Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Програмно-апаратна система керування лабораторними стендами на Raspberry Pi
Назва теми

Рівень вищої освіти *перший (бакалаврський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Шифр КвРКІ 220063.22.02.15 ПЗ

Виконав здобувач IV курсу, група КІ2-22-2 _____ Андрій МУКОМЕЛА
Підпис Ініціали, прізвище

Керівник д.т.н., проф. _____ Василь ЯЦКІВ
Науковий ступінь, учене звання Підпис Ініціали, прізвище

Нормоконтролер канд.фіз.-мат.наук, доц. _____ Тетяна КИСІЛЬ
Науковий ступінь, учене звання Підпис Ініціали, прізвище

До захисту допускаю:
завідувач кафедри КІІС _____ Ольга ПАВЛОВА
«01» червня 2026 р. підпис Ініціали, прізвище
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Рівень вищої освіти перший (бакалаврський)

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІІС
_____ Ольга ПАВЛОВА

" 10 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мукомела Андрій Миколайович
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно-апаратна система керування лабораторними стендами на Raspberry Pi
Керівник проекту (роботи) Яцків Василь Васильович, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 20.01.2026 р. № 7

2. Термін подання здобувачем роботи на кафедру 01.06.2026 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Теоретичні основи побудови систем віддаленого доступу до лабораторних стендів
Аналіз вимог та проєктування програмно-апаратної системи керування лабораторними стендами
Реалізація та дослідження роботи системи на базі Raspberry Pi

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Діаграми роботи системи віддаленого доступу до лабораторних стендів
Схеми функціонування основних модулів системи Raspberry Lab
Основні екрани інтерфейсу вебсистеми Raspberry Lab для керування лабораторними стендами

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня

7. Дата видачі завдання «_10_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітк
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2026	виконан
3	Робота над розділом 1 – теоретичні основи побудови систем віддаленого доступу до лабораторних стендів	01.03.2026	виконан
4	Робота над розділом 2 – аналіз вимог та проєктування програмно-апаратної системи керування лабораторними стендами	01.04.2026	виконан
5	Робота над розділом 3 – Реалізація та дослідження роботи системи на базі Raspberry Pi	29.04.2026	виконан
6	Оформлення пояснювальної записки згідно вимог	25.05.2026	виконан
7	Попередній захист ВКР	26.05.2026	викона
8	Захист ВКР на засіданні ЕК	Червень 2026 року	

Здобувач_____ Андрій МУКОМЕЛА
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи_____ Василь ЯЦКІВ
Підпис Ім'я, ПРІЗВИЩЕ

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 220063.22.02.15 ПЗ	Пояснювальна записка	74		
			Графічні матеріали			
2		КвРКІ 220063.22.02.15 Е8	Діаграми роботи системи	1		
			віддаленого доступу до			
			лабораторних стендів			
3		КвРКІ 220063.22.02.15 Е8	Схеми функціонування	1		
			основних модулів системи			
			Raspberry Lab			
4		КвРКІ 220063.22.02.15 Е8	Основні екрани	1		
			інтерфейсу вебсистеми			
			Raspberry Lab для			
			керування лабораторними			
			стендами			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ 220063.22.02.15 ВП		
Розробив		Мукомела			Відомість проекту	Літера	Аркуш 1 / Аркушів 1
Перевір.		Яцків				У	
						ХНУ, КІ2-22-2	
Н. контр.		Кисіль					
Затв.		Павлова		01.06			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно-апаратна система керування лабораторними стендами на Raspberry Pi».

Автор роботи: Андрій МУКОМЕЛА.

Керівник роботи: Василь ЯЦКІВ.

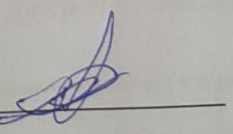
Пояснювальна записка: 74 с., 13 рис., 26 табл., 3 дод., 53 джерел.

Графічна частина: 3 креслення.

АГЕНТ, АРХІТЕКТУРА, БАЗА ДАНИХ, КЕРУВАННЯ СТЕНДАМИ, МОНІТОРИНГ, ТЕРМІНАЛ.

Кваліфікаційна робота бакалавра присвячена розробці та дослідженню програмно-апаратної системи керування лабораторними стендами з використанням одноплатних комп'ютерів Raspberry Pi. Актуальність теми зумовлена зростанням потреби у віддаленому доступі до навчального обладнання, забезпеченні безперервності освітнього процесу та ефективному використанні матеріально-технічної бази закладів освіти. Використання віддалених лабораторій дає змогу студентам виконувати практичні роботи незалежно від їхнього місцезнаходження, оптимізувати використання обладнання, а також підвищити рівень автоматизації навчального процесу.

Метою роботи є проєктування, реалізація та дослідження програмно-апаратної системи, що забезпечує віддалений доступ користувачів до лабораторних стендів на базі Raspberry Pi з використанням веб-інтерфейсу. Для досягнення поставленої мети в роботі виконано аналіз сучасних підходів до побудови систем віддаленого доступу та керування обчислювальними ресурсами, обґрунтовано вибір апаратної платформи та програмних засобів.



Підпис здобувача

30.05.2026

Дата

ЗМІСТ

ВСТУП

У наш час, коли інформаційні технології та цифровізація освіти активно розвиваються, рішення, що забезпечують віддалений доступ до навчальних матеріалів і лабораторного обладнання, стають особливо важливими. При звичайному проведенні лабораторних занять часто є свої обмеження: студентам треба бути особисто в спеціальних аудиторіях, кількість обладнання обмежена, і на все є чіткі часові рамки в навчанні. Через це стає важко забезпечити, щоб всі мали однаковий доступ до практичних занять, особливо коли навчання йде дистанційно або в змішаному форматі.

Одним із перспективних напрямів вирішення цієї проблеми є створення програмно-апаратних систем віддаленого доступу до лабораторних стендів. Такі системи дозволяють організувати взаємодію користувачів з реальним обладнанням через мережу Інтернет, що значно підвищує гнучкість навчального процесу, забезпечує ефективніше використання ресурсів та сприяє розвитку практичних навичок студентів незалежно від їхнього місцезнаходження.

Особливу роль у реалізації подібних рішень відіграють одноплатні комп’ютери, зокрема Raspberry Pi, які завдяки своїй доступності, енергоефективності та достатній обчислювальній потужності широко застосовуються в освітніх цілях, системах автоматизації та вбудованих рішеннях. Використання таких пристроїв як основи лабораторних стендів дозволяє створювати масштабовані та економічно ефективні системи віддаленого доступу.

Тема цієї кваліфікаційної роботи є особливо важливою, оскільки вона стосується кількох ключових аспектів. Йдеться про потребу ефективніше використовувати лабораторне обладнання, забезпечити стабільність освітнього процесу та активно інтегрувати сучасні інформаційні технології в навчання. Така система керування лабораторними стендами, що поєднує програмні та апаратні засоби, дозволяє досягти кількох цілей. Вона допомагає автоматизувати розподіл

ресурсів між студентами та викладачами, забезпечити надійний контроль доступу, значно підвищити рівень безпеки й ефективно централізувати управління всім обладнанням.

Метою роботи є проєктування, реалізація та дослідження програмно-апаратної системи керування лабораторними стендами на базі Raspberry Pi, яка забезпечує віддалений доступ користувачів до обчислювальних ресурсів через веб-інтерфейс.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1) проаналізувати сучасні підходи до організації віддалених лабораторій та систем доступу до обчислювальних ресурсів;

2) дослідити технології віддаленого керування пристроями та забезпечення безпечного доступу;

3) обґрунтувати вибір апаратної та програмної платформи;

4) розробити структуру та архітектуру системи;

5) спроєктувати базу даних для зберігання інформації про користувачів, пристрої та сесії;

6) реалізувати веб-портал для взаємодії користувачів із системою;

7) розробити програмний агент для керування лабораторними стендами;

8) забезпечити механізми автентифікації, авторизації та журналювання дій користувачів;

9) провести тестування та оцінити ефективність роботи системи;

Об’єктом дослідження є процеси організації віддаленого доступу до лабораторного обладнання та керування обчислювальними ресурсами в навчальних системах.

Предметом дослідження є методи, моделі та програмно-апаратні засоби реалізації системи керування лабораторними стендами на базі одноплатних комп’ютерів Raspberry Pi.

Практична цінність здобутих результатів полягає в можливості застосування розробленої системи в навчальних установах. Зокрема, вона

дозволяє організовувати дистанційні лабораторні заняття, сприяє підвищенню раціональності використання наявного обладнання, а також забезпечує ширший доступ до освітніх матеріалів для значної кількості користувачів.

1 ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ СИСТЕМ ВІДДАЛЕНОГО ДОСТУПУ ДО ЛАБОРАТОРНИХ СТЕНДІВ

1.1 Постановка задачі

За результатами проведеного аналізу предметної області, чітко простежується потреба у створенні ефективної програмно-апаратної системи. Вона покликана забезпечити дистанційний доступ до лабораторного обладнання та дозволити централізовано управляти наявними ресурсами. Подібна система повинна усувати обмеження, властиві традиційним лабораторіям, підвищуючи доступність навчальних матеріалів і сприяючи безпечній взаємодії користувачів із технічним оснащенням.

Основна мета цієї кваліфікаційної роботи – розробити, реалізувати та проаналізувати програмно-апаратну систему для керування лабораторними стендами. Ця система базується на одноплатних комп’ютерах Raspberry Pi і надає користувачам віддалений доступ до обчислювального середовища через веб-інтерфейс.

Для досягнення поставленої мети необхідно провести аналіз сучасних підходів до організації віддалених лабораторій, дослідити технології віддаленого доступу та забезпечення безпеки, обґрунтувати вибір апаратної платформи й програмних засобів, а також розробити архітектуру програмно-апаратної системи. Окрім цього, потрібно спроєктувати структуру бази даних для зберігання інформації про користувачів, пристрої та сесії, реалізувати веб-портал для взаємодії користувачів із системою, розробити програмний агент для керування лабораторними стендами, запровадити систему журналювання дій користувачів і впровадити механізми автоматичного завершення сесій.

Об’єктом дослідження є процеси організації віддаленого доступу до лабораторного обладнання та керування обчислювальними ресурсами.

Предметом дослідження є методи, моделі та програмно-апаратні засоби побудови системи керування лабораторними стендами на базі Raspberry Pi.

У рамках даної роботи передбачається розробка системи, яка повинна відповідати визначеним функціональним і нефункціональним вимогам. До функціональних вимог належать забезпечення реєстрації та автентифікації користувачів, підтримка ролей студента, викладача й адміністратора, можливість отримання доступу до лабораторного стенда, організація роботи через веб-термінал або SSH-доступ, ведення журналу дій користувачів, автоматичне завершення сесій після закінчення встановленого часу, а також відображення стану доступних пристроїв.

Нефункціональні вимоги передбачають забезпечення безпеки доступу до системи, її масштабованість у разі збільшення кількості користувачів, стабільність роботи при одночасному доступі кількох користувачів, зручність та інтуїтивність інтерфейсу, ефективне використання обчислювальних ресурсів і можливість подальшого розширення функціоналу системи.

Окрім цього, коли формулюємо завдання, ключовим етапом є визначення обмежень, які неодмінно треба брати до уваги при проєктуванні системи. Серед таких факторів можу виділити апаратні можливості одноплатних комп’ютерів, певні обмеження пропускної здатності мережевих каналів, а також жорсткі вимоги щодо безпеки та конфіденційності інформації.

1.2 Поняття віддалених лабораторій та їх роль у сучасній освіті

У наш час, коли інформаційні технології розвиваються дуже швидко, цифрові рішення активно проникають у багато сфер людської діяльності, включаючи й освіту. Одним з важливих кроків у зміні освітнього процесу стало запровадження дистанційних технологій навчання, які роблять навчальні матеріали доступними для студентів, незалежно від їхнього місця знаходження. Тому в цьому контексті дистанційні лабораторії набувають особливої важливості, адже вони є невід'ємною частиною практичної підготовки для майбутніх фахівців технічних спеціальностей.

Віддалена лабораторія, яку ще називають remote laboratory – це комплекс, що об'єднує програмне та апаратне забезпечення. Він дозволяє виконувати лабораторні завдання через інтернет, використовуючи або справжнє, або віртуальне обладнання. Основна риса таких систем полягає в тому, що користувач працює з обладнанням дистанційно, звертаючись до нього через веб-інтерфейс або за допомогою спеціальних програмних інструментів, при цьому без прямого фізичного контакту з самим пристроєм.

По суті, віддалені лабораторії можуть існувати у двох головних варіантах: це віртуальні лабораторії або ж лабораторії, які дають змогу дистанційно працювати зі справжнім обладнанням. Віртуальні лабораторії, як правило, працюють за принципом моделювання фізичних процесів, що дозволяє за допомогою спеціальних програм відтворювати, як саме поводяться різні системи. З іншого боку, лабораторії з віддаленим доступом дають можливість взаємодіяти зі справжніми фізичними пристроями, підключеними до мережі. Це, безумовно, підвищує як достовірність результатів досліджень, так і якість практичної підготовки студентів.

Використання віддалених лабораторій стає особливо актуальним у тих технічних дисциплінах, де студентам необхідно багато працювати з апаратним забезпеченням, мікроконтролерами, мережевими пристроями та іншими складовими комп'ютерних систем. Такі лабораторії дають змогу не тільки розширити можливості навчання, але й забезпечити доступ до дорогого або дефіцитного обладнання, що інакше було б важко отримати. Крім того, це ефективний спосіб організувати навчальний процес в умовах дистанційної освіти.

Однією з головних переваг віддалених лабораторій є їхня доступність. Для студентів це означає можливість виконувати лабораторні завдання в будь-який зручний час і з будь-якого місця, достатньо лише мати доступ до інтернету. Така гнучкість набуває особливого значення тоді, коли доступ до навчальних приміщень обмежений, або ж при організації змішаного чи повністю

дистанційного навчання. Крім того, віддалені лабораторії допомагають навчальним закладам більш ефективно використовувати наявні ресурси, оскільки одне й те саме обладнання може бути задіяне різними користувачами у різний час.

Не менш важливою перевагою є можливість автоматизувати процеси керування доступом до обладнання. Система здатна самостійно розподіляти ресурси між користувачами, контролювати час їхньої роботи, вести детальний журнал дій та забезпечувати необхідний рівень безпеки доступу. Це значно знижує навантаження на викладачів та технічний персонал, водночас підвищуючи загальну ефективність організації освітнього процесу.

Водночас впровадження віддалених лабораторій пов’язане з низкою викликів. Серед таких викликів – потреба гарантувати безпечний доступ та запобігти неправильному використанню обладнання. Окрім того, важливо мати стабільне мережеве з'єднання. Значною складністю є також розробка програмного забезпечення, що забезпечує взаємодію між користувачем та апаратним комплексом системи. Тому потрібно дуже ретельно підходити до механізмів перевірки особи користувача (автентифікації) та визначення його прав (авторизації). Так само важливою є ізоляція робочих середовищ, щоб уникнути будь-якого несанкціонованого втручання.

В умовах стрімкого технологічного розвитку, системи віддалених лабораторій, що базуються на одноплатних комп'ютерах, таких як Raspberry Pi, набувають значного поширення. Використання таких пристроїв дає змогу створювати гнучкі, масштабовані та економічно вигідні лабораторні стенди, які можна легко інтегрувати до єдиної інформаційної системи навчального закладу. Завдяки підтримці сучасних мережевих протоколів і можливості розгортання різноманітного програмного забезпечення, ці платформи виступають зручним інструментом для організації віддаленого доступу до лабораторного обладнання.

Таким чином, віддалені лабораторії є важливим інструментом модернізації освітнього процесу, що забезпечує поєднання теоретичних знань із практичними

навичками в умовах цифрового середовища. Їх впровадження сприяє підвищенню якості освіти, розширенню доступу до навчальних ресурсів та ефективнішому використанню матеріально-технічної бази навчальних закладів.

1.3 Архітектури клієнт-серверних систем

У сучасних інформаційних системах клієнт-серверна архітектура стала однією з основоположних ідей, коли йдеться про розробку програмного забезпечення. Її застосовують досить часто, наприклад, у веб-застосунках, корпоративних інформаційних системах, різних хмарних сервісах, а також у системах віддаленого доступу, таких як віддалені лабораторії. Цей підхід дає змогу ефективно розподіляти завдання між різними елементами системи. Завдяки цьому забезпечується її масштабованість, полегшується подальше обслуговування, а також підтримується надійний рівень безпеки.

В основі клієнт-серверної архітектури лежить розподіл системи на дві ключові частини: клієнтську та серверну(рис. 1.1). Клієнт – це, по суті, програма чи пристрій, який надсилає запити до сервера. Сервер, своєю чергою, опрацьовує ці запити, виконує потрібні розрахунки, взаємодіє з базою даних і повертає відповідь клієнту. Обмін даними між клієнтом і сервером, як правило, відбувається через мережу. Для цього вони послуговуються стандартними протоколами, такими як HTTP або HTTPS.

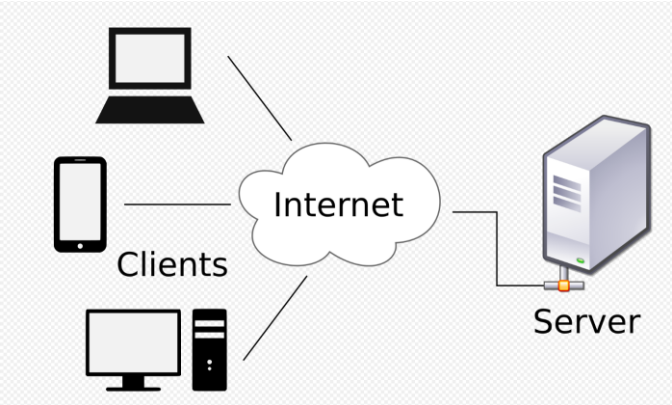


Рисунок 1.1 – Клієнт-серверна архітектура[51]

Якщо розглядати веб-орієнтовані системи, то клієнтом у них зазвичай є веб-браузер. Саме він дозволяє користувачеві взаємодіяти з інтерфейсом системи. Серверна частина, зі свого боку, виконує кілька ключових функцій: вона відповідає за обробку бізнес-логіки, контролює доступ, забезпечує зберігання даних і налагоджує інтеграцію з іншими складовими. Цей підхід забезпечує централізоване управління системою та спрощує процес оновлення програмного забезпечення, оскільки більшість змін впроваджується на серверній частині, і це не вимагає втручання на клієнтській стороні.

Багаторівнева, або як її ще називають, багатошарова архітектура, є своєрідним розвитком класичної клієнт-серверної моделі. Зокрема, йдеться про трирівневу архітектуру, що розбиває систему на кілька окремих логічних частин: це рівень відображення інформації, рівень, що займається логікою програми, та рівень даних. Так, рівень відображення безпосередньо взаємодіє з користувачем. Він включає у себе той інтерфейс, який ви бачите у веб-браузері або ж у спеціально створеній клієнтській програмі. Що стосується рівня логіки застосунку, то саме тут зосереджені основні алгоритми, що обробляють дані, керують сесіями, контролюють доступ та виконують інші системні функції. І нарешті, рівень даних піклується про те, щоб інформація зберігалася, оброблялася та була належним чином захищена. Для цього зазвичай використовуються системи управління базами даних.

Коли ми говоримо про багаторівневу архітектуру, варто виділити кілька її ключових переваг. Насамперед, вона значно покращує масштабованість системи: кожен з рівнів можна розгорнути на окремому сервері або навіть на групі серверів, що дозволяє легко нарощувати продуктивність, просто додаючи необхідні обчислювальні ресурси. По-друге, така архітектура спрощує як підтримку, так і подальшу модернізацію, оскільки зміни в одному її компоненті зазвичай не вимагають суттєвих переробок в інших частинах. І, по-третє, вона сприяє підвищенню безпеки: доступ до бази даних може бути обмежений виключно серверною частиною, що зменшує вектори для потенційних атак.

У системах віддаленого доступу до лабораторного обладнання, клієнт-серверна архітектура є надзвичайно важливою. Клієнт, що зазвичай представлений у вигляді веб-інтерфейсу, дає можливість користувачеві взаємодіяти із системою, авторизуватися, отримувати доступ до необхідних ресурсів та працювати з віддаленим обладнанням. Серверна частина, своєю чергою, відповідає за керування користувачами, розподіл ресурсів, контроль доступу та обробку всіх запитів. Крім того, сервер може виступати як посередник між користувачем і апаратною частиною системи, забезпечуючи цим додатковий рівень абстракції та підвищеної безпеки.

Однією з ключових особливостей систем для керування лабораторними стендами є потреба в додатковому рівні взаємодії з фізичними пристроями. Таким чином, окрім традиційних клієнтських та серверних компонентів, ці системи передбачають використання ще одного елемента — програмного агента. Він функціонує безпосередньо на самому пристрої, наприклад, на платі Raspberry Pi(рис. 1.2). Основним завданням цього агента є виконання команд, що надходять із сервера. Окрім того, він забезпечує доступ до ресурсів пристрою, а також передає результати виконаних операцій назад на сервер. У такий спосіб формується розширена клієнт-серверна архітектура, яка охоплює взаємодію не лише між клієнтом і сервером, але й із безпосередньою апаратною платформою.

Рисунок 1.2 – Плата Raspberry Pi 4 Model B[52]

У сучасних клієнт-серверних системах важливе місце посідає використання асинхронних механізмів для обміну даними, зокрема йдеться про технологію WebSocket. На відміну від класичної схеми “запит-відповідь”, WebSocket забезпечує двосторонній зв’язок у реальному часі між клієнтом та сервером. Такий підхід є критично важливим для створення веб-терміналів, а також систем віддаленого керування, де потрібно оперативно передавати команди користувача та швидко отримувати результати їх виконання.

1.4 Технології віддаленого доступу

В системах дистанційного керування лабораторними установками основну роль відіграють технології, що дозволяють користувачу взаємодіяти з обладнанням на відстані. Вибір цих технологій безпосередньо впливає на зручність використання, безпеку системи, швидкість її роботи та загальну ефективність функціонування всього програмно-апаратного комплексу. На сьогоднішній день у сучасних інформаційних системах найчастіше зустрічаються протоколи та технології, які забезпечують доступ до командного рядка, передачу даних у реальному часі, а також інтерактивну взаємодію з різними пристроями.

Серед основних засобів для віддаленого доступу варто виділити протокол SSH, відомий як Secure Shell(рис. 1.3). Його основна функція полягає у забезпеченні безпечного керування системами, розташованими віддалено, навіть якщо вони підключені через незахищені мережі. Ключовою перевагою SSH вважається його здатність застосовувати криптографічні методи. Це дозволяє шифрувати всі дані, що передаються, гарантуючи їхню конфіденційність та цілісність. Крім того, він надає можливість автентифікувати користувачів, використовуючи як звичайні паролі, так і криптографічні ключі. Завдяки SSH можна не лише виконувати різноманітні команди на віддаленому сервері, а й

безпечно передавати файли. Також він уможливлює організацію тунелювання мережевого трафіку.

Принцип роботи SSH полягає у встановленні захищеного з'єднання між клієнтом і сервером, після чого користувач отримує доступ до командного рядка віддаленої системи. У контексті лабораторних стендів це дозволяє студентам виконувати необхідні дії безпосередньо на пристрої, наприклад, запускати програми, змінювати конфігурацію або працювати з файлами.

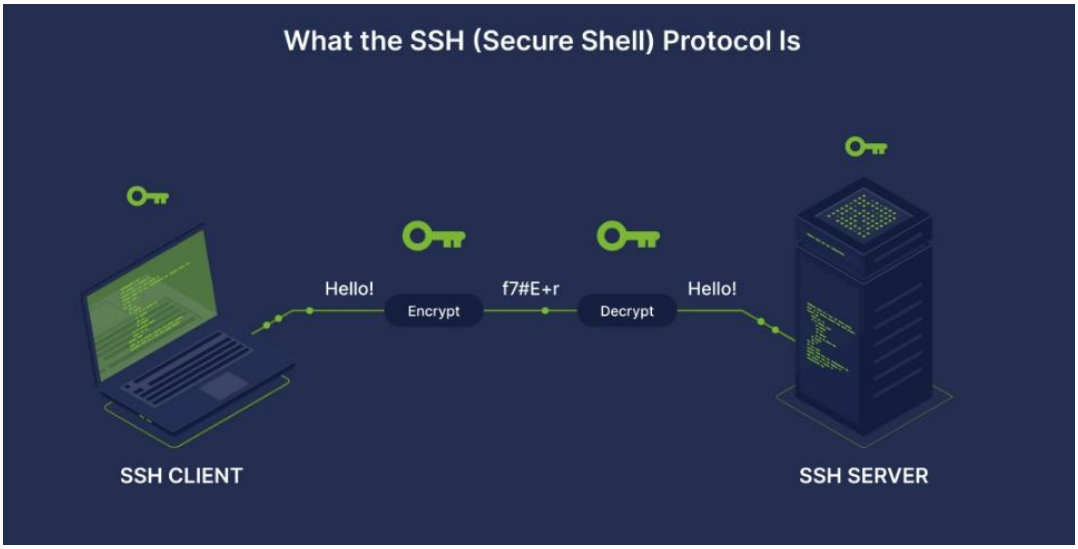


Рисунок 1.3 – Протокол SSH[53]

Однак, прямий доступ через SSH все ж має певні недоліки. Зокрема, це вимагає встановлення додаткового програмного забезпечення на пристрої користувача. Крім того, адміністрування значної кількості користувачів може бути ускладнене, і завжди існує ризик для безпеки, якщо доступ налаштовано некоректно. Враховуючи ці аспекти, сучасні системи дедалі частіше віддають перевагу веб-орієнтованим рішенням.

Як альтернативу пропонується використовувати веб-термінали, що дають змогу керувати віддаленою системою безпосередньо через браузер. Це рішення не вимагає від користувача встановлювати додаткові програми, а інтерфейс при цьому стає значно зручнішим. Принцип дії такого веб-терміналу простий:

команди від користувача передаються на сервер, який потім взаємодіє з віддаленим пристроєм і надсилає назад результат виконання.

Основну роль у реалізації веб-терміналів відіграє технологія WebSocket. Вона забезпечує постійне двостороннє з'єднання між клієнтом і сервером, що дозволяє передавати дані в режимі реального часу. На відміну від класичного HTTP-запиту, де кожна взаємодія ініціюється окремо, WebSocket встановлює постійний канал зв'язку, через який дані можуть передаватися без затримок.

Таблиця 1.1 – Схема взаємодії компонентів веб-терміналу

Крок	Компонент	Опис дії
1	Користувач (браузер)	Вводить команду в терміналі
2	Веб-клієнт	Передає команду через WebSocket
3	Сервер	Обробляє запит та передає його агенту
4	Raspberry Pi (агент)	Виконує команду в системі
5	Сервер	Отримує результат виконання
6	Веб-клієнт	Відображає результат у терміналі

Таким чином, веб-термінал виступає проміжною ланкою між користувачем і віддаленим пристроєм, забезпечуючи зручний і безпечний доступ до ресурсів системи.

Для кращого розуміння різниці між SSH та веб-терміналом доцільно провести їх порівняльний аналіз.

Таблиця 1.2 – Порівняння технологій віддаленого доступу

Характеристика	SSH	Веб-термінал
Тип доступу	Прямий	Через веб-браузер
Необхідність додаткового ПЗ	Так	Ні
Безпека	Висока	Висока (залежить від сервера)
Зручність використання	Середня	Висока

Кінець таблиці 1.2

Масштабованість	Обмежена	Висока
Контроль доступу	Складніший	Централізований

Окрім вже згаданих технологій, у сучасних системах часто застосовують і додаткові засоби для безпеки та контролю доступу. Серед них варто виділити системи автентифікації та авторизації. Важливим є також шифрування трафіку через протокол HTTPS, а ще фіксація всіх дій користувачів у журналах. Всі ці складові елементи, працюючи разом, утворюють цілісну систему захисту, яка має ключове значення при взаємодії з віддаленим обладнанням.

1.5 Одноплатні комп’ютери та їх застосування

У сучасних інформаційно-комунікаційних системах дуже важливу роль відіграють компактні комп'ютерні пристрої. Вони забезпечують необхідну продуктивність, ефективно використовують енергію і коштують недорого. Серед таких пристроїв виділяють одноплатні комп'ютери (англ. Single Board Computer, SBC(рис. 1.2)), які є повноцінними обчислювальними системами, реалізованими на одній друкованій платі. До їхнього складу входять центральний процесор, оперативна пам'ять, а також інтерфейси для введення-виведення та засоби для зберігання даних.

Одноплатні комп'ютери сьогодні широко використовуються в різних галузях. Вони є важливим елементом у системах автоматизації, знаходять своє місце в Інтернеті речей (IoT), застосовуються в робототехніці, вбудованих системах, а також активно залучаються до навчального процесу. Завдяки їхньому застосуванню можна суттєво зменшити витрати на розробку та реалізацію програмно-апаратних рішень, при цьому вони забезпечують достатню продуктивність для виконання широкого спектра завдань.

Raspberry Pi є одним із найпопулярніших одноплатних комп’ютерів(рис. 1.2). Цю платформу спочатку створювали, щоб зацікавити людей програмуванням та комп’ютерними науками в освітньому середовищі. Однак з часом вона стала активно використовуватися і в професійних проєктах. Головні переваги Raspberry Pi включають доступну ціну та широку підтримку програмного забезпечення. Крім того, варто відзначити активну спільноту розробників і значний набір інтерфейсів для підключення різноманітних зовнішніх пристроїв.

Raspberry Pi працює на операційних системах, які використовують ядро Linux. Це дає можливість застосовувати стандартні інструменти для системного адміністрування, програмування, а також для роботи з мережами. Пристрій має мережеві інтерфейси, зокрема Ethernet та Wi-Fi. Це дозволяє легко інтегрувати його як у локальні, так і у глобальні мережі, що є принципово важливим для організації віддаленого доступу. Додатково, завдяки підтримці протоколів для віддаленого керування, наприклад SSH, Raspberry Pi стає надзвичайно зручним інструментом для побудови лабораторних стендів.

Однією з ключових властивостей одноплатних комп’ютерів є те, що вони оснащені універсальними портами введення-виведення (GPIO). Ці порти дозволяють підключати різноманітні датчики, виконавчі механізми, а також інші електронні складові. Така функціональність робить їх корисними не лише для створення програмних проектів, а й для проведення експериментів, які вимагають взаємодії з апаратним забезпеченням. Це має особливе значення для навчання студентів технічних спеціальностей.

Для кращого розуміння можливостей одноплатних комп’ютерів доцільно розглянути їхні основні технічні характеристики на прикладі Raspberry Pi.

Таблиця 1.3 – Основні характеристики Raspberry Pi

Характеристика	Значення
Процесор	ARM (1–4 ядра залежно від моделі)

Кінець таблиці 1.3

Мережа	Ethernet / Wi-Fi
Інтерфейси	USB, HDMI, GPIO
Операційна система	Linux-based
Оперативна пам’ять	1–8 ГБ
Накопичувач	microSD

Порівняємо одноплатні комп’ютери з іншими типами обчислювальних пристроїв, які можуть використовуватися у навчальних лабораторіях.

Таблиця 1.4 – Порівняння обчислювальних платформ

Характеристика	Raspberry Pi	Мікроконтролер (Arduino)	Персональний комп’ютер
Тип пристрою	SBC	Мікроконтролер	Повноцінний ПК
Операційна система	Є	Немає	Є
Продуктивність	Середня	Низька	Висока
Споживання енергії	Низьке	Дуже низьке	Високе
Вартість	Низька	Дуже низька	Висока
Мережеві можливості	Є	Обмежені	Є
Підходить для віддаленого доступу	Так	Обмежено	Так

З проведеного порівняння стає зрозуміло, що одноплатні комп’ютери посідають місце десь посередині між мікроконтролерами та повноцінними комп’ютерами. Вони успішно поєднують у собі достатню продуктивність, щоб справлятися зі складними програмними завданнями, і водночас залишаються доступними за ціною та економними у споживанні енергії. Саме це робить їх оптимальним рішенням для створення систем віддалених лабораторій.

При розробці програмно-апаратної системи для керування лабораторними стендами, використання Raspberry Pi дозволяє створити гнучку інфраструктуру, де кожен пристрій виступає як окремий лабораторний вузол. Керування цими вузлами може здійснюватися централізовано через сервер, що забезпечує розподіл ресурсів між користувачами, контроль доступу та моніторинг загального стану системи.

1.6 Контейнеризація та ізоляція

У сучасних інформаційних системах, особливо там, де одночасно працює багато користувачів, дуже важливою є можливість ізолювати середовища, в яких виконуються завдання. Такий підхід дає змогу не тільки запобігти можливим конфліктам між користувачами, але й значно посилити безпеку всієї системи, що, у свою чергу, допомагає підтримувати її стабільну роботу. І ось тут контейнеризація виступає як один з найдієвіших інструментів для успішного виконання цього завдання.

Контейнеризація – це підхід до віртуалізації, що діє на рівні операційної системи. Він дає можливість запускати окремі, ізольовані середовища, відомі як контейнери, в межах однієї й тієї ж операційної системи. Кожен такий контейнер вміщує все потрібне для роботи програм: сам код, необхідні бібліотеки, всі залежності та файли налаштувань. При цьому цікаво, що контейнери спільно використовують ядро операційної системи. Це робить їх помітно легшими та швидшими, якщо порівнювати зі звичними віртуальними машинами.

Однією з найпоширеніших платформ для реалізації контейнеризації є Docker. Docker надає зручні інструменти для створення, запуску та керування контейнерами, що робить його стандартом де-факто у сфері розробки та розгортання програмного забезпечення.

По суті, контейнеризація покликана створювати ізольоване середовище для кожного користувача або для кожного окремого завдання. Якщо говорити

про системи віддалених лабораторій, це означає, що кожен студент отримує своє власне робоче середовище для виконання лабораторної роботи, і воно повністю незалежне від інших користувачів. Це запобігає ситуаціям, коли дії одного користувача могли б вплинути на роботу когось іншого.

Для кращого розуміння доцільно порівняти контейнеризацію з класичною віртуалізацією.

Таблиця 1.5 – Порівняння контейнерів та віртуальних машин

Характеристика	Контейнери (Docker)	Віртуальні машини
Рівень віртуалізації	ОС	Апаратний
Використання ресурсів	Низьке	Високе
Швидкість запуску	Висока	Низька
Ізоляція	Середня	Висока
Розмір середовища	Малий	Великий

Як видно з таблиці, контейнери є більш ефективними з точки зору використання ресурсів і швидкості запуску, що є важливим фактором для систем, де необхідно динамічно створювати та видаляти середовища для користувачів.

Архітектура контейнеризованої системи ґрунтується на кількох ключових елементах. Серед них – Docker Engine, який є відповідальним за функціонування та запуск контейнерів. Наступним компонентом є Docker Image – це певний шаблон, на основі якого створюється контейнер. І, нарешті, Docker Container, що являє собою безпосередньо виконуване середовище. Кожен контейнер функціонує в ізольованому режимі, проте зберігає можливість взаємодії з іншими контейнерами або зовнішніми системами за допомогою мережевих інтерфейсів.

Контейнеризація, що застосовується в системах керування лабораторними стендами, дає змогу створювати індивідуальні середовища для студентів. Так, наприклад, при підключенні до системи, для користувача автоматично

формується контейнер, який вже має попередньо налаштоване програмне забезпечення. По завершенні роботи такий контейнер можна видалити або перезапустити. Це забезпечує "чисте" середовище для наступного користувача.

Контейнеризація також значно спрощує процес налаштування системи. Завдяки Docker можна створювати стандартизовані образи, які легко переносити між різними пристроями, зокрема із серверів на Raspberry Pi. Це гарантує, що середовище виконання залишається однаковим, незалежно від того, яке використовується обладнання.

Контейнеризація також значно спрощує процес налаштування системи. Завдяки Docker можна створювати стандартизовані образи, які легко переносити між різними пристроями, зокрема із серверів на Raspberry Pi. Це гарантує, що середовище виконання залишається однаковим, незалежно від того, яке використовується обладнання.

Разом з тим, слід зазначити, що контейнеризація має й певні обмеження. Зокрема, їхній рівень ізоляції нижчий, ніж у віртуальних машин, оскільки всі контейнери спільно використовують одне ядро операційної системи. Ця особливість може призводити до появи додаткових ризиків у сфері безпеки, які обов'язково потрібно враховувати вже на етапі проєктування системи. Крім того, щоб використовувати контейнери ефективно, потрібен відповідний рівень підготовки як від розробників, так і від адміністраторів.

1.7 Безпека та контроль доступу в системах віддаленого доступу

При створенні програмно-апаратних систем для віддаленого доступу, забезпечення адекватного рівня безпеки виступає одним з основних завдань. Зважаючи на те, що ці системи дозволяють користувачам взаємодіяти з віддаленим обладнанням за допомогою Інтернету, стає очевидною потреба у захисті як програмних ресурсів, так і апаратної складової від несанкціонованого доступу, ненавмисних дій або зловмисних втручань.

Безпека систем віддаленого доступу зазвичай спирається на кілька ключових засад. До них належать автентифікація, авторизація, ретельний облік дій користувачів, а також надійний захист даних, що передаються. Кожна з цих складових відіграє істотну роль у підтриманні цілісності та забезпечення безперебійної роботи системи.

Автентифікація – це, по суті, процедура, під час якої система перевіряє, чи дійсно той користувач, котрий намагається до неї увійти, є тією особою, за яку він себе видає. Зазвичай для цього використовують добре відомі комбінації логіна та пароля, або ж вдаються до застосування криптографічних ключів. Проте, у системах, де безпека є критично важливою, нерідко впроваджують багатофакторні методи. Вони працюють таким чином, що для доступу потрібно одночасно використовувати кілька різних способів перевірки – наприклад, ввести пароль і додатково вказати одноразовий код.

Після успішної автентифікації користувача, коли його ідентичність підтверджена, відбувається авторизація – процес визначення його прав доступу до ресурсів системи. Для систем керування лабораторними стендами, як правило, ефективно використовувати рольову модель доступу. У такій моделі кожному користувачеві призначається конкретна роль (наприклад, студент, викладач або адміністратор), що безпосередньо визначає спектр доступних йому функцій. Цей підхід забезпечує не лише гнучке керування правами доступу, а й чітке розмежування функціональних можливостей системи.

Для посилення загального рівня безпеки системи критично важливим є регулярне ведення журналу дій користувачів, яке зазвичай називають логуванням. Ця функція дозволяє реєструвати всі операції, які відбуваються в межах системи, включаючи точний час авторизації, перелік виконаних команд, внесені зміни до конфігурації та інші релевантні події. Наявність таких записів надає можливість для всебічного аналізу функціонування системи, ефективного виявлення потенційних несправностей, а також забезпечує необхідну основу для проведення аудиту у випадках виникнення інцидентів, пов'язаних із безпекою.

При роботі з системами віддаленого доступу надзвичайно важливо забезпечити надійний захист даних, що передаються. З цією метою застосовують криптографічні протоколи, наприклад HTTPS, які відповідають за шифрування обміну даними між пристроєм користувача та сервером. Завдяки такому підходу вдається уникнути несанкціонованого перехоплення чутливої інформації, зокрема персональних облікових записів. Подібним чином, протокол SSH слугує для створення захищеного каналу зв'язку, що є критично важливим для дистанційного управління різними пристроями.

Контроль над сесіями користувачів також посідає важливе місце. Система повинна надавати функціонал для обмеження часу, протягом якого користувач може бути активним, а також автоматично завершувати сесії у випадку тривалої бездіяльності. Окрім цього, адміністратор повинен мати можливість примусово припиняти будь-яку сесію. Завдяки цим механізмам можна запобігти несанкціонованому використанню системних ресурсів, що значно підвищує загальний рівень безпеки.

В умовах використання спільного обладнання, зокрема лабораторних стендів на базі Raspberry Pi, ізоляція середовищ виконання є надзвичайно актуальною. Як було зазначено у попередньому розділі, контейнеризація дозволяє формувати незалежні середовища для кожного користувача. Це помітно знижує ризик взаємного впливу між ними та, відповідно, сприяє підвищенню загальної безпеки системи.

Розглянемо основні механізми забезпечення безпеки в системі у вигляді узагальненої таблиці.

Таблиця 1.6 – Основні механізми безпеки системи

Механізм	Призначення
Автентифікація	Підтвердження особи користувача
Авторизація	Визначення прав доступу
Шифрування (HTTPS)	Захист переданих даних

Кінець таблиці 1.6

SSH	Безпечний віддалений доступ
Журналювання	Фіксація дій користувачів
Контроль сесій	Обмеження часу доступу
Контейнеризація	Ізоляція середовищ

Крім технічних засобів, важливу роль відіграють організаційні заходи безпеки, такі як регулярне оновлення програмного забезпечення, контроль доступу до серверів, резервне копіювання даних та навчання користувачів основам безпечної роботи з системою.

Таким чином, забезпечення безпеки становить невід'ємний компонент систем дистанційного доступу до лабораторного обладнання. Застосування всебічного підходу, що передбачає використання таких сучасних технологій, як автентифікація, авторизація, шифрування та ізоляція середовищ, дозволяє сформувати надійну та стійку до різноманітних загроз систему, яка відповідає актуальним вимогам інформаційних технологій.

1.8 Висновки до першого розділу

У першому розділі було розглянуто теоретичні засади створення програмно-апаратних систем для віддаленого доступу до лабораторних стендів. З'ясувалося, що віддалені лабораторії є важливим напрямком розвитку сучасної освіти. Вони дають змогу виконувати практичні роботи, незалежно від місця перебування користувача, і сприяють ефективнішому використанню лабораторного обладнання.

Було проаналізовано клієнт-серверну архітектуру, адже вона є ключовою для створення веб-систем керування. Також розглянуто технології віддаленого доступу, такі як SSH, WebSocket і веб-термінали. Вони дозволяють користувачам взаємодіяти з віддаленими пристроями одразу, в реальному часі.

Окрему увагу приділено одноплатним комп’ютерам Raspberry Pi. Вони добре підходять для створення навчальних лабораторних стендів, адже доступні, компактні, енергоефективні та підтримують мережеві технології. Крім того, було розглянуто контейнеризацію. Вона допомагає ізолювати робочі середовища та робить роботу системи стабільнішою.

2 АНАЛІЗ ВИМОГ ТА ПРОЄКТУВАННЯ ПРОГРАМНО-АПАРАТНОЇ СИСТЕМИ КЕРУВАННЯ ЛАБОРАТОРНИМИ СТЕНДАМИ

2.1 Аналіз предметної області

В умовах стрімкого розвитку інформаційних технологій, а також повної цифровізації навчального процесу, особливої ваги набуває те, як ефективно організована практична підготовка для студентів. Спеціальності технічного профілю, зокрема в галузях комп’ютерних наук, електроніки та телекомунікацій, за своєю суттю вимагають проведення лабораторних робіт з реальним обладнанням. Адже саме завдяки таким роботам студенти мають змогу не тільки міцно закріпити набуті теоретичні знання, але й розвинути практичні навички взаємодії як з апаратним, так і з програмним забезпеченням.

Зазвичай, коли йдеться про організацію лабораторних робіт, ми послуговуємося традиційною моделлю, яка передбачає фізичні лабораторії. У таких умовах студенти отримують доступ до необхідного обладнання лише в певний час і тільки за умови їхньої фізичної присутності. Але цей підхід має чимало важливих обмежень. Насамперед, варто врахувати, що кількість доступного лабораторного обладнання, як правило, обмежена. Це часто ускладнює одночасну роботу для великої кількості студентів. До того ж, відведені для занять часові рамки не завжди дають змогу повністю завершити всі заплановані експерименти. І по-третє, брак віддаленого доступу істотно знижує гнучкість освітнього процесу, що особливо відчутно в умовах дистанційного навчання.

У зв'язку з цим з'являється потреба шукати нові підходи до організації лабораторних занять. Одним із таких рішень може бути використання віддалених лабораторій.

Такі системи дозволяють отримати доступ до обладнання через Інтернет, що суттєво розширює можливості навчального процесу.

Студенти тоді можуть виконувати лабораторні роботи у зручний для них час, не будучи прив’язаними до конкретного місця.

Огляд наявних рішень показує: у світі активно створюють системи віддалених лабораторій, які працюють як на віртуальних симуляторах, так і з реальним обладнанням. Хоча віртуальні лабораторії дозволяють моделювати роботу систем, вони не завжди здатні повністю відтворити реальні умови функціонування обладнання. А от системи, які дають доступ до фізичних пристроїв, допомагають досягти точніших результатів і водночас розвинути справжні практичні навички.

Платформа Raspberry Pi посідає помітне місце серед тих апаратних рішень, що застосовуються для облаштування віддалених лабораторій. Завдяки своїй універсальності, невисокій вартості та здатності працювати з сучасними технологіями, вона дозволяє конструювати гнучкі лабораторні стенди, які можна легко підключити до мережевої інфраструктури навчального закладу.

Разом із тим, впровадження систем віддаленого доступу до лабораторного обладнання пов’язане з низкою технічних і організаційних викликів. До основних проблем належать:

1. Необхідність ефективного розподілу ресурсів між користувачами.
2. Забезпечення безпечного доступу до обладнання.
3. Організація ізоляції середовищ виконання.
4. Контроль і обмеження часу роботи користувачів.
5. Забезпечення стабільності роботи системи при великій кількості підключень.

Крім того, варто зважати й на те, наскільки зручно користувачу працювати із системою. Звісно, інтерфейс має бути зрозумілим одразу, без зайвих пояснень, і не вимагати встановлення жодних додаткових програм. Важливо також, щоб він дозволяв швидко знаходити й використовувати потрібні функції. Саме тому доволі перспективною видається ідея використовувати веб-інтерфейси або,

скажімо, браузерні термінали. Завдяки їм можна взаємодіяти з системою просто через звичайний веб-браузер, не потребуючи нічого іншого.

Ще один ключовий момент – це потреба в єдиній, централізованій системі для керування всім комплексом. Така система повинна охоплювати не тільки контроль над доступом користувачів, а й фіксувати, які пристрої доступні, відстежувати їхній поточний стан, і, звичайно, вести докладні записи про всі проведені операції. Без такого єдиного центру управління, ефективно використовувати лабораторне обладнання стає вкрай складно, якщо не зовсім нереально.

2.2 Вимоги до системи

Коли створюють програмно-апаратну систему для керування лабораторними стендами, дуже важливо точно визначити, що саме вона має робити. Це необхідно, щоб система працювала так, як від неї очікують, і була справді ефективною. Ці вимоги зазвичай ділять на дві основні групи: функціональні та нефункціональні. Завдяки такому поділу можна повноцінно описати, що система робитиме (її поведінку) та наскільки добре вона це робитиме (її якісні характеристики).

Функціональні вимоги визначають перелік можливостей, які повинна реалізовувати система. Вони описують взаємодію користувачів із системою та основні процеси, що відбуваються під час її роботи.

До основних функціональних вимог належать:

1. Реєстрація користувачів у системі з можливістю створення облікового запису.
2. Автентифікація користувачів за логіном і паролем.
3. Підтримка ролей користувачів (студент, викладач, адміністратор).
4. Надання доступу до лабораторного стенда за запитом користувача.
5. Автоматичний вибір вільного пристрою зі списку доступних.

6. Відображення стану пристроїв (вільний, зайнятий, недоступний).

7. Забезпечення віддаленого доступу до середовища виконання (через веб-термінал або SSH).

8. Можливість виконання команд на віддаленому пристрої.

9. Ведення журналу дій користувачів.

10. Автоматичне завершення сесії після закінчення встановленого часу.

11. Завершення сесії за ініціативою користувача або адміністратора.

12. Очищення середовища після завершення роботи користувача.

13. Відображення інформації про активні сесії.

Таблиця 2.1 – Функціональні вимоги системи

№	Функція	Опис
1	Реєстрація	Створення облікового запису користувача
2	Авторизація	Вхід у систему за логіном і паролем
3	Управління ролями	Розподіл прав доступу
4	Видача ресурсу	Надання доступу до Raspberry Pi
5	Веб-термінал	Робота через браузер
6	SSH-доступ	Альтернативний спосіб підключення
7	Журналювання	Запис дій користувачів
8	Контроль сесій	Обмеження часу роботи
9	Моніторинг	Відображення стану пристроїв

Нефункціональні вимоги визначають якісні характеристики системи, такі як продуктивність, надійність, безпека та зручність використання.

Основні нефункціональні вимоги:

1. Безпека: система повинна забезпечувати захист даних користувачів, використання шифрування (HTTPS), а також контроль доступу до ресурсів.

2. Масштабованість: система повинна підтримувати збільшення кількості користувачів та пристроїв без суттєвого зниження продуктивності.

3. Надійність: система повинна стабільно працювати при тривалому використанні.

4. Продуктивність: мінімальні затримки при виконанні команд та обробці запитів.

5. Зручність використання: інтуїтивно зрозумілий інтерфейс без необхідності встановлення додаткового програмного забезпечення.

6. Сумісність: підтримка роботи в сучасних веб-браузерах.

7. Модульність: можливість розширення функціоналу системи без значних змін у її структурі.

Таблиця 2.2 – Нефункціональні вимоги системи

Категорія	Вимога	Опис
Безпека	Захист даних	Використання HTTPS, автентифікація
Продуктивність	Швидкодія	Мінімальні затримки
Надійність	Стабільність	Безперервна робота
Масштабованість	Розширення	Підтримка нових пристроїв
Зручність	UX/UI	Простий інтерфейс
Сумісність	Кросплатформеність	Робота в браузері

При розробці системи необхідно враховувати певні обмеження, які можуть впливати на її функціонування. Зокрема, слід брати до уваги обмежені обчислювальні ресурси пристроїв Raspberry Pi, залежність роботи системи від стабільності мережевого з’єднання та обмеження пропускної здатності мережі. Також важливими факторами є необхідність забезпечення безпеки під час роботи з віддаленим доступом і обмеження щодо одночасної кількості користувачів.

2.3 Архітектура системи

Архітектура програмно-апаратної системи, призначеної для керування лабораторними стендами, визначає не лише, як її основні компоненти взаємодіятимуть, а й встановлює принципи обміну даними та розподіляє функції між усіма складовими. Правильно спроєктована архітектура є вирішальним фактором. Вона гарантує масштабованість, надійність, безпеку та зручність у користуванні системою.

Пропонована система розроблена на базі клієнт-серверної архітектури, до якої додано окремий рівень для взаємодії з апаратними пристроями. Вона включає такі основні компоненти:

1) клієнт (веб-браузер користувача);
2) сервер (веб-додаток);
3) база даних;
4) лабораторні стенди на базі Raspberry Pi;
5) програмний агент, що працює на кожному пристрої.

Узагальнену структуру системи можна представити як багаторівневу модель, де кожен компонент виконує свою функцію.

Таблиця 2.3 – Основні компоненти системи

Компонент	Призначення
Клієнт (браузер)	Інтерфейс взаємодії користувача
Веб-сервер	Обробка запитів і керування логікою
База даних	Зберігання інформації
Raspberry Pi	Виконання команд користувача
Агент	Посередник між сервером і пристроєм

Користувач взаємодіє із системою через веб-інтерфейс, який реалізовано у вигляді веб-додатку. Усі запити користувача передаються на сервер, де

відбувається їх обробка. Сервер виконує перевірку прав доступу, визначає доступність ресурсів і організовує взаємодію з лабораторними стендами.

Сервер є центральним елементом системи, який забезпечує обробку запитів від клієнтів, автентифікацію та авторизацію користувачів, управління сесіями, розподіл лабораторних ресурсів, взаємодію з базою даних, а також обмін даними з агентами, встановленими на пристроях Raspberry Pi.

Серверна частина може бути реалізована з використанням сучасних технологій веб-розробки, що забезпечують підтримку асинхронної взаємодії та обробку великої кількості одночасних підключень.

Клієнтська частина представлена веб-інтерфейсом, який працює у браузері користувача та забезпечує зручну взаємодію із системою. Вона відповідає за відображення стану лабораторних стендів, передавання команд користувача на сервер, а також відображення результатів виконання команд у реальному часі.

Особливістю клієнтської частини є використання веб-терміналу, що дозволяє працювати з віддаленим пристроєм без встановлення додаткового програмного забезпечення.

База даних використовується для зберігання інформації, необхідної для функціонування системи. Вона містить дані про: користувачів, лабораторні пристрої, сесії користувачів та журнал дій.

Використання централізованої бази даних дозволяє забезпечити цілісність інформації та спростити управління системою.

Кожен лабораторний стенд реалізовано на базі Raspberry Pi, підключеного до мережі та доступного для віддаленого керування. На кожному пристрої працює спеціальний програмний агент, який забезпечує встановлення з’єднання із сервером, отримання команд від нього, виконання цих команд у системному середовищі та передавання результатів виконання назад на сервер.

Агент виступає посередником між сервером і апаратною частиною, забезпечуючи контрольований доступ до ресурсів пристрою.

Робота системи базується на взаємодії її компонентів, яка відбувається у кілька етапів:

1. Користувач проходить автентифікацію у веб-інтерфейсі.
2. Користувач запитує доступ до лабораторного стенда.
3. Сервер перевіряє доступність пристроїв.
4. Обирається вільний Raspberry Pi.
5. Сервер встановлює зв’язок із агентом на пристрої.
6. Користувач отримує доступ до середовища виконання.
7. Команди користувача передаються через сервер до пристрою.
8. Результати виконання повертаються користувачу.

Таблиця 2.4 – Послідовність взаємодії компонентів

Крок	Учасник	Дія
1	Користувач	Вхід у систему
2	Сервер	Перевірка даних
3	Сервер	Вибір пристрою
4	Агент	Виконання команд
5	Сервер	Передача результату
6	Клієнт	Відображення результату

2.4 Моделювання системи

Діаргама варіантів використання відображає взаємодію акторів із системою та основні функції, доступні для кожного актора.

Рисунок 2.1 – Діаргама варіантів використання

На рисунку 2.1 зображена UML-діаграма варіантів використання для системи керування лабораторними стендами/терміналами.

У системі є три основні ролі:

Студент може: реєструватися, авторизуватися, переглядати доступні стенди, отримувати доступ до стенда, працювати з терміналом, завершувати сесію, переглядати історію своїх сесій.

Викладач може: керувати користувачами, переглядати журнал дій.

Адмін може: керувати стендами, керувати сесіями, налаштовувати систему.

Також на діаграмі показані зв’язки `<<include>>` та `<<extend>>`, які означають залежність між діями. Наприклад, авторизація пов’язана з управлінням користувачами, а отримання доступу до стенда – з управлінням стендами. Робота з терміналом може розширюватися через управління сесіями.

Діаграма діяльності відображає процес роботи користувача з лабораторним стендом(рис. 2.2).

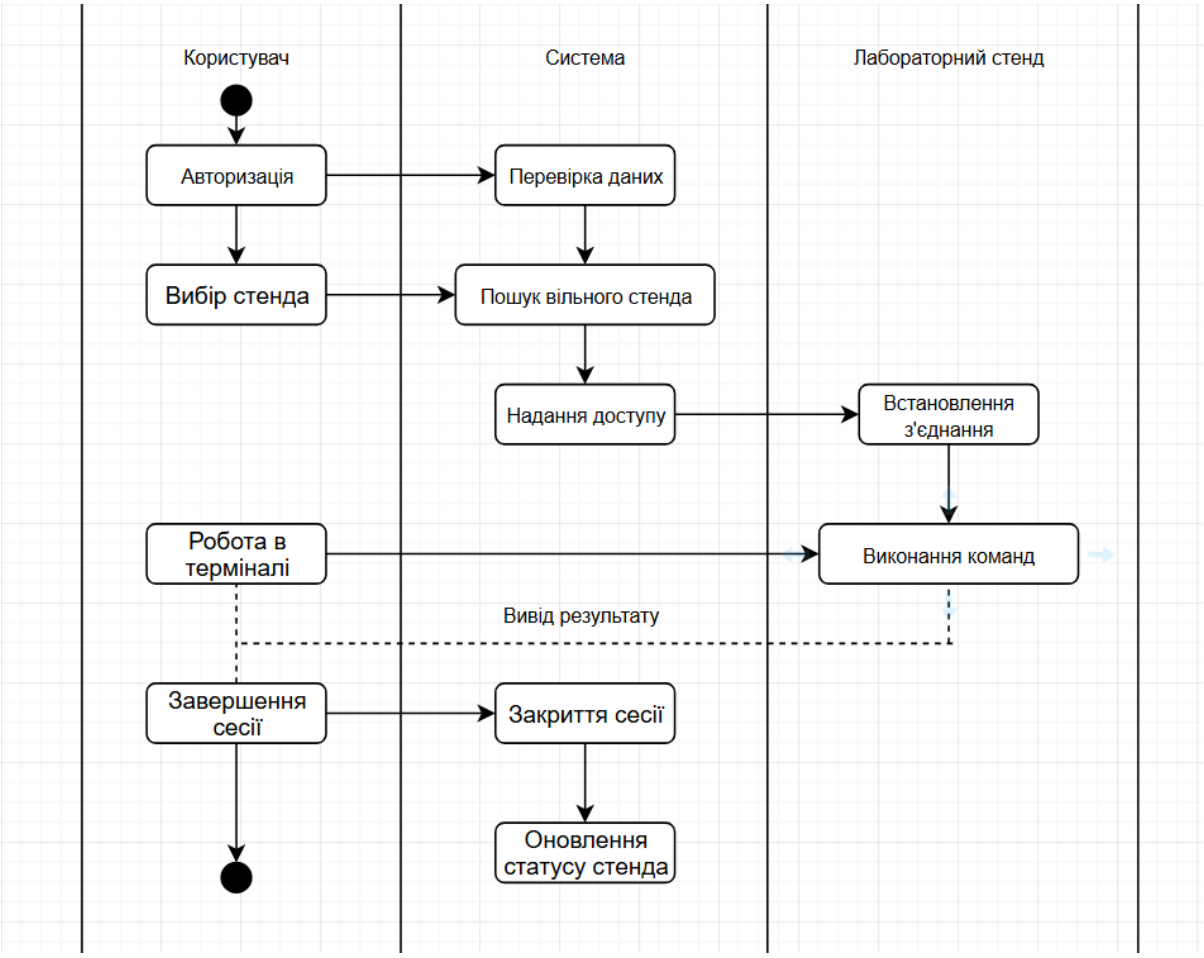


Рисунок 2.2 – Діаграма діяльності

Діаграма показує процес роботи користувача з лабораторним стендом через систему.

Користувач авторизується, обирає стенд, після чого система перевіряє дані, шукає вільний стенд і надає доступ. Далі встановлюється з’єднання зі стендом, користувач працює в терміналі, а стенд виконує команди та повертає результат. Після завершення сесії система закриває її й оновлює статус стенда.

Діаграма послідовності показує взаємодію між користувачем, веб-клієнтом, веб-сервером, базою даних і агентом Raspberry Pi(рис. 2.3).

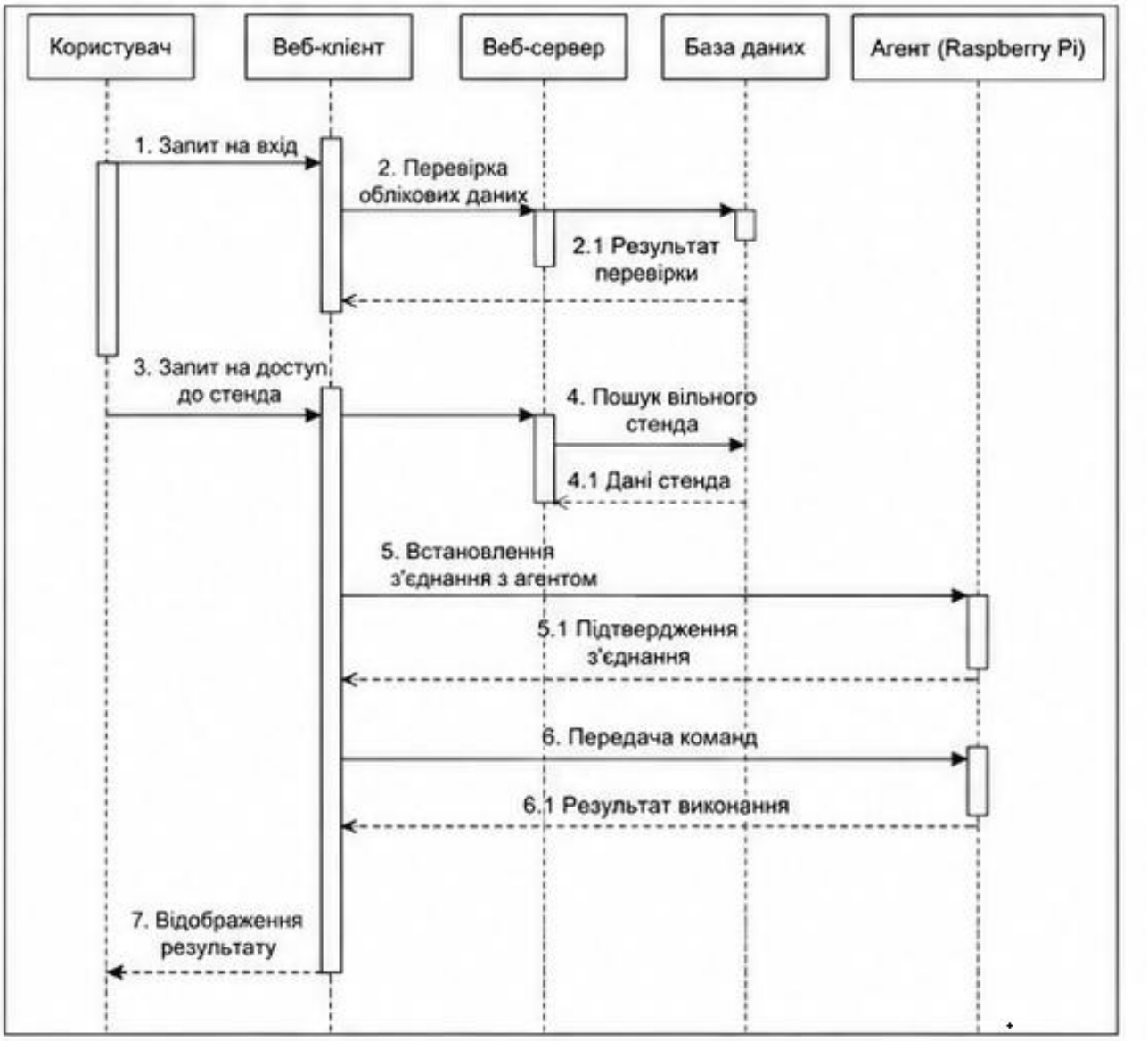


Рисунок 2.3 – Діаргама послідовності

Користувач надсилає запит на вхід, веб-сервер перевіряє облікові дані в базі даних і повертає результат. Потім користувач запитує доступ до стенда, система шукає вільний стенд, отримує його дані та встановлює з'єднання з агентом. Після підтвердження з'єднання веб-клієнт передає команди агенту, отримує результат виконання й відображає його користувачу.

2.5 Проєктування бази даних

Коли робиш якусь систему, де є і програма, і обладнання, то один з головних моментів – це придумати, як буде виглядати база даних. Сама база

даних відповідає за те, щоб зберігати, обробляти і керувати всією інформацією, яка потрібна, щоб система взагалі працювала. Від того, наскільки добре її продумаєш, залежить, як ефективно все функціонуватиме, чи швидко система оброблятиме твої запити, чи будуть дані в порядку і чи можна буде її потім легко доробити чи розширити.

У розроблюваній системі база даних використовується для зберігання інформації про користувачів, лабораторні пристрої, сесії доступу та журнал дій. Для реалізації бази даних доцільно використовувати реляційну модель, яка дозволяє структуровано організувати дані та забезпечити їх взаємозв'язок.

На основі аналізу вимог до системи було визначено основні сутності, які повинні бути представлені в базі даних:

1) користувачі (users);
2) ролі (roles);
3) лабораторні пристрої (devices);
4) сесії (sessions);
5) журнал дій (logs).

Кожна з цих сутностей відповідає окремій таблиці в базі даних.

Опис структури таблиць

Таблиця users: Зберігає інформацію про користувачів системи(табл.2.5).

Таблиця 2.5 – Структура таблиці users

Поле	Тип даних	Опис
id	INT (PK)	Унікальний ідентифікатор
username	VARCHAR	Логін користувача
password	VARCHAR	Хеш пароля
role_id	INT (FK)	Посилання на роль
created_at	DATETIME	Дата створення

Таблиця roles: Містить ролі користувачів(табл.2.6).

Таблиця 2.6 – Структура таблиці roles

Поле	Тип даних	Опис
id	INT (PK)	Ідентифікатор ролі
name	VARCHAR	Назва ролі

Таблиця devices: Зберігає інформацію про лабораторні стенди (Raspberry Pi(табл.2.7)).

Таблиця 2.7 – Структура таблиці devices

Поле	Тип даних	Опис
id	INT (PK)	Ідентифікатор пристрою
name	VARCHAR	Назва пристрою
ip	VARCHAR	IP-адреса
status	VARCHAR	Стан (free, busy)
last_seen	DATETIME	Остання активність

Таблиця sessions: Містить інформацію про активні та завершені сесії(табл.2.8).

Таблиця 2.8 – Структура таблиці sessions

Поле	Тип даних	Опис
id	INT (PK)	Ідентифікатор сесії
user_id	INT (FK)	Користувач
device_id	INT (FK)	Пристрій
start_time	DATETIME	Початок сесії
end_time	DATETIME	Завершення
status	VARCHAR	Активна/завершена

Таблиця logs: Зберігає журнал дій користувачів(табл.2.9).

Таблиця 2.9 – Структура таблиці logs

Поле	Тип даних	Опис
id	INT (PK)	Ідентифікатор запису
user_id	INT (FK)	Користувач
action	TEXT	Виконана дія
timestamp	DATETIME	Час виконання

Для забезпечення цілісності даних між таблицями встановлюються зв’язки:

1) один користувач має одну роль (users → roles);
2) один користувач може мати багато сесій (users → sessions);
3) один пристрій може використовуватись у багатьох сесіях (devices → sessions);
4) один користувач має багато записів у журналі (users → logs).

При проєктуванні бази даних було застосовано принципи нормалізації, що дозволяють уникнути дублювання даних і забезпечити їх цілісність. Структура бази даних відповідає третій нормальній формі (3NF), оскільки кожна таблиця містить дані лише про одну сутність, у ній відсутні транзитивні залежності, а всі неключові поля залежать лише від первинного ключа.

2.6 Висновки до другого розділу

У другому розділі детально проаналізовано й спроектовану програмно-апаратну систему для керування лабораторними стендами, яка працює на одноплатних комп'ютерах Raspberry Pi. Результати цієї роботи закладають як теоретичну, так і проєктну основу для подальшої реалізації системи, яку розглянемо вже у третьому розділі.

На першому етапі аналіз відповідної галузі чітко показав, що існує об'єктивна потреба у запровадженні віддалених лабораторій. При цьому були ідентифіковані ключові недоліки традиційних підходів до організації лабораторних занять. Зокрема, це стосується обмеженого доступу до устаткування, залежності від фізичної присутності та складності масштабування навчального процесу. Відтак, враховуючи виявлені проблеми, було обґрунтовано, що застосування віддаленого доступу до лабораторних стендів є перспективним сучасним рішенням для значного покращення ефективності навчального процесу.

Спершу було сформульовано завдання, а також чітко визначено мету, об'єкт і предмет дослідження. Далі окреслили основні завдання, які необхідно було вирішити в межах цієї роботи. Це дозволило не лише окреслити загальний напрямок розробки системи, але й точно визначити її головні функціональні можливості.

У розділі, який присвятили вимогам до системи, детально розписано функціональні та нефункціональні аспекти. Завдяки цьому вдалося сформувати цілісне уявлення про те, як працює система, які має характеристики та які її обмеження. Особливий акцент зроблений на вимогах щодо безпеки, продуктивності, масштабованості та зручності використання.

Під час формування архітектури системи було запропоновано багаторівневу клієнт-серверну модель, що включає програмний агент для роботи на лабораторних пристроях. Окрім того, чітко окреслено ключові компоненти системи, їх призначення та механізми взаємодії. Розроблена архітектура сприяє досягненню гнучкості, забезпеченню масштабованості та ефективному централізованому управлінню ресурсами.

Наступним етапом стало моделювання системи, використовуючи UML-діаграми. Це дало змогу наочно показати ключові сценарії її застосування, структуру класів та принципи взаємодії компонентів. Такий підхід допомагає

краще зрозуміти логіку роботи системи і значно спрощує процес її впровадження.

У завершальній частині розділу було розроблено структуру бази даних. Вона має забезпечувати зберігання та обробку відомостей про користувачів, пристрої, їхні сесії, а також весь журнал дій.

Для цього було спершу визначено основні сутності, їхні атрибути та зв’язки, що існують між ними. Крім того, щоб підтримати цілісність даних, були застосовані принципи нормалізації.

Отож, у другому розділі виконано повний цикл аналізу та проєктування системи, який включав визначення вимог, розробку архітектури, моделювання та проєктування бази даних. Отримані результати тепер слугують основою для практичної реалізації системи керування лабораторними стендами, яку детальніше розглянемо у наступному розділі.

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ РОБОТИ СИСТЕМИ НА БАЗІ RASPBERRY PI

3.1 Реалізація серверної частини системи

Серверна частина становить ключовий елемент програмно-апаратної системи, призначеної для керування лабораторними стендами. Вона забезпечує опрацювання запитів, які надходять від користувачів, здійснює їхню автентифікацію та авторизацію, а також безпосередньо керує лабораторними пристроями; крім того, на неї покладено створення та завершення сесій, взаємодію з базою даних, ведення журналу дій, а ще вона відповідає за організацію WebSocket-зв’язку між веб-терміналом і агентом, що працює на Raspberry Pi.

Для бекенду проєкту використано Node.js та фреймворк Express.js. Серверна частина має модульну структуру, що дозволяє рознести різні функції системи по окремих файлах та каталогах. Це спрощує підтримку коду, полегшує його тестування, і дає змогу в майбутньому легко розширювати систему новими можливостями.

Основним файлом ініціалізації HTTP-застосунку є app.js. У ньому створюється екземпляр Express-застосунку, підключаються middleware-компоненти та реєструються маршрути API. Для підвищення безпеки використовується бібліотека helmet, яка додає захисні HTTP-заголовки. Для дозволу взаємодії frontend-частини з backend використовується cors. Також підключено express.json() для обробки JSON-запитів і morgan для логування HTTP-запитів під час роботи сервера.

У файлі app.js також реалізовано маршрут перевірки працездатності системи: GET /api/health

Цей маршрут повертає відповідь про стан backend-сервера і може використовуватися для швидкої перевірки запуску системи.

Окрему роль виконує файл server.js. У ньому створюється HTTP-сервер, підключається WebSocket-модуль та запускається прослуховування порту. Крім того, саме у цьому файлі реалізовано періодичну перевірку активних сесій. За допомогою setInterval кожні 60 секунд викликається функція завершення прострочених сесій. Якщо сесія завершена через закінчення часу, сервер також закриває відповідний термінал.

1. Реалізація конфігурації backend.

Конфігураційні параметри серверної частини винесено у файл: backend/src/config/env.js

У ньому зчитуються змінні середовища, необхідні для роботи системи. До обов'язкових параметрів належать: DATABASE_URL(адреса підключення до PostgreSQL) та JWT_SECRET(секретний ключ для підпису JWT-токенів).

Також у конфігурації задаються:

– порт backend-сервера;
– середовище виконання;
– дозволені адреси клієнтської частини;
– тривалість сесії користувача.

Тривалість сесії задається через параметр SESSION_MINUTES, а якщо він не встановлений, використовується значення за замовчуванням – 30 хвилин.

2. Реалізація авторизації та автентифікації.

Для керування доступом користувачів у системі реалізовано механізм реєстрації, входу та перевірки поточного користувача. Цей функціонал винесено у файл: backend/src/routes/auth.routes.js

У системі передбачено такі маршрути: POST /api/register; POST /api/login; та GET /api/me.

Маршрут POST /api/register відповідає за створення нового користувача. При реєстрації сервер перевіряє, чи не існує вже користувача з таким іменем. Якщо ім'я вільне, пароль хешується за допомогою bcryptjs, після чого новий

запис додається до таблиці users. За замовчуванням новому користувачу призначається роль student.

Зберігання паролів користувачів у їхньому початковому, відкритому вигляді не практикується. Натомість, до бази даних вноситься лише криптографічне хеш-значення відповідного пароля. Такий підхід є критично важливою вимогою безпеки, оскільки навіть у ситуації несанкціонованого проникнення до сховища даних, справжні облікові дані користувачів залишатимуться недоступними для прямого отримання.

Маршрут POST /api/login виконує автентифікацію користувача. Сервер шукає користувача за іменем, після чого порівнює введений пароль із хешем у базі даних. Якщо дані правильні, користувачу повертається JWT-токен. Цей токен надалі використовується frontend-частиною для доступу до захищених маршрутів.

JWT-токен містить інформацію про ім’я користувача, роль та ідентифікатор користувача. Термін дії токена становить 8 годин. Завдяки цьому користувач може працювати із системою без повторного введення пароля при кожному запиті.

Для валідації вхідних даних використовується бібліотека zod. У маршрутах реєстрації та входу перевіряється, щоб ім’я користувача мало допустиму довжину, а пароль відповідав мінімальним вимогам. Це дозволяє зменшити кількість помилкових або некоректних запитів до сервера.

3. Middleware авторизації та рольової моделі

Перевірка доступу до захищених маршрутів реалізована у файлі: backend/src/middleware/auth.js

У цьому модулі реалізовано дві основні функції: Authenticate і requireRole.

Функція authenticate перевіряє наявність JWT-токена у заголовку Authorization. Токен повинен передаватися у форматі: Bearer <token>

Після отримання токена сервер перевіряє його справжність за допомогою секретного ключа JWT_SECRET. Далі з бази даних отримуються актуальні дані

користувача та його роль. Якщо токен відсутній, недійсний або користувача більше не існує, сервер повертає помилку доступу.

Функція requireRole використовується для обмеження доступу до окремих маршрутів залежно від ролі користувача. У системі передбачено три ролі:student, teacher і admin.

Наприклад, створення нового пристрою або перегляд журналів доступні лише користувачам з ролями admin або teacher.

4. Реалізація керування лабораторними пристроями.

Функціонал керування Raspberry Pi пристроями реалізовано у файлі: backend/src/routes/devices.routes.js

У цьому модулі передбачено маршрути: GET /api/devices; POST /api/request-device та POST /api/devices.

Маршрут GET /api/devices повертає список усіх лабораторних пристроїв. Для кожного пристрою відображається його ідентифікатор, назва, IP-адреса, статус, час останньої активності, а також інформація про активну сесію, якщо пристрій зайнятий.

У системі використовуються такі статуси пристроїв: free, busy та offline.

Маршрут POST /api/request-device відповідає за видачу вільного Raspberry Pi користувачу. При виклику цього маршруту backend звертається до сервісу сесій і намагається знайти доступний пристрій зі статусом free.

Маршрут POST /api/devices використовується для додавання нового лабораторного пристрою до системи. Цей маршрут захищений рольовою перевіркою, тому доступний лише користувачам з правами admin або teacher. При створенні пристрою сервер генерує унікальний agent_token, який у подальшому використовується агентом Raspberry Pi для підключення до WebSocket-сервера.

5. Реалізація керування сесіями.

Логіка створення, завершення та автоматичного закінчення сесій винесена у файл: backend/src/services/session.service.js. Цей модуль є одним із ключових у

серверній частині, оскільки саме він реалізує механізм розподілу лабораторних стендів між користувачами.

Основною функцією видачі пристрою є requestFreeDevice. Вона відкриває транзакцію в базі даних, перевіряє відсутність у користувача вже активної сесії, шукає вільний пристрій зі статусом free, створює новий запис у таблиці sessions, змінює статус пристрою на busy, записує дію користувача до журналу та завершує транзакцію.

Важливо, що при виборі пристрою використовується SQL-конструкція: FOR UPDATE SKIP LOCKED.

Вона дозволяє безпечно обробляти ситуацію, коли кілька користувачів одночасно запитують доступ до лабораторного стенда. Завдяки блокуванню рядків база даних не дозволяє видати один і той самий пристрій двом користувачам одночасно.

Функція endSession відповідає за завершення активної сесії. Завершити сесію може сам користувач, якому вона належить, або користувач із роллю admin чи teacher. Після завершення сесії її статус змінюється на ended, встановлюється час завершення, а пристрій переводиться у стан free, якщо він не перебуває у стані offline.

Для автоматичного завершення сесій реалізовано функцію expireSessions. Вона знаходить усі активні сесії, у яких час expires_at вже минув, переводить їх у статус expired, звільняє відповідні пристрої та створює записи у журналі. Ця функція періодично викликається з файлу server.js.

У системі сесії можуть мати такі статуси: active, ended та expired.

6. Реалізація журналювання дій.

Для фіксації дій користувачів використовується таблиця logs і сервіс: backend/src/services/log.service.js.

У ньому реалізовано функцію writeLog, яка додає запис до журналу. Журналювання використовується у різних частинах backend, зокрема при:

– реєстрації користувача;

– вході в систему;
– запиті пристрою;
– відкритті веб-терміналу;
– закритті веб-терміналу;
– завершенні сесії;
– автоматичному закінченні сесії;
– підключенні агента.

Перегляд журналів реалізовано у файлі: backend/src/routes/logs.routes.js. Для цього використовується маршрут: GET /api/logs

Цей маршрут доступний лише користувачам з ролями admin або teacher. Він повертає останні 200 записів журналу із зазначенням користувача, дії та часу виконання.

7. Взаємодія backend з WebSocket-модулем.

Окремим важливим компонентом серверної частини є WebSocket-модуль, розміщений у каталозі: backend/src/websocket/

Його основна задача – забезпечити двосторонній зв’язок між браузером користувача та агентом Raspberry Pi. У системі реалізовано два WebSocket-напрями: /ws/terminal і /ws/agent.

Канал /ws/terminal використовується для підключення веб-терміналу користувача. При підключенні перевіряється JWT-токен користувача та активність відповідної сесії. Якщо сесія активна і користувач має право доступу, сервер дозволяє підключення до терміналу.

Канал /ws/agent використовується для підключення агента Raspberry Pi. Агент автентифікується за допомогою agent_token, який зберігається в таблиці devices. Після підключення агента пристрій переводиться у статус free або busy залежно від наявності активної сесії.

Backend виступає посередником між веб-терміналом і Raspberry Pi. Команди, які користувач вводить у браузері, передаються серверу через

WebSocket, після чого сервер пересилає їх агенту. Агент виконує команди у термінальному середовищі та повертає результат назад через сервер до браузера.

Таблиця 3.2 – Основні маршрути backend API

Метод	Маршрут	Призначення
GET	/api/health	Перевірка стану backend
POST	/api/register	Реєстрація користувача
POST	/api/login	Авторизація користувача
GET	/api/me	Отримання даних поточного користувача
GET	/api/devices	Отримання списку пристроїв
POST	/api/request-device	Видача вільного Raspberry Pi
POST	/api/devices	Додавання нового пристрою
GET	/api/sessions	Отримання списку сесій
POST	/api/end-session	Завершення сесії
GET	/api/logs	Отримання журналу дій

Таблиця 3.3 – Основні серверні модулі системи

Модуль	Призначення
app.js	Налаштування Express-застосунку та REST API
server.js	Запуск HTTP-сервера, WebSocket і таймера сесій
auth.routes.js	Реєстрація, вхід і перевірка користувача
devices.routes.js	Керування лабораторними пристроями
sessions.routes.js	Отримання та завершення сесій
logs.routes.js	Перегляд журналу дій
auth.js	Перевірка JWT і ролей користувачів
session.service.js	Логіка видачі пристроїв і керування сесіями
log.service.js	Запис дій користувачів у журнал

websocket/index.js	Організація WebSocket-зв’язку між клієнтом і агентом

Серверна частина системи виконує основну бізнес-логіку програмно-апаратного комплексу. Вона відповідає за перевірку користувачів і розподіл прав доступу відповідно до їхніх ролей, керує пристроями Raspberry Pi, а також забезпечує відкриття та закриття робочих сесій, автоматичне вивільнення зайнятих ресурсів, ретельне журналювання всіх дій та налагоджує зв'язок із WebSocket-терміналом.

3.2 Реалізація бази даних

База даних становить собою один з основних елементів у програмно-апаратних системах, призначених для керування лабораторними стендами. Її функціонал передбачає централізоване накопичення інформації, яка охоплює відомості про користувачів, їхні повноваження (ролі), перелік лабораторних пристроїв, дані про активні та вже завершені сесії, а також детальний журнал усіх виконаних операцій. Для зберігання такої інформації в рамках даного проєкту було обрано реляційну систему керування базами даних PostgreSQL.

PostgreSQL добре підходить для цієї системи, адже вона має все потрібне для точного опису всіх її елементів: підтримує зовнішні ключі, забезпечує цілісність даних, працює з транзакціями, індексами та різними типами даних. Особливо важливо те, що система для управління лабораторними стендами має справу з дуже важливими станами ресурсів. Наприклад, пристрій може бути вільним, зайнятим чи недоступним, а сесія буває активною, завершеною або простроченою. Саме тому база даних має забезпечувати, щоб ці стани завжди були узгодженими.

1. Структура бази даних.

У реалізованій системі база даних складається з п’яти основних таблиць:

1) roles;
2) users;
3) devices;
4) sessions;
5) logs.

Кожна таблиця відповідає окремій сутності системи.

Таблиця 3.4 – Основні таблиці бази даних

Таблиця	Призначення
roles	Зберігання ролей користувачів
users	Зберігання облікових записів користувачів
devices	Зберігання інформації про Raspberry Pi
sessions	Зберігання даних про сесії доступу
logs	Зберігання журналу дій користувачів

2. Таблиця ролей користувачів.

Таблиця roles використовується для зберігання ролей користувачів системи. У проєкті передбачено три ролі: admin, teacher та student.

Ці ролі використовуються для розмежування прав доступу. Наприклад, студент може отримувати доступ до лабораторного стенда та працювати з веб-терміналом, тоді як викладач або адміністратор мають додаткові можливості перегляду журналів, сесій і керування пристроями.

Структура таблиці 3.5:

```
CREATE TABLE IF NOT EXISTS roles (
  id SERIAL PRIMARY KEY,
  name TEXT NOT NULL UNIQUE CHECK (name IN ('admin', 'teacher',
'student')));
```

Таблиця 3.5 – Структура таблиці roles

Поле	Тип даних	Призначення
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор ролі
name	TEXT UNIQUE	Назва ролі користувача

3. Таблиця користувачів.

Таблиця users містить облікові записи користувачів системи. Вона використовується під час реєстрації, авторизації та перевірки прав доступу.

Структура таблиці 3.6:

```
CREATE TABLE IF NOT EXISTS users (
  id SERIAL PRIMARY KEY,
  username TEXT NOT NULL UNIQUE,
  password_hash TEXT NOT NULL,
  role_id INTEGER NOT NULL REFERENCES roles(id),
  created_at TIMESTAMPTZ NOT NULL DEFAULT now());
```

Таблиця 3.6 – Структура таблиці users

Поле	Тип даних	Призначення
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор користувача
username	TEXT UNIQUE	Логін користувача
password_hash	TEXT	Хеш пароля
role_id	INTEGER	Посилання на роль користувача
created_at	TIMESTAMPTZ	Дата та час створення запису

4. Таблиця лабораторних пристроїв.

Таблиця devices використовується для зберігання інформації про лабораторні стенди на базі Raspberry Pi. Кожен запис у цій таблиці відповідає одному фізичному пристрою або лабораторному вузлу.

Структура таблиці 3.7:

```
CREATE TABLE IF NOT EXISTS devices (
```

```
        id SERIAL PRIMARY KEY,
        name TEXT NOT NULL,
        ip INET,
        status TEXT NOT NULL DEFAULT 'offline' CHECK (status IN
('free', 'busy', 'offline')),
        agent_token TEXT NOT NULL UNIQUE,
        last_seen TIMESTAMPTZ,
        created_at TIMESTAMPTZ NOT NULL DEFAULT now());
```

Таблиця 3.7 – Структура таблиці devices

Поле	Тип даних	Призначення
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор пристрою
name	TEXT	Назва лабораторного стенда
ip	INET	IP-адреса пристрою
status	TEXT	Поточний стан пристрою
agent_token	TEXT UNIQUE	Токен для автентифікації агента
last_seen	TIMESTAMPTZ	Час останньої активності агента
created_at	TIMESTAMPTZ	Дата створення запису

Для поля status передбачено три можливі значення: free, busy та offline.

5. Таблиця сесій доступу.

Таблиця sessions є однією з найважливіших у системі, оскільки вона зберігає інформацію про видані користувачам лабораторні ресурси. Кожна активна сесія означає, що певний користувач отримав доступ до певного Raspberry Pi.

Структура таблиці 3.8:

```
CREATE TABLE IF NOT EXISTS sessions (
        id SERIAL PRIMARY KEY,
        user_id INTEGER NOT NULL REFERENCES users(id) ON DELETE
CASCADE,
```

```
    device_id INTEGER NOT NULL REFERENCES devices(id) ON DELETE
CASCADE,
    start_time TIMESTAMPTZ NOT NULL DEFAULT now(),
    end_time TIMESTAMPTZ,
    expires_at TIMESTAMPTZ NOT NULL,
    status TEXT NOT NULL DEFAULT 'active' CHECK (status IN
('active', 'ended', 'expired')));
```

Таблиця 3.8 – Структура таблиці sessions

Поле	Тип даних	Призначення
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор сесії
user_id	INTEGER	Посилання на користувача
device_id	INTEGER	Посилання на пристрій
start_time	TIMESTAMPTZ	Час початку сесії
end_time	TIMESTAMPTZ	Час завершення сесії
expires_at	TIMESTAMPTZ	Час автоматичного завершення
status	TEXT	Стан сесії

У таблиці передбачено три стани сесії: active, ended та expired.

6. Таблиця журналу дій.

Таблиця logs використовується для фіксації важливих дій користувачів і системних подій. Журналювання необхідне для аналізу роботи системи, контролю використання лабораторних стендів і виявлення можливих помилок або порушень.

Структура таблиці 3.9:

```
CREATE TABLE IF NOT EXISTS logs (
  id SERIAL PRIMARY KEY,
  user_id INTEGER REFERENCES users(id) ON DELETE SET NULL,
  action TEXT NOT NULL,
  timestamp TIMESTAMPTZ NOT NULL DEFAULT now());
```

Таблиця 3.9 – Структура таблиці logs

Поле	Тип даних	Призначення
id	SERIAL PRIMARY KEY	Унікальний ідентифікатор запису
user_id	INTEGER	Посилання на користувача

Кінець таблиці 3.9

action	TEXT	Опис виконаної дії
timestamp	TIMESTAMPTZ	Дата та час виконання дії

7. Зв’язки між таблицями.

Між таблицями бази даних встановлено такі зв’язки:

1) roles 1 – * users;
2) users 1 – * sessions;
3) devices 1 – * sessions;
4) users 1 – * logs.

Це означає, що одна роль може належати багатьом користувачам, один користувач може мати багато сесій, один пристрій може використовуватись у багатьох сесіях у різний час, а один користувач може мати багато записів у журналі.

3.3 Реалізація клієнтської частини

Клієнтський інтерфейс системи, що забезпечує управління лабораторними стендами, розроблений для того, щоб користувачі могли легко взаємодіяти з backend-сервером, базою даних, а також віддаленими пристроями Raspberry Pi. Власне, завдяки цьому інтерфейсу користувач має можливість входити до системи, переглядати список доступних лабораторних стендів, отримувати доступ до відповідних Raspberry Pi, керувати поточною сесією та використовувати вбудований веб-термінал.

Клієнтську частину цього проєкту ми розробляли, використовуючи бібліотеку React, а для збірки залучили інструмент Vite. React дозволяє ефективно будувати інтерфейс, застосовуючи компонентний підхід, тоді як Vite значно прискорює процес запуску середовища розробки та формування фінального застосунку. Що стосується інших елементів, для веб-терміналу ми обрали бібліотеку xterm.js, а візуальні іконки інтегрували за допомогою lucide-react.

1. Загальна структура React-застосунку.

У frontend-частині реалізовано односторінковий застосунок, який змінює відображуваний інтерфейс залежно від стану авторизації користувача та наявності активної термінальної сесії.

Основним компонентом є App, який відповідає за перевірку збереженого токена користувача, збереження стану авторизації, перемикання між сторінкою входу, dashboard і терміналом, вихід користувача із системи, а також передачу даних про сесію до сторінки веб-терміналу.

При запуску застосунку виконується перевірка localStorage. Якщо в локальному сховищі браузера є JWT-токен і дані користувача, застосунок вважає користувача авторизованим і відкриває dashboard. Якщо дані відсутні, користувач переходить на сторінку входу або реєстрації(рис. 3.1).

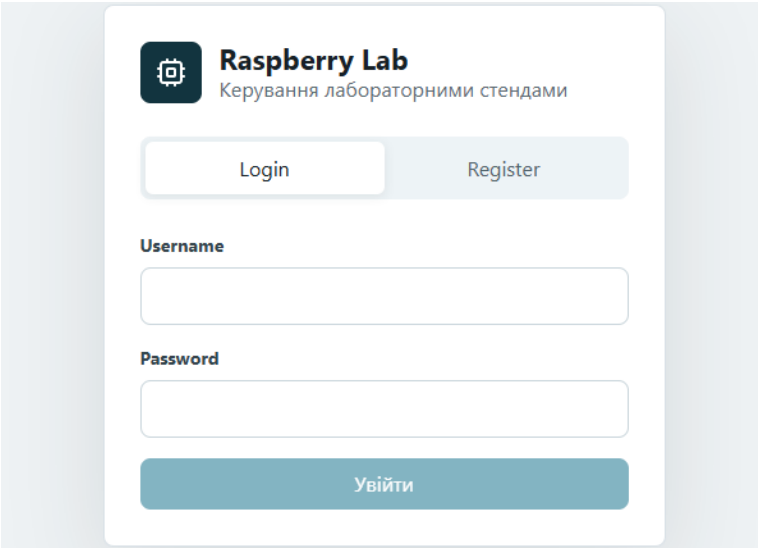


Рисунок 3.1 – Сторінка входу/реєстрації

2. Реалізація сторінки авторизації.

Для входу та реєстрації користувача реалізовано компонент AuthPage. Він містить форму з двома основними полями: Username і Password.

Також реалізовано перемикання між режимами: Login і Register.

У режимі Login користувач вводить свої облікові дані для входу в систему. У режимі Register створюється новий обліковий запис.

У випадку, коли серверна частина системи фіксує помилку, наприклад, через некоректно введений пароль або спробу реєстрації з уже використаним іменем користувача, відповідне повідомлення виводиться безпосередньо у формі для введення даних. Це забезпечує користувачеві чітке розуміння причин виникнення проблеми.

3. Реалізація головної панелі користувача.

Після успішної авторизації користувач переходить до головної панелі системи – компонента Dashboard. Цей компонент є основним робочим екраном системи та надає доступ до основних функцій(рис. 3.2).

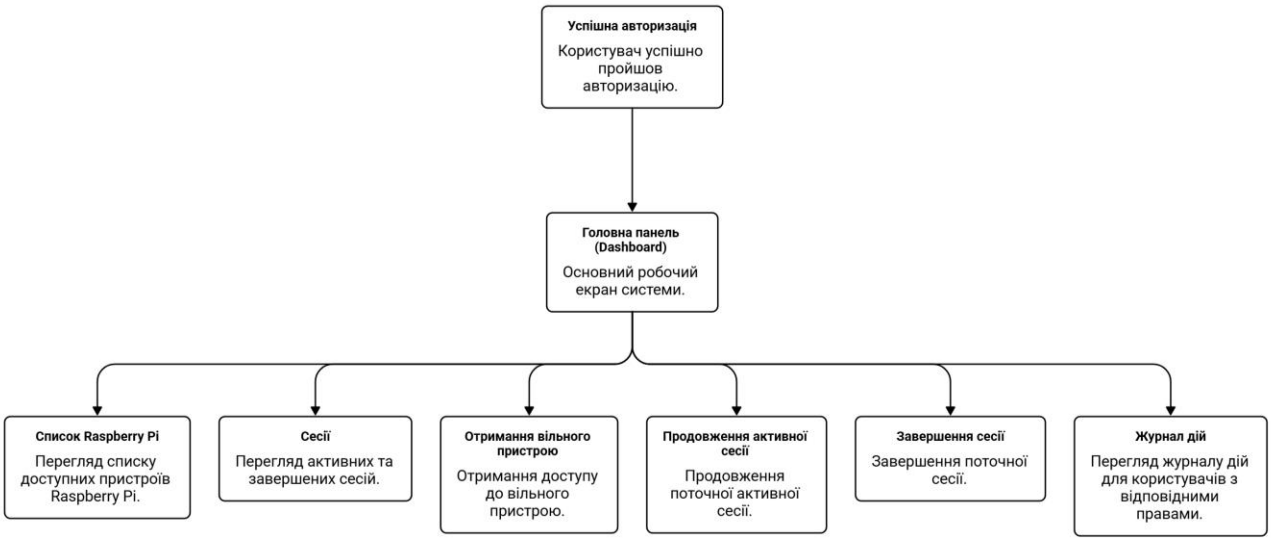


Рисунок 3.2 – Доступ до головної панелі

З метою підтримання актуальності даних інтерфейсу, система автоматично оновлює інформацію кожні 10 секунд. Це надає користувачеві можливість оперативно відстежувати зміни статусів пристроїв та сесій без необхідності ручного перезавантаження сторінки.

У верхній частині dashboard відображається інформація про поточного користувача: його ім’я та роль. Також розміщено кнопки оновлення даних і виходу із системи(рис. 3.3).

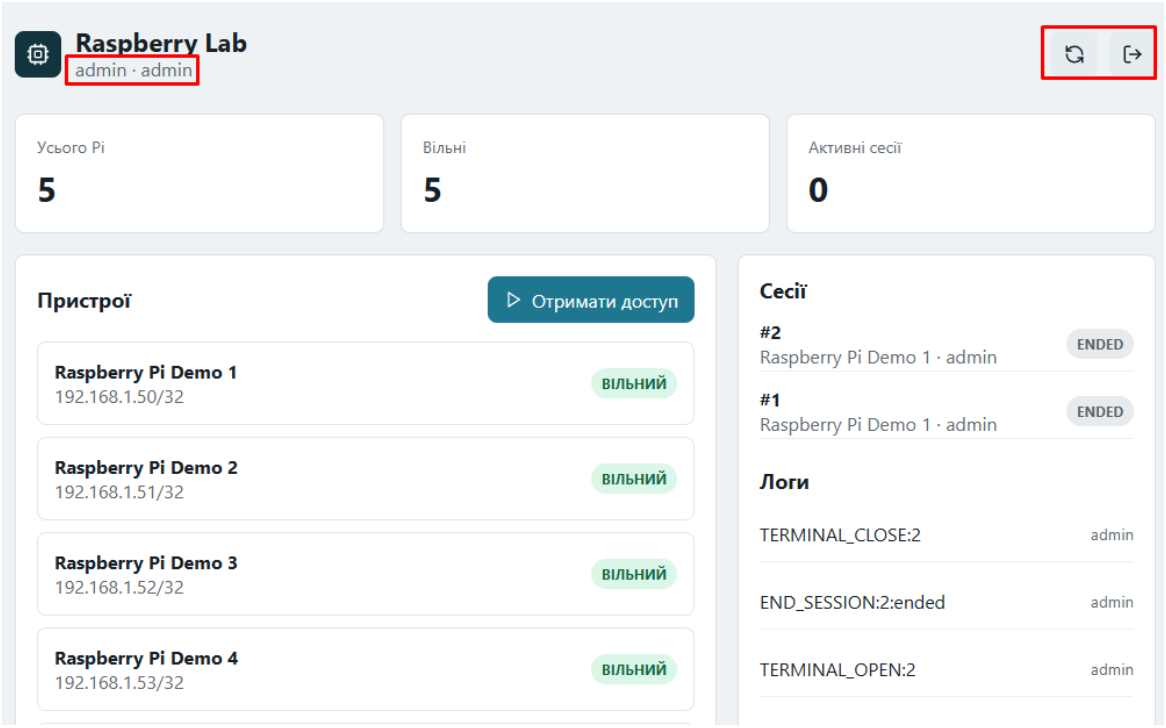


Рисунок 3.3 – Верхня частина dashboard

На головній панелі виводяться основні метрики: загальна кількість Raspberry Pi, кількість вільних пристроїв та кількість активних сесій.

Ці показники дають користувачу швидке уявлення про поточний стан лабораторної системи.

4. Відображення стану пристроїв.

Список лабораторних стендів відображається у вигляді карток пристроїв. Для кожного Raspberry Pi показується:

1) назва пристрою;

2) IP-адреса;

3) поточний статус;

4) ім’я активного користувача, якщо пристрій зайнятий.

У frontend-частині передбачено словник STATUS_LABEL, який перетворює технічні статуси backend у зрозумілі для користувача написи.

Якщо пристроїв у системі ще немає, у відповідному блоці виводиться повідомлення: пристроїв ще немає.

5. Отримання доступу до Raspberry Pi.

Для отримання доступу до лабораторного стенда користувач натискає кнопку: отримати доступ(рис. 3.4).

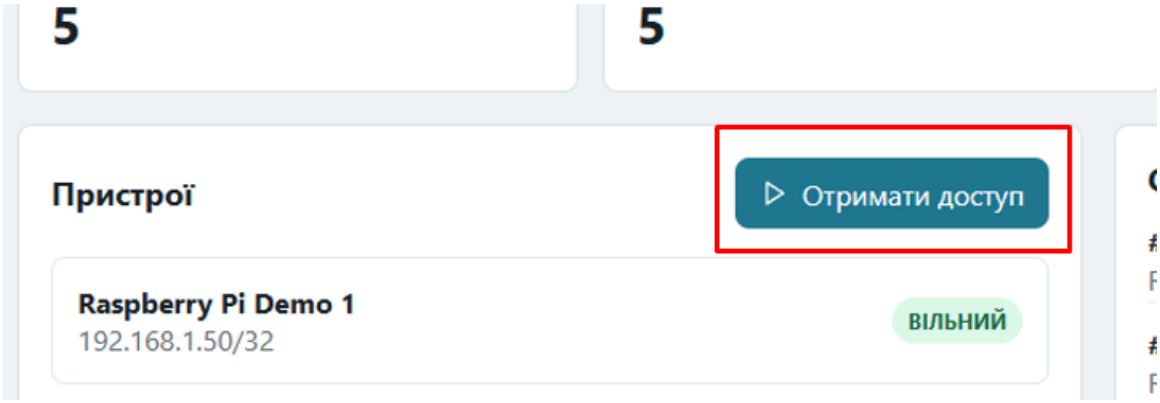


Рисунок 3.4 – Отримання доступу до лабораторного стенда

Backend перевіряє, чи є доступний Raspberry Pi, створює сесію і повертає дані про сесію та пристрій. Після успішної відповіді frontend відкриває сторінку веб-терміналу, передаючи їй інформацію про активну сесію.

Кнопка отримання доступу блокується, якщо: триває завантаження даних та у користувача вже є активна сесія.

Якщо у користувача вже є активна сесія, у dashboard відображається спеціальний блок, який містить номер сесії, назву пристрою, таймер залишку часу, кнопку продовження роботи та кнопку завершення сесії(рис. 3.5).

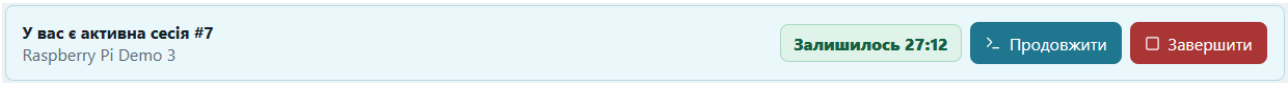


Рисунок 3.5 – Активна сесія у dashboard

6. Реалізація сторінки веб-терміналу.

Для роботи з віддаленим Raspberry Pi реалізовано компонент TerminalPage. Він відкривається після отримання пристрою або при продовженні активної сесії.

Після встановлення з’єднання користувач бачить повідомлення про підключення до Raspberry Pi. Якщо сервер надсилає повідомлення типу output, воно записується у термінал. Якщо користувач вводить дані в терміналі, вони відправляються на сервер у вигляді WebSocket-повідомлення типу input.

Коли відкривається сторінка, створюється термінал за допомогою бібліотеки xterm.js. Вона відповідає за те, щоб термінал відображався у веб-браузері: показує набраний текст, результати команд, системні повідомлення та інший вивід. Далі підключається FitAddon. Він автоматично підлаштовує розмір терміналу під доступну область екрана. Це робить роботу зручною, незалежно від розміру вікна браузера.

Після цього встановлюється WebSocket-з’єднання із сервером. Воно потрібне, щоб веб-клієнт і серверна частина системи, яка працює з Raspberry Pi, могли обмінюватися даними в реальному часі. Коли користувач вводить символи або команди в терміналі, ці дані через WebSocket потрапляють на сервер, а потім надсилаються до віддаленого пристрою.

Raspberry Pi, виконавши команди, надсилає результат на сервер. Сервер потім передає ці дані назад до компонента TerminalPage. Отримані дані одразу відображаються в терміналі, тому користувач бачить результат майже миттєво, як у звичайній локальній командній оболонці.

В кінці можна закрити сесію. Так можна зупинити роботу з віддаленим пристроєм, звільнити системні ресурси і вимкнути активне з'єднання.

Рисунок 3.6 – Алгоритм роботи веб-терміналу

7. Завершення сесії з інтерфейсу.

Frontend дозволяє завершити активну сесію двома способами: з dashboard і з сторінки веб-терміналу.

В обох випадках виконується запит: `POST /api/end-session`

Після завершення сесії backend змінює її статус, звільняє пристрій і записує подію до журналу. Frontend після цього оновлює дані або повертає користувача на dashboard.

Для користувачів з ролями admin або teacher передбачена можливість завершувати не лише власні, а й інші активні сесії.

8. Відображення журналу дій.

У dashboard реалізовано блок відображення журналу дій. Він доступний лише для користувачів із ролями: admin і teacher.

Звичайний студент не бачить цей блок. Це відповідає вимогам рольового доступу, описаним у попередніх розділах.

У журналі показуються останні записи, отримані з backend. Для кожного запису відображається: опис дії та ім’я користувача або позначення system, якщо дія не прив’язана до конкретного користувача.

3.4 Реалізація WebSocket-терміналу

Центральним елементом розробленої програмно-апаратної системи є веб-термінал. Він забезпечує користувачеві можливість інтерактивної взаємодії з лабораторним стендом на базі Raspberry Pi безпосередньо через веб-браузер. Це усуває необхідність встановлювати окремий SSH-клієнт або здійснювати налаштування прямого підключення до пристрою. Увесь робочий процес зручно реалізується через веб-інтерфейс системи.

Реалізація веб-терміналу базується на поєднанні трьох основних компонентів: frontend-компонента на основі xterm.js, WebSocket-модуля backend-сервера та програмного агента Raspberry Pi.

Основна ідея цього рішення полягає в тому, що браузер користувача не встановлює прямого з'єднання з Raspberry Pi. Натомість, взаємодія відбувається через спеціальний backend-сервер, який виконує функцію посередника. Такий метод значно підвищує безпеку всієї системи, оскільки усуває потребу прямого відкритого доступу студентів до Raspberry Pi ззовні.

1. Автентифікація користувача веб-терміналу.

Для підключення користувача до веб-терміналу frontend формує WebSocket-адресу такого вигляду:

```
/ws/terminal?token=<JWT_TOKEN>&sessionId=<SESSION_ID>
```

У параметрах передаються: token(JWT-токен авторизованого користувача) і sessionId(ідентифікатор активної сесії).

На стороні backend виконується функція автентифікації термінального підключення. Вона перевіряє:

1) чи передано JWT-токен;
2) чи передано коректний sessionId;
3) чи існує користувач із вказаного токена;
4) чи існує активна сесія з таким ідентифікатором;
5) чи має користувач право працювати з цією сесією;
6) чи не завершився час доступу до сесії.

Користувач отримує доступ до терміналу за наявності активної сесії, або якщо йому призначена роль адміністратора чи викладача. Такий механізм забезпечує студентам можливість працювати виключно зі своїм виділеним лабораторним стендом. Водночас викладачі та адміністратори отримують змогу контролювати відповідні сесії у межах наданих їм повноважень.

Якщо хоча б одна перевірка не проходить, WebSocket-з’єднання не створюється, а сервер повертає помилку 401 Unauthorized.

2. Автентифікація агента Raspberry Pi.

Агент Raspberry Pi підключається до backend через WebSocket-канал: `/ws/agent?token=<DEVICE_TOKEN>`

Для автентифікації використовується токен пристрою DEVICE_TOKEN, який відповідає полю agent_token у таблиці devices. Backend перевіряє цей токен у базі даних. Якщо пристрій із таким токеном знайдено, агенту дозволяється підключення.

Після успішного підключення агент зберігається у Map-структурі agents, де ключем є ідентифікатор пристрою. Це дозволяє backend швидко знаходити активного агента для конкретного Raspberry Pi.

Після підключення агента backend оновлює дані пристрою в базі:

1) поле last_seen отримує поточний час;
2) статус пристрою стає free, якщо активної сесії немає;
3) статус пристрою стає busy, якщо для цього пристрою вже існує активна сесія.

Якщо агент відключається, backend переводить пристрій у статус: offline

3. Контроль часу роботи терміналу.

Для кожної активної термінальної сесії backend створює таймер, який розраховується на основі поля expires_at. Якщо час сесії завершується, сервер надсилає користувачу повідомлення: [server] Session timer finished. Після цього WebSocket-з’єднання закривається.

Якщо сесія вже завершена або прострочена, backend надсилає повідомлення expired і закриває WebSocket-з’єднання.

4. Тестування WebSocket-терміналу.

Після створення сесії користувач переходить на сторінку веб-терміналу. Frontend встановлює WebSocket-з’єднання з backend за адресою: /ws/terminal?token= <JWT_TOKEN>&sessionId=<SESSION_ID>

Backend перевіряє JWT-токен, активність сесії та права користувача. Якщо перевірка проходить успішно, користувач отримує доступ до терміналу(рис. 3.6).

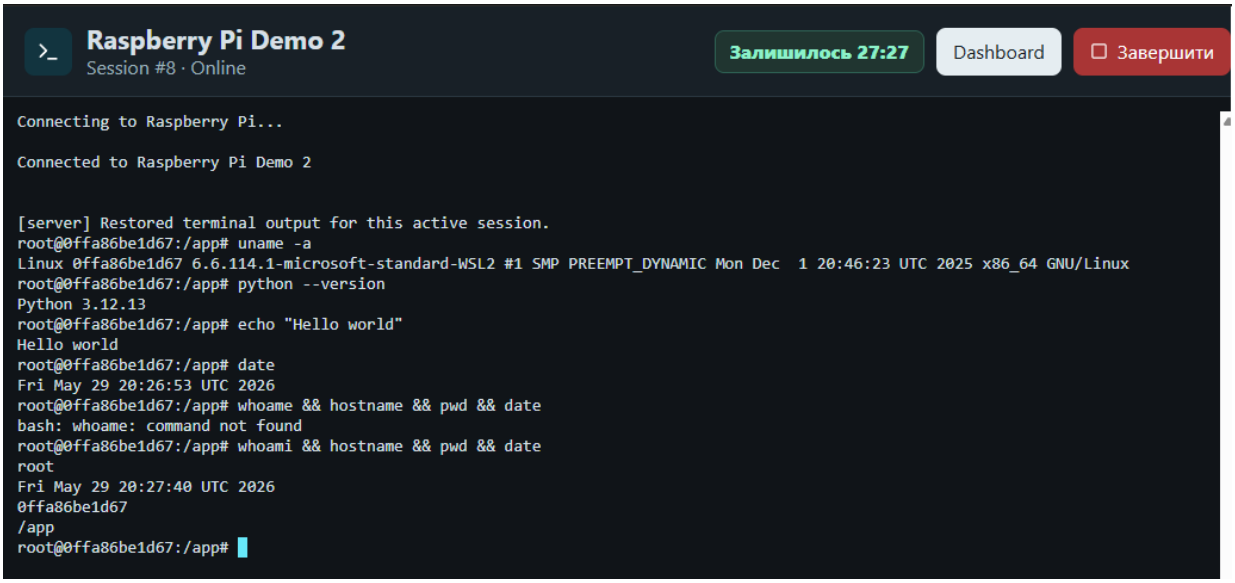


Рисунок 3.7 – WebSocket–термінал

У веб-терміналі були перевірені базові команди: pwd, ls, whoami, echo test, date.

Проведене тестування підтвердило, що веб-термінал забезпечує інтерактивну передачу команд і результатів виконання в режимі реального часу.

3.5 Висновки до третього розділу

У третьому розділі описано, як саме зроблено програмно-апаратну систему для керування лабораторними стендами на Raspberry Pi. Було обґрунтовано, чому вибрані такі технології та інструменти для розробки: Node.js, Express.js, PostgreSQL, React, Vite, WebSocket, xterm.js, Python та Docker. Цей набір технологій допоміг нам створити систему, яка включає веб-інтерфейс, серверну частину, базу даних, інтерактивний термінал і спеціальну програму для Raspberry Pi.

Для серверної частини розроблено застосунок, який дозволяє реєструвати й авторизувати користувачів. Він також забезпечує розмежування доступу за ролями, керування лабораторними пристроями, створення та завершення сесій, автоматичне звільнення ресурсів і ведення журналу дій. Доступ захищено JWT-

токенами, а паролі користувачів зберігаються у вигляді хешів, що підвищує загальний рівень безпеки системи.

Було реалізовано базу даних на PostgreSQL. У ній зберігається інформація про користувачів, їхні ролі, пристрої, сесії та всі записи (журнали). Завдяки такій будові, база даних забезпечує точність інформації, дозволяє стежити за станом Raspberry Pi та відстежувати, як використовували лабораторне обладнання раніше.

Клієнтська частина системи зроблена на React. Вона допомагає користувачам зручно працювати з системою. Через веб-інтерфейс можна зайти в систему, подивитися, які пристрої доступні, вибрати лабораторний стенд, працювати з ним через веб-термінал і закінчити сесію.

Особливу увагу було приділено створенню WebSocket-терміналу. Він забезпечує обмін даними в реальному часі між браузером, сервером і агентом на Raspberry Pi. Щоб виконувати команди на пристрої, ми розробили Python-агент. Він підключається до сервера, створює псевдотермінальне середовище та передає результати команд назад користувачеві.

ВИСНОВКИ

У кваліфікаційній роботі було спроєктовано та реалізовано програмно-апаратну систему керування лабораторними стендами на базі Raspberry Pi. Головна мета цієї роботи полягала в тому, щоб створити систему, яка б дозволяла студентам віддалено підключатися до лабораторного обладнання через веб-портал. Ця система також підтримує роботу з пристроєм через веб-термінал, контролює сесії користувачів і веде журнал їхніх дій.

У першому розділі було розглянуто, на чому базуються системи віддаленого доступу. З'ясовано, що таке віддалені лабораторії, чому вони важливі для сучасної освіти і які переваги дають у навчанні. Окрему увагу приділено на клієнт-серверній архітектурі та таких технологіях віддаленого доступу, як SSH, WebSocket і веб-термінали. Також було сказано, як використовувати Raspberry Pi для створення лабораторних стендів, а ще торкнулися контейнеризації, ізоляції робочих середовищ і безпеки.

У другому розділі було виконано аналіз предметної області та проєктування системи. Визначено головні проблеми зі звичайними лабораторними заняттями. Це, наприклад, обмежений доступ до обладнання, необхідність присутності студентів і труднощі з ефективним розподілом ресурсів. Виходячи з цього, було сформульовано мету, завдання, об'єкт і предмет дослідження, а також визначено функціональні та нефункціональні вимоги до системи.

Під час проєктування було розроблено архітектуру системи. Вона включає клієнтську частину, backend-сервер, базу даних PostgreSQL, модуль WebSocket та програмний агент для Raspberry Pi. Систему також змодельовано за допомогою UML-діаграм. Крім того, було спроєктовано структуру бази даних, де зберігається інформація про користувачів, їхні ролі, пристрої, сесії та журнал дій.

Третій розділ описує, як система реалізована на практиці. Серверна частина зроблена на основі Node.js та Express.js. Вона забезпечує реєстрацію та авторизацію користувачів, розподіл доступу за ролями, керування пристроями, створення та завершення сесій, ведення журналу дій і взаємодію через WebSocket. Для захисту доступу використовують JWT-токени, а паролі користувачів зберігаються у вигляді хешів.

Дані зберігаються у PostgreSQL. У цій базі є таблиці для користувачів, ролей, пристроїв, сесій та журналів. Така будова бази дає змогу стежити за станом Raspberry Pi, бачити активні сесії, зберігати історію дій користувачів і бути впевненим, що дані не пошкоджені.

Клієнтська частина зроблена на React і Vite. Через цей інтерфейс користувач може увійти в систему, подивитися доступні лабораторні стенди, взяти Raspberry Pi, відкрити веб-термінал і завершити сесію. Для терміналу у браузері використали xterm.js, а команди та їхні результати передаються через WebSocket.

Окремий компонент системи – це агент Raspberry Pi, написаний на Python. Він підключається до серверної частини через WebSocket, перевіряє свою ідентичність за допомогою токена пристрою, створює віртуальний термінал і потім виконує команди користувача на віддаленому Raspberry Pi. Це дозволяє студентам отримувати доступ до Raspberry Pi, не підключаючись до нього напряму через SSH.

Мета цієї кваліфікаційної роботи була досягнута. Розроблена система дозволяє централізовано керувати лабораторними стендами, віддалено працювати з Raspberry Pi, слідкувати за сесіями користувачів, фіксувати їхні дії та має зручний веб-інтерфейс. На практиці її можна використовувати як основу для проведення дистанційних лабораторних занять у навчальних закладах.

Подальший розвиток системи міг би передбачати створення системи бронювання часу, розширення адміністративної панелі, підтримку різних типів

лабораторних стендів, перехід на HTTPS/WSS у робочому середовищі та інтеграцію з навчальними платформами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Raspberry Pi Documentation. Computers. URL: https://www.raspberrypi.com/documentation/computers/ (дата звернення: 25.04.2026).

2. Raspberry Pi Documentation. Raspberry Pi OS. URL: https://www.raspberrypi.com/documentation/computers/os.html (дата звернення: 25.04.2026).

3. Raspberry Pi Documentation. Remote access. URL: https://www.raspberrypi.com/documentation/remote-access/ (дата звернення: 26.04.2026).

4. Raspberry Pi Documentation. SSH. URL: https://www.raspberrypi.com/documentation/remote-access/ssh/ (дата звернення: 25.04.2026).

5. Raspberry Pi Documentation. Raspberry Pi Connect. URL: https://www.raspberrypi.com/documentation/services/connect.html (дата звернення: 26.04.2026).

6. Raspberry Pi Documentation. Configuration. URL: https://www.raspberrypi.com/documentation/computers/configuration.html (дата звернення: 28.04.2026).

7. Raspberry Pi Documentation. GPIO and the 40-pin header. URL: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html (дата звернення: 28.04.2026).

8. Fette I., Melnikov A. The WebSocket Protocol : RFC 6455. Internet Engineering Task Force, 2011. URL: https://www.rfc-editor.org/rfc/rfc6455 (дата звернення: 11.05.2026).

9. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT) : RFC 7519. Internet Engineering Task Force, 2015. URL: https://www.rfc-editor.org/rfc/rfc7519 (дата звернення: 15.05.2026).

10. Ylonen T., Lonvick C. The Secure Shell (SSH) Protocol Architecture : RFC 4251. Internet Engineering Task Force, 2006. URL: https://www.rfc-editor.org/rfc/rfc4251 (дата звернення: 17.05.2026).

11. Node.js Documentation. Node.js v22 API. URL: https://nodejs.org/download/release/latest-v22.x/docs/api/ (дата звернення: 22.05.2026).

12. Node.js Documentation. HTTP. URL: https://nodejs.org/download/release/latest-v22.x/docs/api/http.html (дата звернення: 22.05.2026).

13. Node.js Documentation. Events. URL: https://nodejs.org/download/release/latest-v22.x/docs/api/events.html (дата звернення: 22.05.2026).

14. Node.js Documentation. Process. URL: https://nodejs.org/download/release/latest-v22.x/docs/api/process.html (дата звернення: 23.05.2026).

15. Express.js. Express routing. URL: https://expressjs.com/en/guide/routing.html (дата звернення: 27.05.2026).

16. Express.js. Writing middleware for use in Express apps. URL: https://expressjs.com/en/guide/writing-middleware.html (дата звернення: 27.05.2026).

17. Express.js. Error handling. URL: https://expressjs.com/en/guide/error-handling.html (дата звернення: 29.05.2026).

18. Express.js. Using middleware. URL: https://expressjs.com/en/guide/using-middleware.html (дата звернення: 27.05.2026).

19. Express.js. CORS middleware. URL: https://expressjs.com/en/resources/middleware/cors.html (дата звернення: 28.05.2026).

20. ws. Simple to use, blazing fast and thoroughly tested WebSocket client and server for Node.js. URL: https://www.npmjs.com/package/ws (дата звернення: 30.05.2026).

21. React Documentation. Learn React. URL: https://react.dev/learn (дата звернення: 25.05.2026).

22. React Documentation. Installation. URL: https://react.dev/learn/installation (дата звернення: 25.05.2026).

23. React Documentation. Components and Props. URL: https://react.dev/learn/your-first-component (дата звернення: 25.05.2026).

24. React Documentation. State: A Component’s Memory. URL: https://react.dev/learn/state-a-components-memory (дата звернення: 25.05.2026).

25. Vite Documentation. Getting Started. URL: https://vite.dev/guide/ (дата звернення: 27.05.2026).

26. Vite Documentation. Configuring Vite. URL: https://vite.dev/config/ (дата звернення: 27.05.2026).

27. Vite Documentation. Env Variables and Modes. URL: https://vite.dev/guide/env-and-mode (дата звернення: 27.05.2026).

28. xterm.js Documentation. URL: https://xtermjs.org/docs/ (дата звернення: 27.05.2026).

29. xterm.js Documentation. Downloading. URL: https://xtermjs.org/docs/guides/download/ (дата звернення: 27.05.2026).

30. xterm.js Documentation. Security. URL: https://xtermjs.org/docs/guides/security/ (дата звернення: 28.05.2026).

31. PostgreSQL Documentation. The PostgreSQL Global Development Group. URL: https://www.postgresql.org/docs/ (дата звернення: 25.05.2026).

32. PostgreSQL Documentation. Data Types. URL: https://www.postgresql.org/docs/current/datatype.html (дата звернення: 21.05.2026).

33. PostgreSQL Documentation. Constraints. URL: https://www.postgresql.org/docs/current/ddl-constraints.html (дата звернення: 20.05.2026).

34. PostgreSQL Documentation. Indexes. URL: https://www.postgresql.org/docs/current/indexes.html (дата звернення: 20.05.2026).

35. PostgreSQL Documentation. Transactions. URL: https://www.postgresql.org/docs/current/tutorial-transactions.html (дата звернення: 21.05.2026).

36. PostgreSQL Documentation. Network Address Types. URL: https://www.postgresql.org/docs/current/datatype-net-types.html (дата звернення: 20.05.2026).

37. node-postgres Documentation. URL: https://node-postgres.com/ (дата звернення: 17.05.2026).

38. Docker Documentation. Docker overview. URL: https://docs.docker.com/get-started/docker-overview/ (дата звернення: 26.05.2026).

39. Docker Documentation. Dockerfile reference. URL: https://docs.docker.com/reference/dockerfile/ (дата звернення: 26.05.2026).

40. Docker Documentation. Docker Compose. URL: https://docs.docker.com/compose/ (дата звернення: 26.05.2026).

41. Docker Documentation. Compose file reference. URL: https://docs.docker.com/reference/compose-file/ (дата звернення: 24.05.2026).

42. Docker Documentation. Volumes. URL: https://docs.docker.com/engine/storage/volumes/ (дата звернення: 25.05.2026).

43. Docker Documentation. Networking overview. URL: https://docs.docker.com/engine/network/ (дата звернення: 25.05.2026).

44. Python Documentation. asyncio – Asynchronous I/O. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 20.05.2026).

45. Python Documentation. Pty – Pseudo-terminal utilities. URL: https://docs.python.org/3/library/pty.html (дата звернення: 20.05.2026).

46. Python Documentation. os – Miscellaneous operating system interfaces. URL: https://docs.python.org/3/library/os.html (дата звернення: 20.05.2026).

47. websockets Documentation. URL: https://websockets.readthedocs.io/ (дата звернення: 20.05.2026).

48. MDN Web Docs. WebSocket API. URL: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API (дата звернення: 24.05.2026).

49. MDN Web Docs. Cross-Origin Resource Sharing (CORS). URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS (дата звернення: 27.05.2026).

50. OWASP Top 10:2021. URL: https://owasp.org/Top10/2021/ (дата звернення: 25.05.2026).

51. Вікіпедія – вільна енциклопедія. URL: https://en.wikipedia.org/wiki /Client%E2%80%93server_model (дата звернення: 30.05.2026).

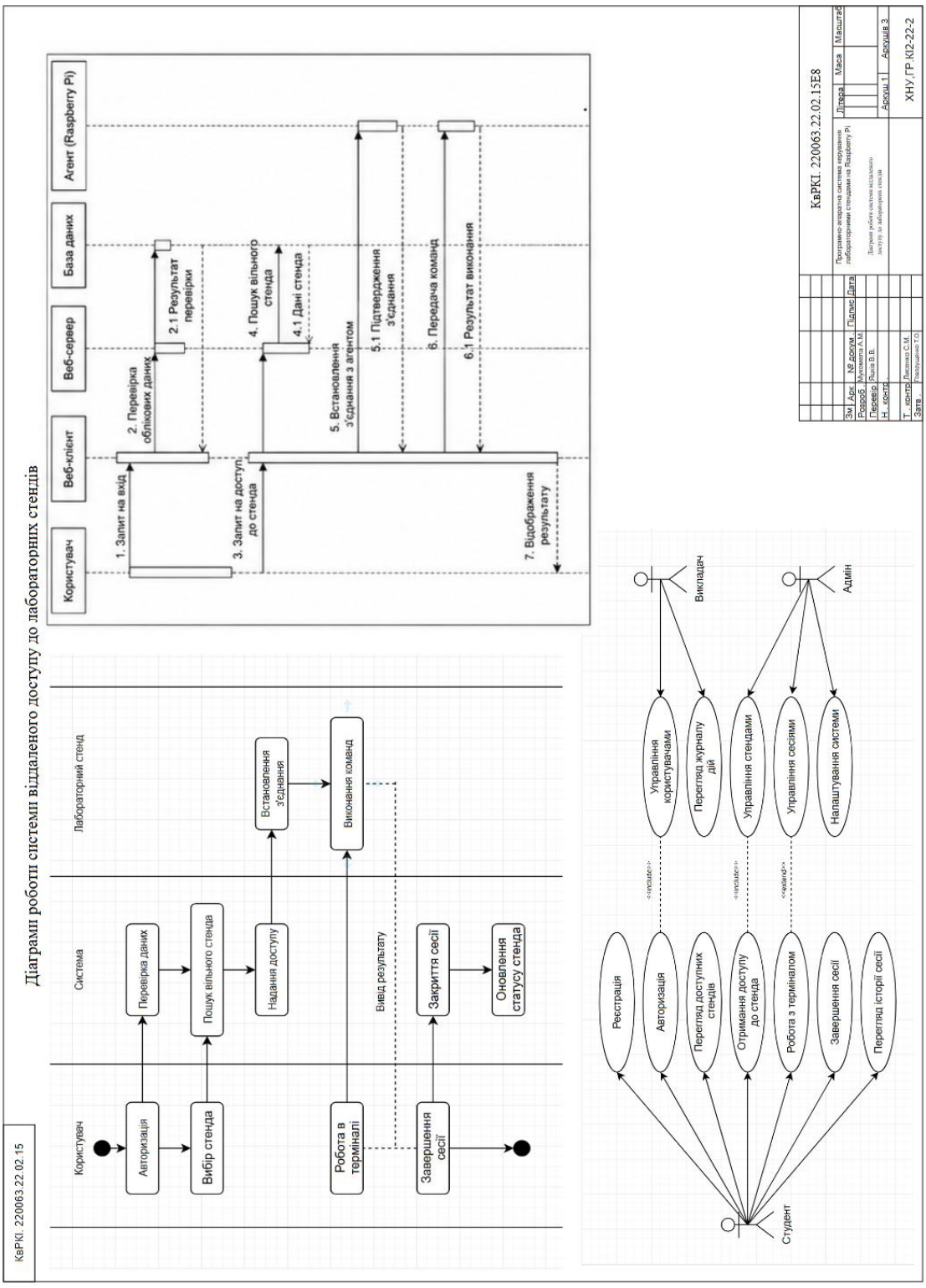
52. Мікрокомп'ютер Raspberry Pi 4 Model B 4 GB (RPI4-MODBP-2GB). Prom.ua. URL: https://picockpit.com/raspberry-pi/what-is-a-raspberry-pi-and-what-can-i-do-with-it/ (дата звернення: 30.05.2026).

53. The SSH Protocol. DEV Community. URL: https://dev.to/0xw3ston/the-ssh-protocol-1k1e (дата звернення: 30.05.2026).

ДОДАТОК А

(обов’язковий)

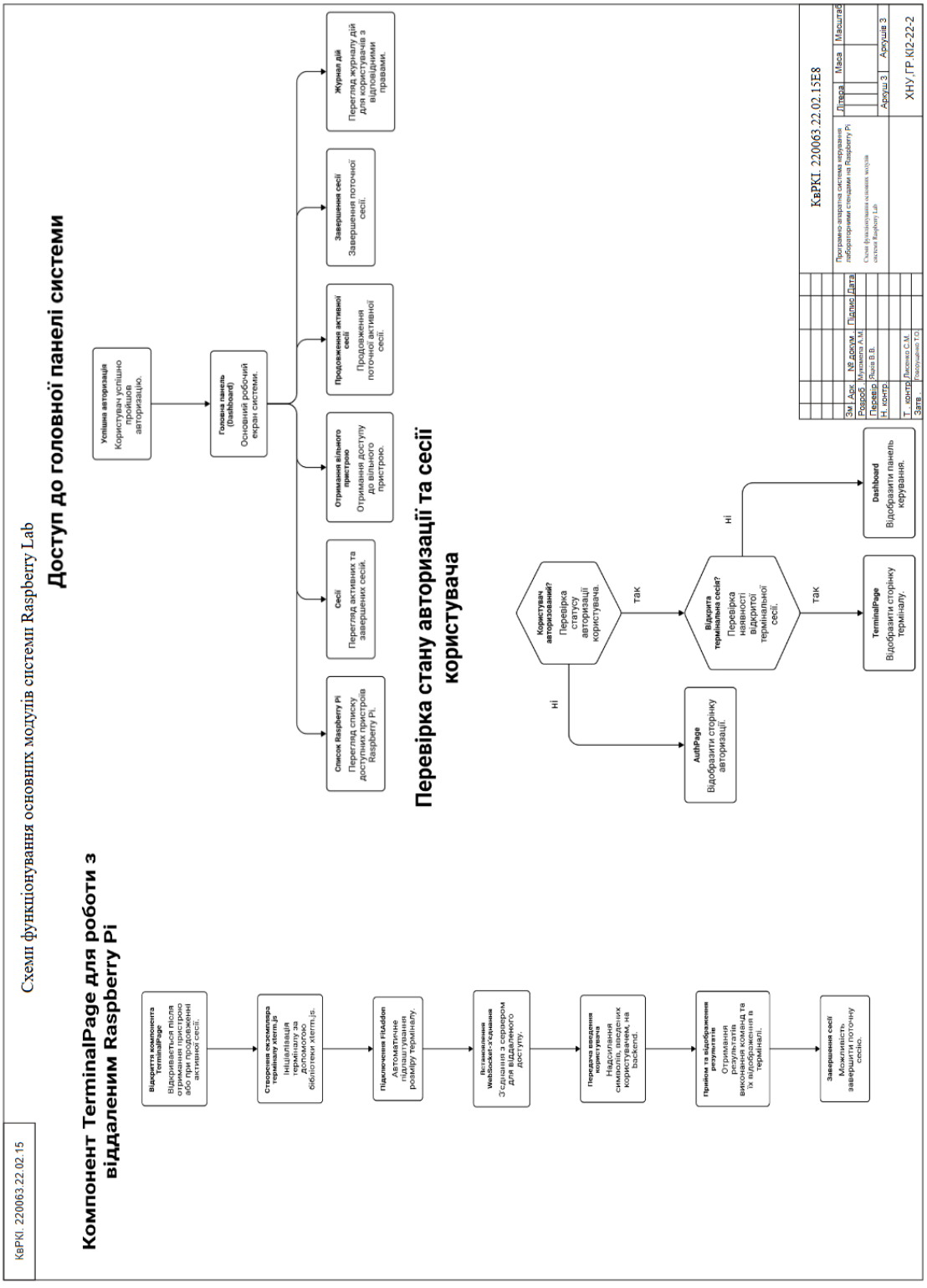
Копія креслення «Діаграми роботи системи віддаленого доступу до лабораторних стендів»



ДОДАТОК Б

(обов’язковий)

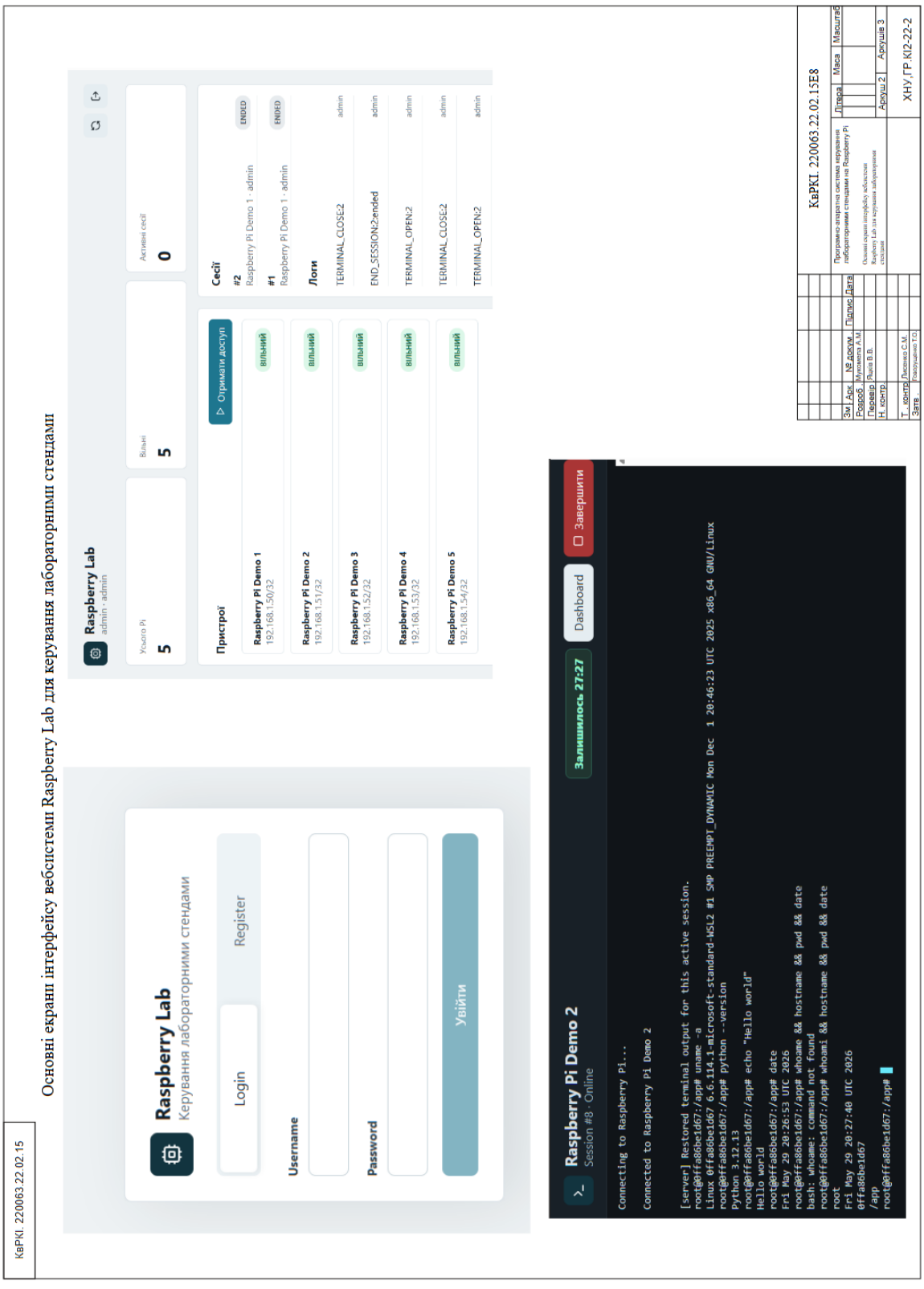
Копія креслення «Схеми функціонування основних модулів системи Raspberry Lab»



ДОДАТОК В

(обов’язковий)

Копія креслення «Основні екрани інтерфейсу вебсистеми Raspberry Lab для керування лабораторними стендами»



Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Андрій МУКОМЕЛА

Співавтор:

Назва: Програмно-апаратна система керування лабораторними стендами на Raspberry Pi

Експерт: Василь ЯЦКІВ

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:2.15%

Коефіцієнт подібності 2:0.41%

Мікропробіли: 3

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2026-06-03 17:53:34.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-06-03 Доцент Андрій Нічепорук

Дата експерт

Wed Jun 03 12:44:40 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 9%**

ID: 273304 Назва: БКР Програмно-апаратна система керування лабораторними стендами на Raspberry Pi Додано в БД: 2026-06-03 Автора: Андрій МУКОМЕЛА Керівники: Василь ЯЦКІВ Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	90592	800	2571 (3%)	36 (5%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Мукомела Андрій Миколайович

Тема: Програмно-апаратна система керування лабораторними стендами на Raspberry Pi

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___74___

1. Короткий зміст роботи та прийнятих рішень: Метою роботи є проєктування, реалізація та дослідження програмно-апаратної системи, що забезпечує віддалений доступ користувачів до лабораторних стендів на базі Raspberry Pi з використанням веб-інтерфейсу.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі кваліфікаційної роботи проведено аналіз теоретичних основ побудови систем віддаленого доступу до лабораторних стендів. Розглянуто поняття віддалених лабораторій, клієнт-серверну архітектуру, технології SSH, WebSocket, веб-терміналів, контейнеризації та засоби забезпечення безпеки. У другому розділі виконано аналіз предметної області та проєктування програмно-апаратної системи керування лабораторними стендами на базі Raspberry Pi. Визначено функціональні й нефункціональні вимоги, розроблено архітектуру системи, UML-моделі та структуру бази даних. У третьому розділі виконано практичну реалізацію системи з використанням сучасних технологій Node.js, Express.js, React, PostgreSQL, Docker, WebSocket, xterm.js та Python-агента. Реалізовано веб-портал, механізм авторизації, керування сесіями, журналювання дій, веб-термінал і взаємодію з Raspberry Pi, а також проведено тестування працездатності системи.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатньо уваги приділено тестуванню безпеки розробленої системи.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Дяченко Володимир Миколайович

доцент каф. Кібербезпеки

"12" 06 2026 р. _____(підпис)

Зав. кафедри КІІС
д-р. філософії Ользі ПАВЛОВІЙ

Мукомела Андрій Миколайович
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-22-2

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Програмно-апаратна система керування лабораторними стендами на Raspberry Pi
Автор Андрій МУКОМЕЛА
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: Василь ЯЦКІВ

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 2,15%; та системою Anti-Plagiarism складає 0,41%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

01.06.2026

Завідувач кафедри _____ Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)

Гарант освітньої програми _____ Підпис — Андрій НІЧЕПОРУК (Ім'я, ПРІЗВИЩЕ)

Керівник кваліфікаційної роботи _____ Підпис — Василь ЯЦКІВ (Ім'я, ПРІЗВИЩЕ)