передавач.

Розроблена система повністю відповідає технічному завданню. Результати тестування засвідчили стабільну роботу всіх компонентів. Система готова до впровадження в реальні умови експлуатації і може бути рекомендована для застосування у сучасних інтегрованих системах контролю доступу.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Теслюк В.М. Проектування пристроїв IoT з використанням мікроконтролерів AVR та ESP: навч. посіб. – Львів. Львівська політехніка, 2024 – 220 с.
2. ESP8266 Pinout Reference. Random Nerd Tutorials. – [Електронний ресурс] - https://randomnerdtutorials.com/esp8266-pinout-reference-gpios/. – Дата звернення: 02.03.2025.
3. NRF24L01 Module. Components101. – [Електронний ресурс] - https://components101.com/wireless/nrf24l01-pinout-features-datasheet. – Дата звернення: 03.03.2025.
4. Wi-Fi модуль ESP8266-12F. Arduino в Україні. – [Електронний ресурс] - https://arduino.ua/prod4217-wi-fi-modyl-esp8266-versiya-esp-12f?srsltid=AfmBOoohFvXZgF0ZbNCxhQSDspuRh_OscbXAXxuIdtvboWNVzYVO1Tcm. – Дата звернення: 12.03.2025.
5. Wi-Fi модуль NodeMCU V3 ESP8266 (iFT232-s16). Arduino в Україні. - [Електронний ресурс] - https://arduino.ua/prod1492-wi-fi-modyl-nodemcu-esp8266?srsltid=AfmBOoo8Iyii-WomGVOVyTAG-TCPsh7Gl9tGoz0yi-46bIUSBTF74U8L. – Дата звернення: 05.03.2025.
6. Ch341.driver: CH340 Drivers for Windows. GitHub. – [Електронний ресурс] - https://github.com/storef/ch341.driver. – Дата звернення: 19.03.2025.
7. GitHub - Witty ESP-12F WiFi module, ESP8266, Arduino, ino. GitHub. – [Електронний ресурс] - https://github.com/amkuipers/witty. – Дата звернення: 20.03.2025.
8. NRF24/RF24: OSI Layer 2 driver for nRF24L01 on Arduino. GitHub. - [Електронний ресурс] - https://github.com/nRF24/RF24. – Дата звернення: 17.03.2025.
9. Optimized High Speed Driver for nRF24L01 2.4GHz Wireless Transceiver. GitHub Pages. – [Електронний ресурс] - https://nrf24.github.io/RF24/. – Дата звернення: 22.03.2025.

ДОДАТОК А. Програмний код апаратної частини

```
// Код прошивки плати-передавача

// Плати: ESP8266-12F + nRF24, передавач UID через радіо
#include <SPI.h> // Підключення бібліотеки SPI для роботи з nRF24
#include <RF24.h>  // Підключення бібліотеки RF24 для радіозв'язку

//Налаштування RF24
#define CE_PIN 4    // GPIO15 (D8)
#define CSN_PIN 5    // GPIO2  (D4)

RF24 radio(CE_PIN, CSN_PIN);
const byte address[6] = 00001; // Адреса приймача для радіозв'язку
const char uid[] = RS5656FT;  // унікальний UID

void setup() {
  Serial.begin(115200);  // Ініціалізація серійного порта для Monitor
  delay(100);
  radio.begin(); / Запуск модуля nRF24
  radio.openWritingPipe(address); // Відкриття каналу для передачі даних адресу
  radio.setPALevel(RF24_PA_LOW); // Встановлення низького рівня потужності
  radio.stopListening(); // Перехід у режим передачі
}

void loop() {
  bool ok = radio.write(&uid, sizeof(uid));  // Передача UID через радіомодуль
  Serial.print(Передано UID: ); Serial.println(uid); // Вивід UID у серійний монітор для контролю
  delay(10000); // передавати кожні 10 сек
}

//Код прошивки плати-приймача

#include <ESP8266WiFi.h> // Бібліотека для підключення до WiFi
#include <ESP8266HTTPClient.h> // Бібліотека для роботи з HTTP-запитами
#include <SPI.h> // Для SPI-зв’язку з nRF24
```

```
#include <RF24.h>  // Бібліотека для роботи з радіомодулем nRF24

const char* ssid = SSID;         // Назва WiFi-мережі
const char* password = password;   // Пароль WiFi

// --- Налаштування RF24 ---
#define CE_PIN 4 // CE-пін радіомодуля (GPIO4)
#define CSN_PIN 5 // CSN-пін радіомодуля (GPIO5)

RF24 radio(CE_PIN, CSN_PIN); // Ініціалізація об'єкта для роботи з nRF24
const byte address[6] = 00001; // Адреса, яку слухає приймач

// --- Сервер Flask ---
const char* server = http://192.168.1.110:5000/add_uid; //IP  ПК із сервером

WiFiClient wifiClient; // Об'єкт для WiFi-з'єднання

void setup() {
  Serial.begin(115200); // Запуск серійного монітору
  delay(100);
  // Підключаємося до WiFi
  WiFi.begin(ssid, password); // Починаємо підключення до WiFi
  while (WiFi.status() != WL_CONNECTED) { // Чекаємо, поки не під’єднається
    delay(300);
    Serial.print(.);
  }
  Serial.println(\nWiFi підключено!);
  // Налаштування радіо-модуля NRF24
  radio.begin();   // Ініціалізація радіомодуля
  radio.openReadingPipe(0, address); // Відкриваємо канал прослуховування на адресу
  radio.setPALevel(RF24_PA_LOW); // Встановлюємо низьку потужність передачі
  radio.startListening();  // Переходимо в режим прийому
  Serial.println(Читання радіо...); // Підтвердження початку читання
}
void loop()
```

```
if (radio.available()) {  // Якщо прийшли дані по радіо
  char uid[32] = {0};   // Масив для отриманого UID
  radio.read(&uid, sizeof(uid)); // Читання UID з радіомодуля
  Serial.print(Прийнято UID: );
  Serial.println(uid);
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http; // Створюємо HTTP-клієнт
    http.begin(wifiClient, server); // Вказуємо адресу сервера
    http.addHeader(Content-Type, application/json); // Заголовок запиту
    String payload = {\uid\:\ + String(uid) + \}; // Тіло запиту з UID
    int response = http.POST(payload); // Відправляємо POST-запит до Flask
    Serial.print(HTTP-відповідь: );
    Serial.println(response);   // Вивід коду відповіді
    http.end();  // Завершення HTTP-з'єднання
  } else {
    Serial.println(WiFi не підключено!); // Якщо немає підключення до мережі
  }
  delay(3000); // захист від спаму (3 сек)
 }
```

ДОДАТОК Б. Програмний код серверної частини

```
// Програмний код web-серверу
from flask import Flask, request, jsonify, send_file
import os

app = Flask(__name__)

UID_FILE = 'key'
LAST_UID_FILE = 'last_uid'

# Ендпоінт, куди приймач скидає UID
@app.route('/add_uid', methods=['POST'])
def add_uid():
    data = request.get_json(force=True)
    uid = data.get('uid', '').strip()
    if not uid:
        return jsonify({'message': 'Не надано UID!'}), 400
    # Записати останній отриманий UID
    with open(LAST_UID_FILE, 'w', encoding='utf-8') as f:
        f.write(uid)
    print(f[INFO] Одержано UID від пристрою: {uid})
    return jsonify({'message': 'UID оновлено!'})

# Повернути зареєстрований UID
@app.route('/uid')
def get_uid():
    uid = ''
    if os.path.exists(UID_FILE):
        with open(UID_FILE, 'r', encoding='utf-8') as f:
            uid = f.read().strip()
    return jsonify({'uid': uid if uid else ''})

# Повернути останній отриманий UID від приймача (не збережений!)
@app.route('/last_uid')
def get_last_uid()
```

```
    uid = ''
    if os.path.exists(LAST_UID_FILE):
        with open(LAST_UID_FILE, 'r', encoding='utf-8') as f:
            uid = f.read().strip()
    return jsonify({'uid': uid if uid else ''})

# Зберегти останній UID як зареєстрований (по кнопці Додати UID)
@app.route('/save_uid', methods=['POST'])
def save_uid():
    # Читаємо останній UID, зберігаємо як зареєстрований
    if not os.path.exists(LAST_UID_FILE):
        return jsonify({'message': 'Ще не надійшов UID від пристрою!'}), 400
    with open(LAST_UID_FILE, 'r', encoding='utf-8') as fr:
        last_uid = fr.read().strip()
    if not last_uid:
        return jsonify({'message': 'Ще не надійшов UID від пристрою!'}), 400
    with open(UID_FILE, 'w', encoding='utf-8') as fw:
        fw.write(last_uid)
    print(f[INFO] UID зареєстровано: {last_uid})
    return jsonify({'message': 'UID додано!'})

# Верифікація (авторизація) — порівнюємо реальний останній UID і збережений
@app.route('/authorize', methods=['POST'])
def authorize():
    if not os.path.exists(LAST_UID_FILE) or not os.path.exists(UID_FILE):
        return jsonify({'message': 'UID не збережено або не надійшов!'}), 200
    with open(LAST_UID_FILE, 'r', encoding='utf-8') as f1:
        last_uid = f1.read().strip()
    with open(UID_FILE, 'r', encoding='utf-8') as f2:
        reg_uid = f2.read().strip()
    if not last_uid or not reg_uid:
        return jsonify({'message': 'UID не збережено або не надійшов!'}), 200
    if last_uid == reg_uid:
        return jsonify({'message': 'Авторизація успішна!'})
    else:
```

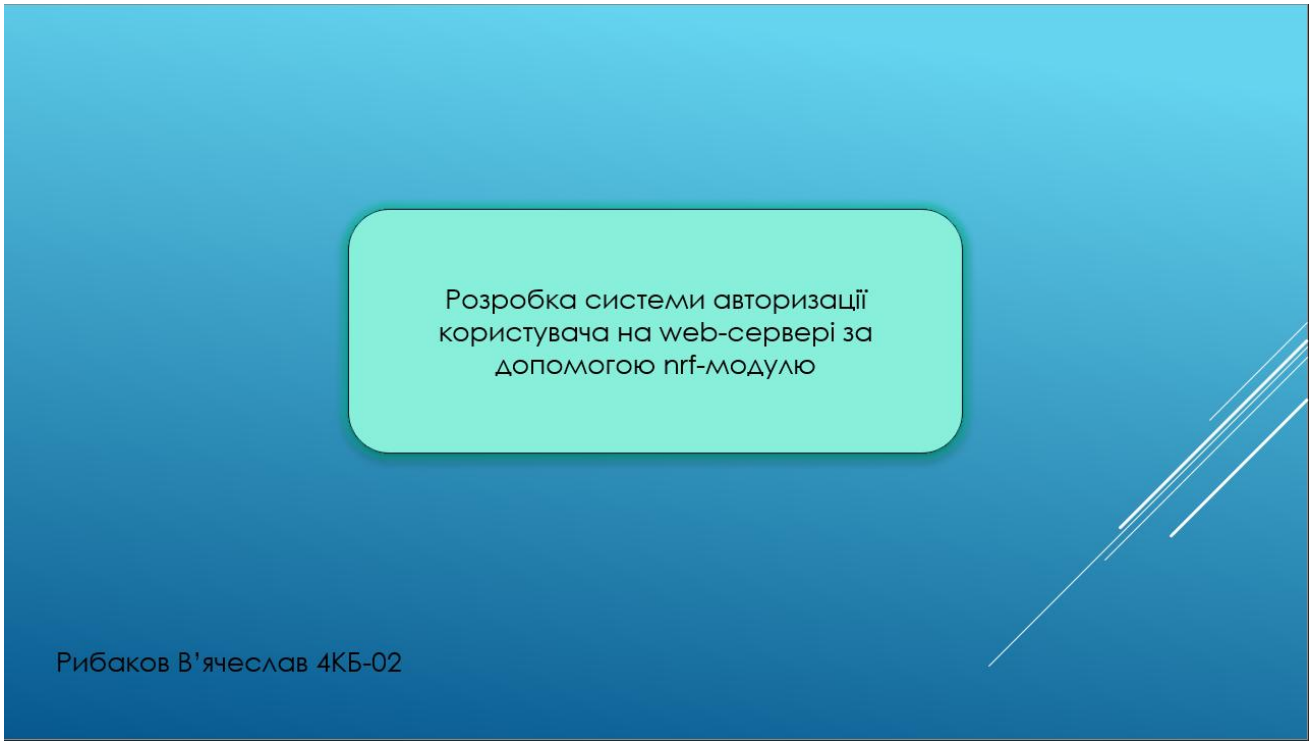
```
return jsonify({'message': 'Авторизація не пройшла!'})

# Видалити зареєстрований UID
@app.route('/delete_uid', methods=['POST'])
def delete_uid():
    open(UID_FILE, 'w').close()
    print('[INFO] UID видалено')
    return jsonify({'message': 'UID видалено!'})

# Головна сторінка
@app.route('/')
def index():
    return send_file('page.html')

if __name__ == '__main__':
    app.run(host=0.0.0.0, port=5000, debug=True)



//Програмий код для web-серверу
<!DOCTYPE html>
<html lang=uk>
<head>
    <meta charset=UTF-8>
    <title>Контроль UID</title>
    <style>
        body { font-family: Arial, sans-serif; margin: 45px;}
        h1 {text-align: center; margin-bottom: 18px;}
        #uid-current {text-align:center; font-size: 1.45em; margin-bottom:7px;}
        .btn-row {
            display: flex;
            justify-content: center;
            gap: 40px;
```

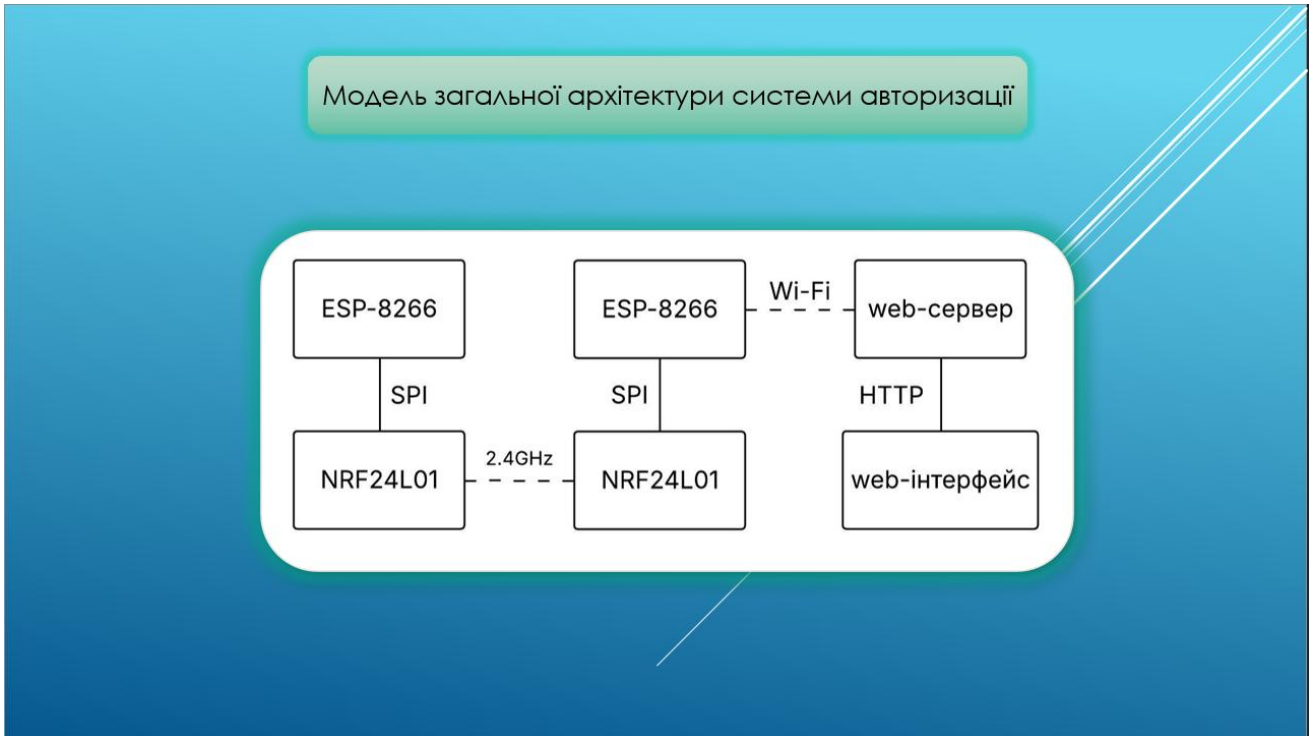
```
margin-bottom: 18px;
      }
      button {
        font-size: 1.17em;
        padding: 17px 38px;
        border-radius: 9px;
        border: 2px solid #222;
        background: #fff;
        cursor:pointer;
        transition: background 0.2s;
        min-width: 160px;
      }
      button:hover { background: #f0f0f0; }
      #message {
        min-height: 2.3em;
        text-align: center;
        font-size: 1.13em;
        margin-top: 20px;
        color: #166332;
      }
    </style>
</head>
<body>
    <h1>Зареєстрований UID:</h1>
    <div id=uid-current>Завантаження...</div>

    <div class=btn-row>
      <button onclick=saveUID()>Додати UID</button>
      <button onclick=authorizeUID()>Авторизація</button>
      <button onclick=deleteUID()>Видалити UID</button>
    </div>

    <div id=message></div>

<script>
```

```
function showMessage(msg, color=#166332) {
    let m = document.getElementById('message');
    m.textContent = msg;
    m.style.color = color;
    if(msg)
        setTimeout(() => { m.textContent=''; }, 3000);
}

function updateUIDs() {
    fetch('/uid').then(r => r.json()).then(data => {
        let curr = document.getElementById('uid-current');
        curr.textContent = data.uid ? data.uid : UID не додано;
    });
}

function saveUID() {
    fetch('/save_uid', {method:'POST'})
    .then(r=>r.json())
    .then(resp => {
        updateUIDs();
        showMessage(resp.message, #1151b2);
    });
}

function authorizeUID() {
    fetch('/authorize', {method:'POST'})
    .then(r=>r.json())
    .then(resp => {
        let color = /успішна/i.test(resp.message) ? #2e993b : #bc1d18;
        showMessage(resp.message, color);
    });
}

function deleteUID() {
    fetch('/delete_uid', {method:'POST'})
```

```
    .then(r=>r.json())
    .then(resp => {
        updateUIDs();
        showMessage(resp.message, #1151b2);
    });
}

updateUIDs();
setInterval(updateUIDs, 3000);
</script>
</body>
</html>
```

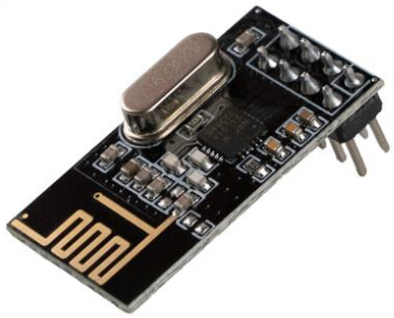
ДОДАТОК Б. Слайди мультимедійної презентації





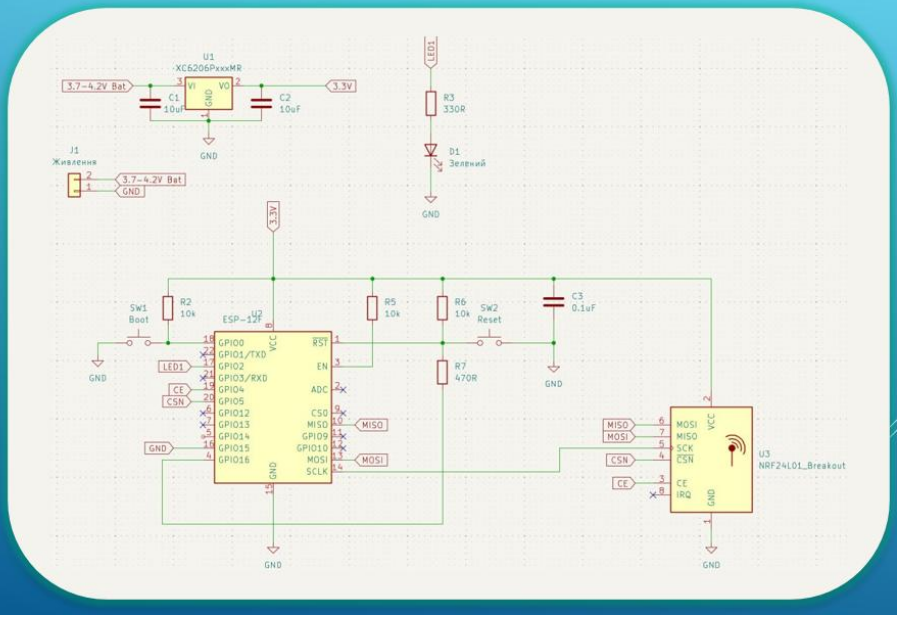
Модулі системи передавача

NRF24L01



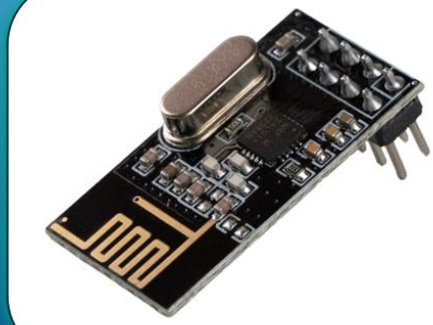
ESP-8266 12F

Принипово-електрична схема передавача



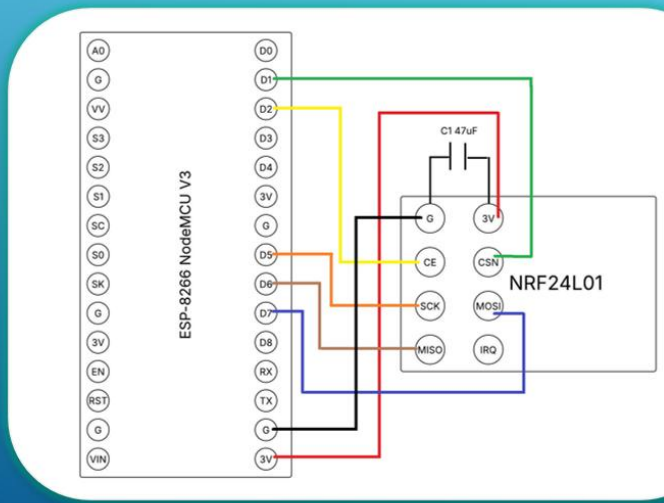
Модулі системи приймача

NRF24L01



ESP-8266-12E NodeMCU V3

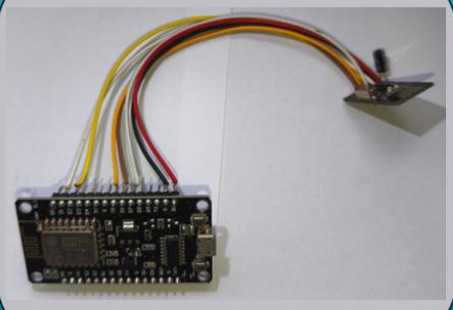
Принципова схема приймача



Демонстрація виготовлених пристроїв

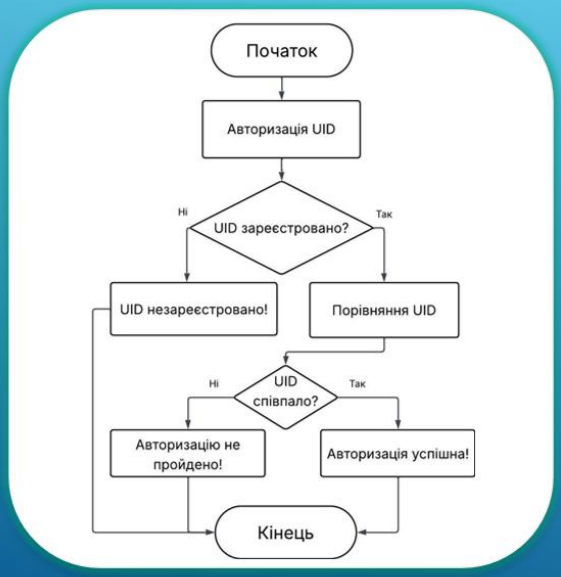
Передавач

Приймач

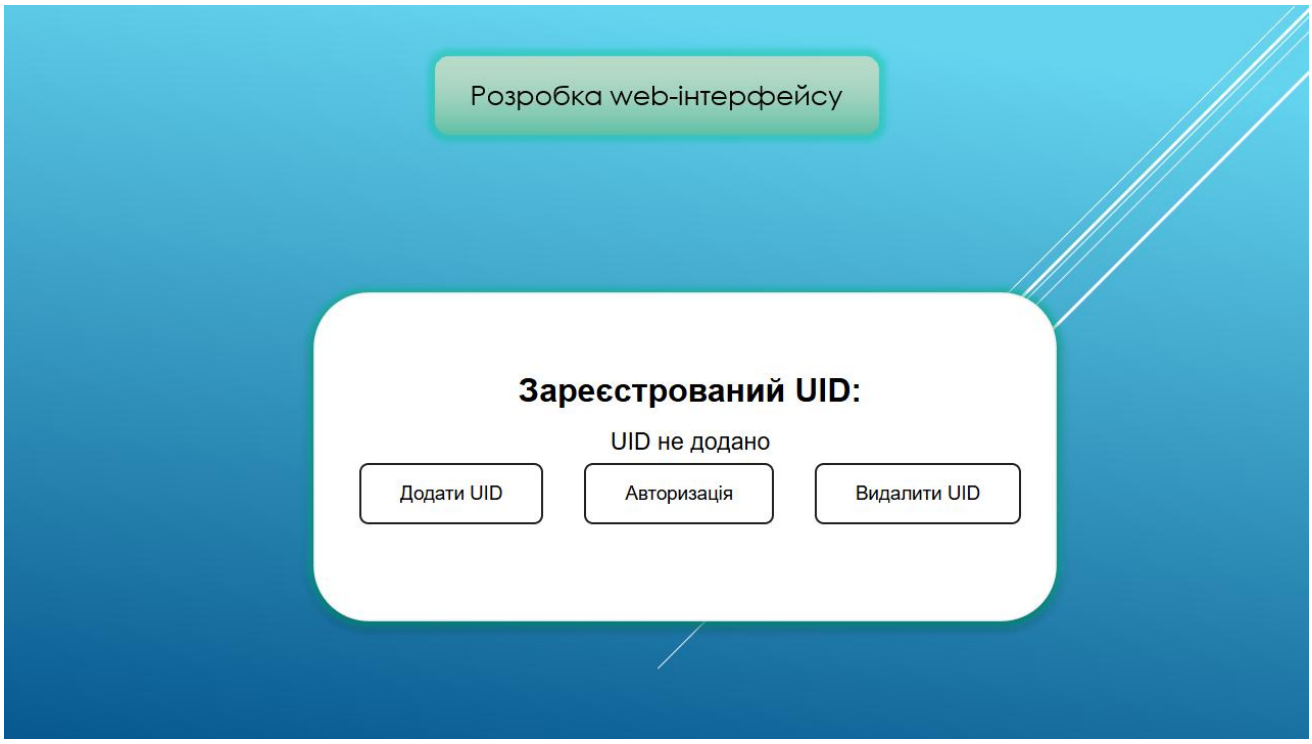


Алгоритм реєстрації UID

Алгоритм авторизації UID



Алгоритм видалення UID


Розробка web-інтерфейсу
Зареєстрований UID:
UID не додано
Додати UID
Авторизація
Видалити UID

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект (роботу) здобувача (здобувачки) освіти

відділення комп'ютерних систем

Рибаков В'ячеслав Олегович

(прізвище, ім`я та по батькові)

Спеціальність 123 " Комп'ютерна інженерія "

ОПП «Безпека комп'ютерних систем і мереж»

Керівник дипломного проекту (роботи) Залапін Олексій Ігорович

(прізвище, ім`я та по батькові)

Тема дипломного проекту (роботи) *Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю*

Обсяг розрахунково-пояснювальної записки 78 сторінок

Обсяг графічної (презентаційної) частини 11 аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту (роботи) завданню

Представлений на рецензію дипломний проект повністю відповідає меті проектування та технічному завданню. Тематика дипломного проекту є актуальною для своєї галузі та присвячена питанням кібербезпеки та її реалізації за допомогою апаратної та програмної частин.

б) характеристика виконання кожного розділу дипломного проекту (роботи)

Дипломний проект складається зі вступу, трьох розділів, висновків, переліку використаних джерел. У основному розділі розглянуті питання розробки апаратної та технічної частини, сформовано механізм та web-сервер згідно до теми дипломного проекту та завданню, виконано проектування основних аспектів плати з використанням nrf-модулю. За допомогою відповідного програмного забезпечення реалізовані всі намічені роботи з процесом розробки.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту (роботи) Графічна частина виконана на достатньо високому рівні у вигляді презентації із використанням офісного пакету Microsoft PowerPoint та Visio. Пояснювальна записка виконана акуратно та у відповідності до норм оформлення документів із використанням офісного пакету Microsoft Word. Загальна якість виконання документації – добра, академічного плагіату у роботі не виявлено

г) перелік позитивних якостей дипломного проекту (роботи)

1. Детально описано процес виконання розробки апаратної та програмної частини;
2. Виконано проектування елементів плати та розробки програмного забезпечення з поясненнями на схемах та за допомогою коду;
3. Розробка апаратної частини велася здебільшого самостійно

д) основні недоліки дипломного проекту (роботи)

1. Web-сервер має просту візуальну складову;
2. Взаємодія між апаратною та програмними частинами відбувається лише при зазначеному в коді бездротовому з'єднанні;
3. Присутні деякі помилки оформлення пояснювальної записки

Оцінка розрахункової частини Відмінно

Оцінка графічної частини Добре

Загальна оцінка Відмінно

Прізвище, ім'я, по батькові рецензента *к.т.н. Шибаєва Наталя Олегівна*

Місце роботи і посада рецензента *Національний університет «Одеська політехніка», доцент кафедри інформаційних технологій*

Підпис

«20» *червня* 2025 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Рибакова В'ячеслава Олеговича

(прізвище, ім`я та по батькові)

Спеціальність: *123 "Комп'ютерна інженерія"*

Освітньо-професійна програма: *«Безпека комп'ютерних систем і мереж»*

Тема дипломного проекту: *Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка містить 78 сторінки. У пояснювальній записці наведено етапи розробки системи авторизації користувача на web-сервері за допомогою nrf-модулю, а також його програмне забезпечення. Графічна частина складається з 11 слайдів мультимедійної презентації, які також містять зображення та схеми, передбачені технічним завданням. Якість виконання пояснювальної записки та графічної частини добра, розробку виконано в повному обсязі.*

б) самостійність роботи над проектом: *Протягом всього строку дипломного проектування та переддипломної практики здобувач освіти Рибаков В.О. поступово та послідовно виконував всі етапи розробки. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Рибаков В.О. під час роботи над дипломним проектом вивчив достатню кількість літературних джерел та матеріалів за даною тематикою. Вважаю, що теоретична підготовка дипломника добра і він готовий до захисту дипломного проекту*

г) вміння розв'язувати виробничі та конструкторські питання *Під час дипломного проектування здобувач освіти Рибаков В.О. мав змогу самостійно приймати окремі рішення з реалізації апаратної та програмної частини системи авторизації за допомогою nrf-модулю та показав вміння організовано працювати над поставленим завданням, розробляти структурні схеми та програмно-апаратні зв'язки із застосуванням сучасних комп'ютерних програмних засобів, таких як Arduino IDE, Python, а також готувати презентаційні та звітні матеріали.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Залапін Олексій Ігорович*

Місце роботи і посада керівника дипломного проекту *ВСП "Одеський технічний фаховий коледж ОНТУ", викладач спецдисциплін комісії комп'ютерних технологій та програмної інженерії.*

Підпис _____

«16» червня 2025 р.

ДОВІДКА

циклової комісії КТ та ПІ
про допуск до захисту дипломного проєкту
здобувача (здобувачки) освіти IV курсу
відділення комп'ютерних систем групи 4КБ-02

Рибакова В'ячеслава Олеговича

на тему *Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю*

Висновок відповідальної особи за проведення нормоконтролю:
пояснювальна записка до дипломного проєкту виконана з некритичними порушеннями ДСТУ та оформлена відповідно до вимог Положення про дипломне проєктування

(підпис) *16.06.2025* (дата) *Петрашова В.І.* (П.І.Б.)

Висновок відповідальної особи за перевірку роботи на наявність академічного плагіату *згідно звіту про перевірку від 15.06.2025 р. значення коефіцієнту подібності в роботі становить 6,39%, коефіцієнт цитування – 0,66%.*

(підпис) *16.06.2025* (дата) *Краснокутська К.Г.* (П.І.Б.)

Попередня експертиза (малий захист) дипломного проєкту

здобувача (здобувачки) освіти *Рибакова В.О.* (П.І.Б.)

проведена « *16* » *червня* 20*25* р.

Висновки *Пояснювальна записка до дипломного проєкту виконана у повному обсязі. Випускна кваліфікаційна робота (дипломний проєкт) відповідає вимогам Положення про дипломне проєктування та рекомендована до захисту.*

Голова ЦК КТ та ПІ (підпис) *Кривченко Ю.В.* (П.І.Б.)

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Рибаков В'ячеслав Олегович
здобувач освіти гр. 4КБ-02, та

Залапін Олексій Ігорович,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю» (автор роботи – Рибаков В.О., керівник роботи – Залапін О.І.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2025 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Рибаков В.О. /

Керівник _____ / Залапін О.І. /

«16» червня 2025 р.

StrikePlagiarism.com

Дата звіту 6/15/2025
Дата редагування 6/15/2025

Документ прийнятий

Звіт подібності

метадані

Назва організації
Odesa Technical Professional College of Odesa National University of Technology
Заголовок
Розробка системи авторизації користувача на web-сервері за допомогою nrf-модулю

Автор
Рибаков В'ячеслав Олегович
Науковий керівник / Експерт
Залапін Олексій Ігорович

підрозділ
Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету"

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

25	13485	103487
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	25
Інтервали	A→	0
Мікропробіли	·	168
Білі знаки	ß	0
Парафрази (SmartMarks)	a	54

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз		Колір тексту
ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	106 0.79 %
2	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	65 0.48 %
3	https://card-file.ontu.edu.ua/bitstreams/b1c4b329-c3e8-4b5a-a1fc-ae232ec677bd/download	45 0.33 %
4	https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download	37 0.27 %
5	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	35 0.26 %

6	https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download	34 0.25 %
7	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	34 0.25 %
8	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	31 0.23 %
9	https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download	28 0.21 %
10	https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content	26 0.19 %

з домашньої бази даних (0.31 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	Проектування локальної мережі з точками доступу для навчального корпусу ВСП "ОТФК ОНТУ" [redacted] 6/15/2025 Odesa Technical Professional College of Odesa National University of Technology (Відокремлений структурний підрозділ "Одеський технічний фаховий коледж Одеського національного технологічного університету")	42 (3) 0.31 %

з програми обміну базами даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з Інтернету (6.08 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)
1	**https://card-file.ontu.edu.ua/server/api/core/bitstreams/c63b91ba-d04f-4715-890d-b16277695c7e/content**	**267 (9) 1.98 %**
2	**https://card-file.ontu.edu.ua/bitstreams/0e72a3b9-bdd7-4711-a3c6-dedc1d4287cc/download**	**257 (15) 1.91 %**
3	**https://card-file.ontu.edu.ua/bitstreams/82a6d375-2b69-4233-b80f-fbfd149b7747/download**	**62 (3) 0.46 %**
4	**https://card-file.ontu.edu.ua/bitstreams/b1c4b329-c3e8-4b5a-a1fc-ae232ec677bd/download**	**45 (1) 0.33 %**
5	**https://card-file.ontu.edu.ua/bitstreams/53ed22ad-8700-4162-b97a-082a1ad472d6/download**	**34 (1) 0.25 %**
6	https://card-file.ontu.edu.ua/bitstreams/bbaf3f38-16a8-4070-bead-5562769b7c71/download	28 (2) 0.21 %
7	https://card-file.ontu.edu.ua/bitstreams/549ee9fe-7574-4ae5-b500-9fe2711f33e6/download	24 (2) 0.18 %
8	https://card-file.ontu.edu.ua/bitstreams/f789da43-3034-4ad8-bf34-640a47414f93/download	22 (1) 0.16 %
9	https://www.nbtuit.uz/article/download/6	18 (2) 0.13 %
10	https://randomnerdtutorials.com/esp8266-relay-module-ac-web-server/	15 (2) 0.11 %
11	https://mehanika.net.ua/product/modul-stilnikovogo-zvyazku-gsm-i-gprs-dlya-arduino-sim800l-evb-z-vinosnoyu-antenoyu/	13 (1) 0.10 %
12	https://arduino.ua/index.php?categoryID=122&show_all=yes	12 (1) 0.09 %
13	https://lastminuteengineers.com/nrf24l01-arduino-wireless-communication/	12 (2) 0.09 %
14	https://wiki.lpnu.ua/wiki/%D0%A2%D0%B5%D1%81%D0%BB%D1%8E%D0%BA_%D0%92%D0%B0%D1%81%D0%B8%D0%BB%D1%8C_%D0%9C%D0%B8%D0%BA%D0%BE%D0%BB%D0%B0%D0%B9%D0%BE%D0%B2%D0%B8%D1%87	11 (1) 0.08 %

Список прийнятих фрагментів (немає прийнятих фрагментів)