Х`Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Автоматики і комп'ютеризованих технологій
(повна назва)

Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА
## Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Розроблення програми на основі хмарного середовища для дистанційного навчання студентів
(тема)

Виконав:
здобувач 2 року навчання,
групи КТРСм-23-2

Ткачов Микита Русланович

Спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Тип програми Освітньо-професійна

Освітня програма Комп'ютеризовані та робототехнічні системи

Керівник к.т.н. Сичова О.В.

Допускається до захисту
Зав. кафедри КІТАР _____ Невлюдов І.Ш.
(підпис) (прізвище, ініціали)

2025р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Автоматики і комп'ютеризованих технологій
Кафедра Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
Рівень вищої освіти другий (магістерський)
Спеціальність 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Тип програми Освітньо-професійна
Освітня програма Комп'ютеризовані та робототехнічні системи
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри КІТАР_____
(підпис)
«____» січня 2024 р.

**ЗАВДАННЯ**
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ткачову Микиті Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення програми на основі хмарного середовища для дистанційного навчання студентів
Затверджена наказом по університету від 25 листопад 2024р. № 1239 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 18.01.2025 р.

3. Вихідні дані до роботи
3.1 Сервер СКБД – PostgreSQL
3.2 Середовище розробки – MS Visual Studio
3.3 Реалізувати функцію віддаленого контролю за виконанням завдань здобувачами освіти
4. Перелік питань, що потрібно опрацювати в роботі
4.1 Аналіз предметної області
4.2 Проектування автоматизованої системи на основі хмарного середовища для дистанційного навчання студентів
4.3 Розробка архітектури бази даних
4.3 Розроблення програмного забезпечення хмарного сховища
4.4 Висновки

5. 5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Презентаційний матеріал представлений у форматі презентації PowerPoint (*.ppt) – 12 с. формату А4.

6. Консультанти розділів роботи

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

**КАЛЕНДАРНИЙ ПЛАН**

| № | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз літератури за темою кваліфікаційної роботи | 10.10.24 – 17.10.24 | Виконано |
| 2 | Аналіз предметної області | 18.10.24 – 30.10.24 | Виконано |
| 3 | Проектування автоматизованої системи на основі хмарного середовища для дистанційного навчання студентів | 01.11.24 – 16.11.24 | Виконано |
| 4 | Розробка архітектури бази даних | 17.11.24 – 10.12.24 | Виконано |
| 5 | Розроблення програмного забезпечення хмарного сховища | 11.12.24 – 20.12.24 | Виконано |
| 6 | Оформлення пояснювальної записки | 21.12.24 – 13.01.25 | Виконано |
| 7 | Подання роботи на перевірку Інтернет-сервісом Unicheck | 14.01.25 – 15.01.25 | |
| 8 | Подання роботи на рецензію | 16.01.25 – 17.01.25 | |
| 9 | Подання роботи на підпис зав. кафедри | 17.01.25 – 18.01.25 | |
| 10 | Подання кваліфікаційної роботи в ЕК | 18.01.25 | |

Дата видачі завдання 02.09.2024 р.

Здобувач _____ Ткачов Микита Русланович
(підпис)

Керівник роботи _____ к.т.н. Сичова О.В.
(підпис) (посада, прізвище, ініціали)

Я, як студент ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

«10» січня 2025 р.  Ткачов М.Р.

# РЕФЕРАТ

Пояснювальна записка: 79 с., 2 табл., 37 рис., 2 дод., 17 джерел.

ВІРТУАЛЬНИЙ МАКЕТ, ДИСТАНЦІЙНЕ НАВЧАННЯ, ХМАРНЕ СЕРЕДОВИЩЕ, ПРОТОКОЛИ IOT.

Об'єктом дослідження в даній роботі є дистанцій процес навчання здобувачів вищої освіти.

Предмет дослідження – автоматизована система колективного доступу до інформаційних ресурсів на основі хмарного сервісу.

Метою даної роботи є підвищення ефективності навчального процесу завдяки впровадженню новітніх інструментів дистанційного навчання з використанням хмарного сервісу для колективного доступу до інформаційних ресурсів.

Проведено аналіз предметної області, а саме, аналіз актуальності дослідження. Розроблення програми на основі хмарного середовища для дистанційного навчання студентів є актуальним завданням, яке відповідає сучасним викликам цифрової трансформації освіти.

Проведено аналіз протоколів Інтернету речей, описані рівні архітектури IOT для поєднання хмарного середовища з віртуальними макетами.

Розроблено хмарне середовища для дистанційного навчання студентів. Розглянуто приклад роботи програми на основі автоматичного опрацювання завдань, що пов’язані з моделюванням ситуації кібератаки на засоби автоматизації АСУТП.

Також, отримані результати роботи можна віднести до Цілі сталого розвитку 9 «Промисловість, інновації та інфраструктура», а саме п. 9.2 «Забезпечити розширення використання електротранспорту та відповідної мережі інфраструктури».

# ABSTRACT

Explanatory note: 79 p., 2 tables, 37 figures, 2 appendices, 17 sources.

VIRTUAL LAYOUT, DISTANCE LEARNING, CLOUD ENVIRONMENT, IOT PROTOCOLS.

The object of research in this paper is the distance learning process of higher education students.

The subject of research is an automated system of collective access to information resources based on a cloud service.

The purpose of this work is to increase the efficiency of the educational process through the introduction of the latest distance learning tools using a cloud service for collective access to information resources.

The subject area is analyzed, namely, the analysis of the relevance of the study. The development of a program based on a cloud environment for distance learning of students is an urgent task that meets the current challenges of digital transformation of education.

An analysis of the Internet of Things protocols is carried out, the levels of IOT architecture are described to combine the cloud environment with virtual layouts.

A cloud environment for distance learning of students is developed. An example of a program based on the automatic processing of tasks related to modeling the situation of a cyberattack on automated process control systems is considered.

Also, the results of the work can be attributed to Sustainable Development Goal 9 “Industry, Innovation and Infrastructure”, namely, clause 9.2 “Ensure the expansion of the use of electric transport and the corresponding infrastructure network”.

**ЗМІСТ**

# ПЕРЕЛІК СКОРОЧЕНЬ

АЦП – аналого-цифровий перетворювач;

БД – база даних;

ПЗ – програмний засіб;

ЦАП – цифро-аналоговий перетворювач;

AMQP – Advanced Message Queuing Protocol;

API – Application Programming Interface;

CoAP – Constrained Application Protocol;

DDS – Data Distribution Service;

DTLS – Datagram Transport Layer Security;

ETL – Extract, Transform, Load;

IOT – Internet of Things;

IIOT – Industrial Internet of Things;

MQTT – Message Queuing Telemetry Transport;

REST – Representational State Transfer;

SCADA – Supervisory Control And Data Acquisition;

QoS – Quality of service;

UDP – User Datagram Protocol;

XMPP – Extensible Messaging and Presence Protocol.

# ВСТУП

Сучасний світ активно переходить до цифрових технологій, а нещодавня пандемія суттєво прискорила процес впровадження дистанційного навчання. Водночас виникли нові виклики, пов'язані з доступністю, якістю та ефективністю навчальних процесів, що потребують інноваційних підходів.

Агресія рф в Україні суттєво вплинула на всі аспекти життя, зокрема й на освіту. В умовах воєнного стану українська освітня система зіштовхнулася з безпрецедентними викликами, що вимагають нових рішень для забезпечення безперервного навчального процесу та доступу до якісної освіти для всіх студентів.

Через воєнний стан багато студентів були змушені залишити свої домівки та переїхати до інших регіонів України або за кордон. Це призвело до серйозних перешкод у доступі до традиційних форм навчання. Хмарне середовище дозволяє студентам продовжувати навчання незалежно від їхнього місця перебування, що є критично важливим у таких умовах.

У багатьох регіонах України через активні бойові дії навчальні заклади не можуть функціонувати повноцінно. Використання дистанційних технологій дозволяє забезпечити безпеку студентів, даючи їм можливість навчатися з дому або з безпечних локацій, зберігаючи безперервність навчального процесу навіть у кризових умовах.

Чимало навчальних закладів в Україні постраждали внаслідок бомбардувань і обстрілів, що унеможливило проведення очних занять. Хмарні платформи для дистанційного навчання стають єдиною можливою альтернативою, яка дозволяє компенсувати втрату фізичної інфраструктури та надавати доступ до навчальних матеріалів і комунікаційних засобів.

Метою даної роботи є підвищення ефективності навчального процесу завдяки впровадженню новітніх інструментів дистанційного навчання з

використанням хмарного сервісу для колективного доступу до інформаційних ресурсів.

Об'єктом дослідження в даній роботі є дистанцій процес навчання здобувачів вищої освіти.

Предмет дослідження – автоматизована система колективного доступу до інформаційних ресурсів на основі хмарного сервісу.

Для досягнення мети необхідно вирішити наступні завдання:

– виконати аналіз програмних інструментів для дистанційного вивчення апаратних дисциплін;

– виконати аналіз існуючих протоколів для віддаленого доступу до хмарних сервісів з використанням технології IoT;

– подобувати архітектура автоматизованої системи для дистанційного навчання студентів на основі хмарного середовища;

– виконати побудову алгоритму та розробити програму для проведення експериментальних досліджень;

– розробити програму для проведення експериментальних досліджень;

– виконати експериментальні дослідження для підтвердження правильності теоретичних рішень.

Роботу необхідно оформити згідно нормативної документації [1 – 6].

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Аналіз актуальності дослідження

Розроблення програми на основі хмарного середовища для дистанційного навчання студентів є надзвичайно актуальним завданням у контексті агресії рф в Україні. В умовах воєнного стану українська освітня система зіштовхнулася з безпрецедентними викликами, що вимагають нових рішень для забезпечення безперервного навчального процесу та доступу до якісної освіти для всіх студентів.

Такі рішення дозволяють забезпечити безперервність навчального процесу в умовах кризи, сприяють збереженню освітньої якості та допомагають створювати нові форми доступної та безпечної освіти, яка може адаптуватися до змінних обставин.

Через воєнний стан багато студентів були змушені залишити свої домівки та переїхати до інших регіонів України або за кордон. Це призвело до серйозних перешкод у доступі до традиційних форм навчання. Хмарне середовище дозволяє студентам продовжувати навчання незалежно від їхнього місця перебування, що є критично важливим у таких умовах.

Хмарні платформи забезпечують адаптацію навчальних матеріалів, доступ до інтерактивних ресурсів, можливість індивідуального навчання, що підвищує інклюзивність навчання в кризових умовах.

Навіть після завершення активних бойових дій очікується, що гібридне навчання стане новою нормою для багатьох університетів та шкіл. Хмарні технології дозволяють інтегрувати онлайн-освіту з традиційною, що забезпечить більше можливостей для адаптації освітніх процесів до різних умов і запитів студентів.

Хмарні рішення також забезпечують можливість використання новітніх технологій, таких як штучний інтелект, машинне навчання та аналітика

великих даних, для аналізу освітніх результатів та адаптації навчальних програм до індивідуальних потреб студентів. Це допомагає підвищити якість освіти та забезпечити її відповідність сучасним вимогам [7].

Хмарні технології дозволяють створювати інтерактивні освітні платформи з можливістю доступу до матеріалів у будь-який час і з будь-якого пристрою, що значно покращує доступ до освіти. Це також дає можливість студентам працювати в групах, спілкуватися з викладачами та колегами в реальному часі або асинхронно.

Багато провідних університетів та освітніх платформ (наприклад, Coursera, edX) вже активно використовують хмарні технології для організації дистанційного навчання. Це демонструє тенденцію до глобальної цифровізації освіти. Відсутність національних та локальних рішень у цій сфері може призвести до відставання у розвитку освітніх систем, особливо в регіонах із недостатньо розвиненою цифровою інфраструктурою.

Хмарні рішення допомагають значно знизити витрати на навчання, оскільки для їх функціонування не потрібно купувати дороге програмне забезпечення або апаратні засоби. Університети можуть зосередитися на наданні якісного навчання без витрат на підтримку фізичних інфраструктур. Для студентів це також означає зниження витрат на проїзд та проживання, що підвищує доступність освіти.

Використання хмарних технологій дозволяє легко адаптувати освітні матеріали до індивідуальних потреб кожного студента. Це особливо важливо в контексті інклюзивної освіти, де кожен студент може отримати доступ до адаптованих матеріалів або додаткових ресурсів для ефективного навчання.

Таким чином, розроблення програми на основі хмарного середовища для дистанційного навчання студентів є актуальним завданням, яке відповідає сучасним викликам цифрової трансформації освіти. Така програма здатна зробити навчання доступнішим, гнучкішим і більш ефективним, підвищуючи якість освіти та створюючи нові можливості для студентів і викладачів у всьому світі.

## 1.2 Рівні архітектури IOT для поєднання хмарного середовища з віртуальними макетами

В процесі розробки засобів автоматизації, що використовуються в процесі дистанційного навчання з використанням віртуальних макетів, мережі IOT та IIOT пристроїв, необхідно враховувати особливості побудови даної мережі в концепції Інтернету речей [7 – 9]. На рис. 1.1 подані відмінності між традиційними інтернет протоколами та тими, що застосовуються в IIOT.

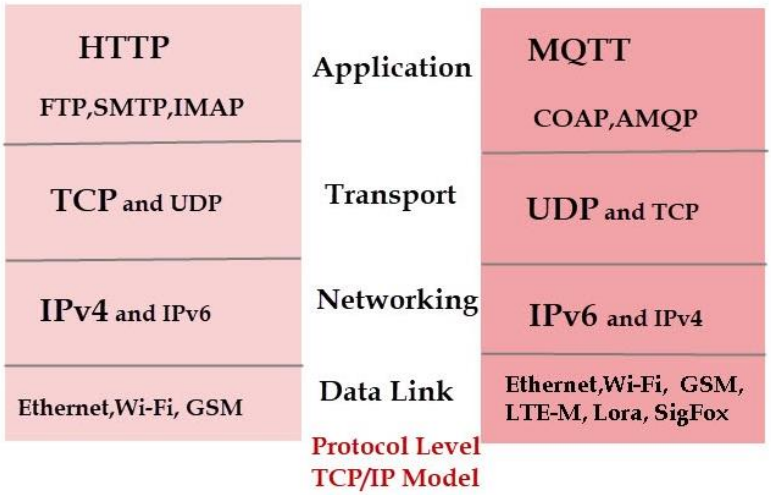


Рисунок 1.1 – Відмінності між традиційними інтернет протоколами та тими, що застосовуються в IIOT

На відміну від типової WEB-системи, що працює за принципом клієнт-серверної програми, модель IIoT-рішень набагато складніша. Аналогічно клієнт-серверному принципу в IIoT-архітектурі можна виділити дві різні за фізичним розташуванням групи обов'язкових компонентів:

– периферія (Edge) – це кінцеві smart-пристрої, розташовані на технологічному обладнанні, за яким здійснюється віддалений моніторинг та керування;

– потужні Big Data інструменти, що розгорнуті у центрі обробки даних на серверах чи в хмарі (Backend).

Тим не менш, через особливості використання IIoT-рішень, пов’язаних з умовами експлуатації та прикладної специфіки, наприклад, екстремальні температури, вібрації, опади, велика довжина каналів передачі даних тощо, в архітектурі Industrial Internet of Things можна виділити цілих 12 рівнів (шарів) (рис. 1.2).

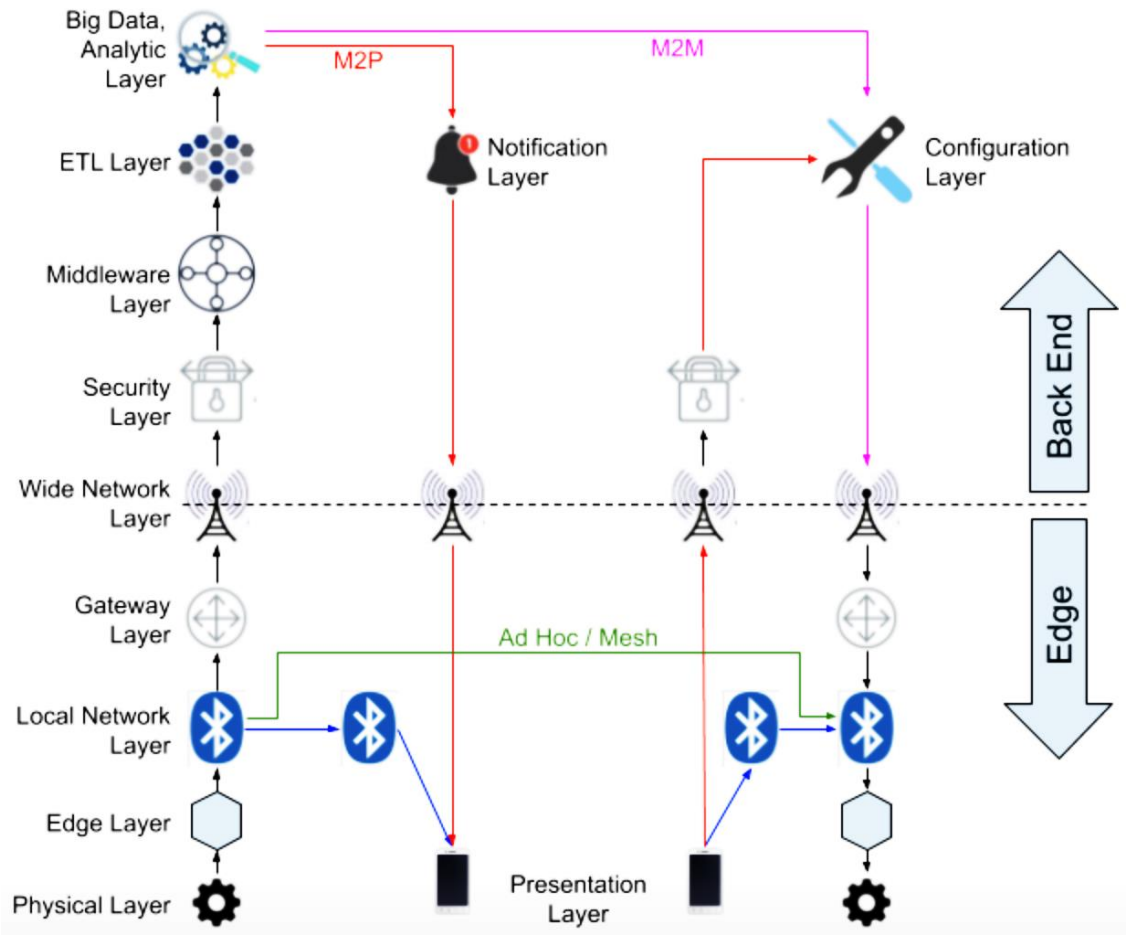


Рисунок 1.2 – Рівні архітектури IIOT

На фізичному рівні (Physical Layer) розташовані кінцеві пристрої – датчики та сенсори, камери та інше обладнання (таймери, п’єзоелементи, мікрофони, фотодіоди/транзистори/резистори, вимикачі, акселерометри, GPS-трекери, баркод-рідери тощо), що збирає інформацію та надсилає її далі, на рівень периферійної обробки даних (Edge Layer).

Периферійні обчислення (Edge Layer) забезпечують мінімальну обробку даних, зокрема перетворення між аналоговим та цифровим поданням інформації (АЦП/ЦАП). Як правило, ці функції реалізуються за допомогою мікроконтролерів Raspberry Pi, Arduino та інші однокристальні системи SoC-модулі (System-on-a-Chip) – електронні схеми, що виконує функції цілого пристрою, в т. ч. бездротові мережі на кристалі (Wireless network-on-chip, WNOC).

Пристрої фізичного та периферійного рівнів Інтернету речей (IoT) виконують важливу функцію у збиранні даних та взаємодії з фізичним середовищем, тому до них висуваються певні вимоги [10]. Одна з ключових вимог – це енергетична ефективність. Оскільки багато таких пристроїв працюють на батареях або інших автономних джерелах живлення, важливо, щоб вони мали низьке енергоспоживання для збільшення часу роботи без підзарядки або заміни батарей. Крім того, важливим є використання енергоефективних протоколів зв'язку, які дозволяють знизити витрати енергії під час передачі даних.

Друга важлива вимога – це надійність зв'язку. Пристрої IoT повинні підтримувати стабільну і безперебійну передачу даних, навіть у складних умовах, таких як великі відстані або наявність перешкод. Це вимагає підтримки надійних бездротових технологій, таких як Wi-Fi, Bluetooth, Zigbee або LoRa, в залежності від конкретного застосування.

Третя вимога стосується безпеки. Пристрої IoT повинні забезпечувати захист даних від несанкціонованого доступу та атак. Це включає в себе шифрування переданих даних, автентифікацію пристроїв і захист від фізичного втручання.

Ще одна важлива вимога – масштабованість. Система IoT повинна підтримувати можливість підключення великої кількості пристроїв без втрати продуктивності. Це вимагає від пристроїв підтримки сучасних протоколів, які дозволяють їм працювати в умовах високої щільності з'єднань.

Також до пристроїв IoT висуваються вимоги щодо здатності до роботи в реальному часі, оскільки багато додатків потребують негайного реагування на події. Крім того, необхідно враховувати стійкість пристроїв до екстремальних умов навколишнього середовища, таких як температурні коливання, волога, пил тощо.

Рівень периферійної комунікації (Local Network Layer) в структурі Інтернету речей виконує важливу функцію передачі даних між пристроями фізичного рівня, такими як сенсори та виконавчі пристрої, і шлюзами або локальними серверами (рис. 1.3). Цей рівень забезпечує зв'язок у межах локальної мережі, дозволяючи збирати дані, передавати їх та, у деяких випадках, попередньо обробляти перед тим, як відправити їх на хмарні або віддалені обчислювальні системи [11].

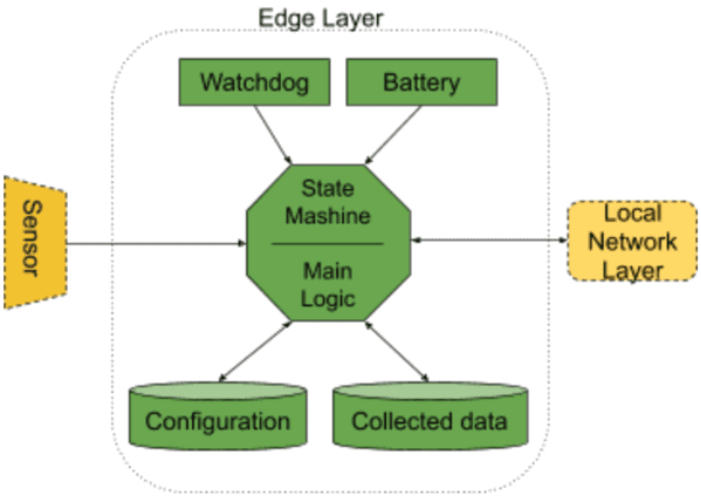


Рисунок 1.3 – Схема периферійного IIoT-пристрою

Основною функцією цього рівня є передача даних від сенсорів до контролерів або шлюзів. Дані можуть бути передані в сирому вигляді або після попередньої обробки, залежно від складності та вимог системи. При цьому на периферійному рівні можуть використовуватися різні протоколи передачі даних, такі як Wi-Fi, Bluetooth, Zigbee, LoRa або Ethernet, які забезпечують

різні рівні швидкості, відстані передачі та енергоспоживання, залежно від конкретного завдання.

Периферійний рівень також включає в себе функції маршрутизації, тобто передачу даних між різними пристроями всередині мережі. Це дозволяє організувати ефективний обмін інформацією між сенсорами і виконавчими пристроями або централізованими вузлами управління. Крім того, у багатьох випадках шлюзи або контролери можуть виконувати попередню обробку даних. Це дозволяє зменшити обсяг інформації, яка передається на вищі рівні системи або в хмару, а також дозволяє приймати рішення на місці у режимі реального часу.

Ще однією важливою вимогою до цього рівня є забезпечення безпеки. Оскільки периферійні пристрої можуть взаємодіяти з великою кількістю інших пристроїв і передавати чутливі дані, важливо захищати передачу даних за допомогою шифрування, автентифікації пристроїв і протоколів безпеки, які запобігають несанкціонованому доступу.

Периферійний рівень також повинен бути сумісним з різними протоколами та підтримувати масштабованість системи, тобто здатність обробляти велику кількість пристроїв в одній локальній мережі. Це важливо, оскільки зростання кількості пристроїв IoT вимагає гнучкої системи, здатної до ефективної інтеграції нових компонентів.

Шлюз (Gateway Layer), який виконує ETL-операції, є компонентом архітектури даних, що відповідає за отримання, трансформацію та завантаження даних між різними системами чи базами даних [12]. Цей шар є посередником між джерелами даних і цільовими системами або сховищами даних.

На рисунку 1.4 показано приклад застосування шлюзу в хмарних середовищах IoT.

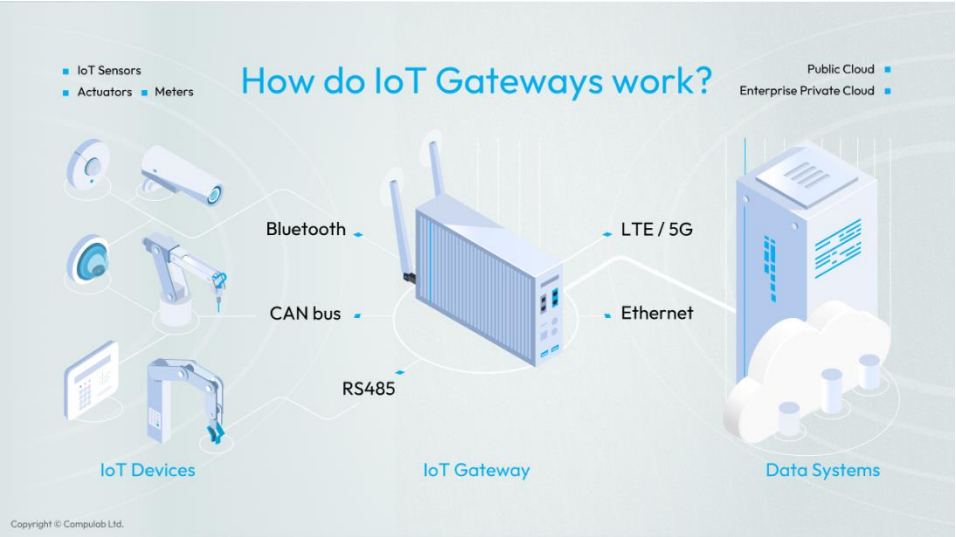


Рисунок 1.4 – Приклад застосування шлюзу в хмарних середовищах IoT

ETL (Extract, Transform, Load) – це процес, що складається з трьох основних етапів. Перший етап – Extract (Отримання) – передбачає витягування даних з різних джерел (бази даних, файли, API тощо). На цьому етапі важливо забезпечити правильне з'єднання з джерелами даних і коректне збирання необхідної інформації. Другий етап – Transform (Трансформація) – включає в себе перетворення даних у потрібний формат для подальшого використання. Це може включати очищення даних, їх агрегацію, нормалізацію або обчислення нових значень, що відповідають бізнес-правилам. Третій етап – Load (Завантаження) – полягає у завантаженні трансформованих даних у цільову систему, таку як сховище даних, аналітичну платформу або іншу базу даних.

Шлюз, що виконує ці операції, зазвичай автоматизує процес обробки та передачі даних. Він може включати в себе механізми для планування завдань (наприклад, за допомогою cron або спеціальних ETL-інструментів), моніторингу процесів, а також обробки помилок, щоб забезпечити стабільність і точність обміну даними.

Важливими аспектами шлюзу є продуктивність, масштабованість та безпека, оскільки він має справу з великими обсягами даних, що часто мають конфіденційний характер. Таким чином, для забезпечення ефективності та надійності операцій у складі шлюзу зазвичай використовуються методи оптимізації та забезпечення доступу до даних з високою доступністю.

Шлюз також може підтримувати різноманітні формати даних і протоколи, що дозволяє інтегрувати різноманітні системи та джерела даних, що можуть бути використані для різних аналітичних чи операційних задач.

Рівень представлення (Presentation Layer) (рис. 1.2) застосовується коли оброблена інформація від периферійних пристроїв презентується кінцевому користувачеві за допомогою UI/UX-інтерфейсів на екрані комп'ютерного монітора, планшета або мобільного телефону. Рівень подання також відповідає за обслуговування, конфігурацію та зміни стану кожного компонента IIoT-системи, дозволяючи віддалено керувати навіть периферійними пристроями. Поки не існує стандартизованих UI/UX-засобів для рівня представлення IIoT-системи, незважаючи на наявність міжнародних та національних стандартів, які регламентують поняття людино-машинного інтерфейсу, що широко використовуються в SCADA-системах.

Рівень конфігурацій (Configuration Layer) забезпечує сховище статусів периферійних пристроїв (актуальний стан, новий стан, який буде завантажено та проміжний статус, що вказує на процес оновлення стану). Це необхідно через особливості зв'язку периферії та шлюзу з Backend-сервером, тому що комунікація кінцевих пристроїв з хмарою тримається не завжди, а періодичними сеансами, під час яких відбувається відправлення та отримання даних.

## 1.3 Основні протоколи організації зв’язку в IOT

Серед багатьох протоколів, що застосовуються для організації обміну повідомленнями в IOT можна виділити декілька тих, що застосовуються найчастіше [8, 13]. До них відносяться:

– DDS OMG;

– AMQP;

– MQTT;

– REST;

– CoAP;

– XMPP.

Data Distribution Service (DDS) – це стандарт для розподіленої комунікації в реальному часі, розроблений для забезпечення ефективної передачі даних між розподіленими додатками або пристроями в складних та високонавантажених системах [14] (рис. 1.5). DDS забезпечує механізми публікації і підписки на дані, дозволяючи зменшити залежність між компонентами системи і забезпечити високу продуктивність при обміні даними.

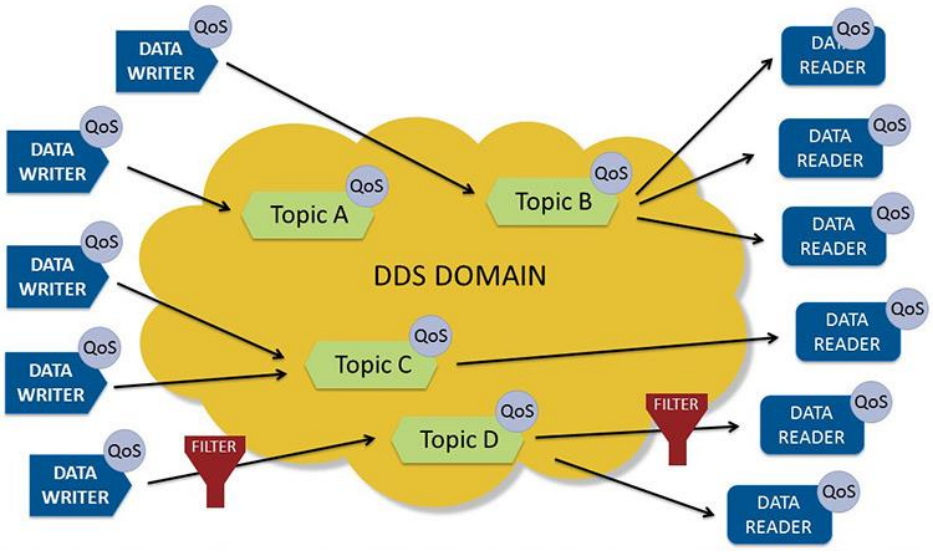


Рисунок 1.5 – Контрольований обмін даними за допомогою протоколу DDS

Основні особливості DDS включають підтримку розподілених архітектур, де дані можуть передаватися між численними пристроями або програмами в реальному часі з низькими затримками. Це робить DDS ідеальним для використання в додатках, які потребують високої надійності і оперативності передачі даних, таких як автоматизація, оборонні технології, автомобільна електроніка та Інтернет речей (IoT).

DDS функціонує через модель публікації та підписки. Публікатори публікують дані, які можуть бути отримані підписниками. Дані передаються через спеціально визначені Topics (Теми), які служать для класифікації та організації даних відповідно до їх типів і змісту. Важливу роль у DDS відіграють параметри Quality of Service (QoS), які дозволяють налаштовувати різні аспекти взаємодії, такі як затримка, надійність, швидкість доставки даних і інші параметри. Завдяки цим можливостям DDS забезпечує ефективну та надійну комунікацію навіть в умовах складних і динамічних мереж.

Однією з ключових переваг DDS є підтримка як однонаправленої комунікації (від одного до багатьох), так і двосторонньої комунікації (багато до багатьох), що дозволяє гнучко налаштовувати систему залежно від потреб. DDS також забезпечує високий рівень узгодженості і масштабованості, що дозволяє зберігати ефективність обміну даними навіть в великих розподілених системах.

AMQP (Advanced Message Queuing Protocol) – це відкритий стандарт протоколу повідомлень, який призначений для забезпечення надійної, асинхронної комунікації між різними додатками та сервісами в розподілених системах. AMQP підтримує орієнтовану на повідомлення архітектуру обміну даними, де обробка повідомлень відбувається через черги, що дозволяє забезпечити ефективну комунікацію між компонентами навіть у випадку, коли вони працюють в різних середовищах або в умовах непостійного зв'язку.

Основні характеристики AMQP включають високу надійність, підтримку складних маршрутизаційних схем і гарантії доставки повідомлень. Протокол дозволяє застосовувати черги повідомлень, що дає змогу додаткам

асинхронно обмінюватися даними, зберігаючи їх цілісність і надійно передаючи їх від відправника до отримувача. AMQP підтримує черги, обробку повідомлень за допомогою фільтрації, маршрутизації, а також різні механізми підтвердження отримання повідомлень, що забезпечує точність і безпеку доставки.

Спрямування даних, отриманих з бірж, і розміщення їх у чергах здійснюється виробником (рис. 1.6).

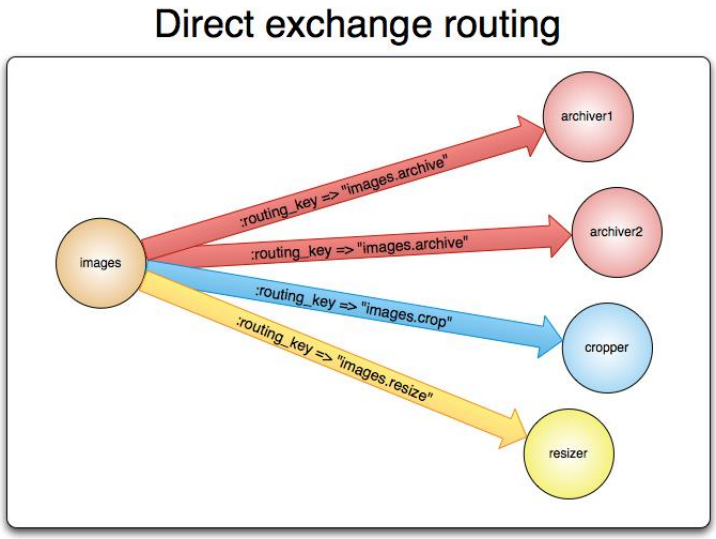


Рисунок 1.6 – Принцип взаємодії компонентів за протоколом AMQP

AMQP працює за принципом клієнт-сервер, де відправник (публікатор) і отримувач (підписник) взаємодіють через посередника – сервер повідомлень, який забезпечує зберігання і передачу повідомлень між клієнтами [15]. Сервер обробляє черги, визначає правила маршрутизації і управляє доставкою повідомлень. Завдяки такій архітектурі AMQP дозволяє реалізовувати надійний обмін повідомленнями навіть в умовах високих навантажень і при виникненні неполадок у мережі.

AMQP зазвичай використовують для інтеграції різних систем, зокрема в розподілених обчислювальних середовищах, таких як мікросервісні архітектури, де необхідно забезпечити ефективну та безпечну передачу даних

між різними сервісами. Він також широко застосовується в системах, де важлива гарантія доставки і збереження повідомлень, таких як фінансові платформи, системи обробки транзакцій або платформи для обміну даними в реальному часі.

MQTT (Message Queuing Telemetry Transport) – це легкий протокол передачі повідомлень, який розроблений для забезпечення ефективної, надійної та асинхронної комунікації в розподілених системах, особливо в умовах обмежених ресурсів, таких як пристрої Інтернету речей [16]. MQTT використовує модель публікації та підписки, де пристрої (клієнти) можуть публікувати повідомлення на певні теми або підписуватися на теми для отримання повідомлень. Цей протокол дозволяє зменшити обсяг переданих даних і за рахунок цього підвищити ефективність передачі навіть при низьких швидкостях з'єднання або нестабільному зв'язку.

Принцип взаємодії між пристроями за протоколом MQTT показано на рисунку 1.7.

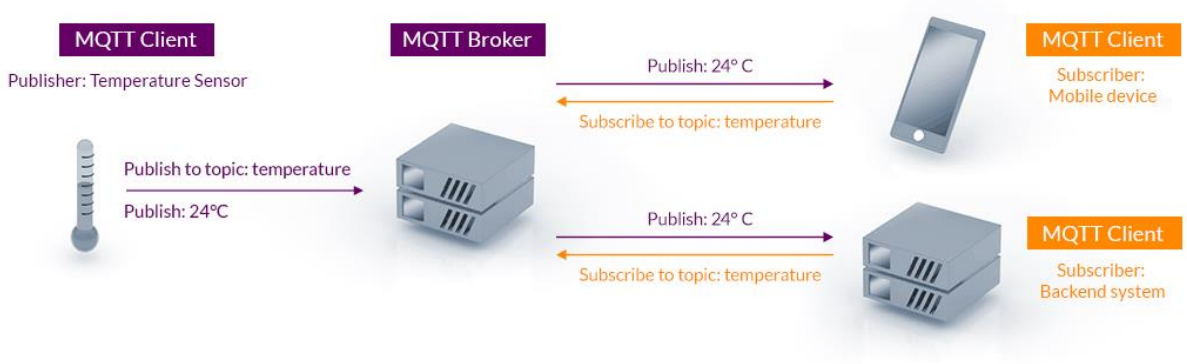


Рисунок 1.7 – Принцип взаємодії між пристроями за протоколом MQTT

Основні особливості MQTT включають малий розмір повідомлень, простоту реалізації і можливість роботи в умовах обмеженого пропускного каналу або високих затримок. Завдяки цим характеристикам MQTT широко використовується в системах IoT, де пристрої часто працюють з малими

обсягами даних і підключаються через нестабільні або обмежені мережі, такі як мобільні мережі або Wi-Fi.

Протокол працює за принципом клієнт-сервер. Клієнти (пристрої або додатки) підключаються до центрального брокера повідомлень, який забезпечує маршрутизацію повідомлень між публікаторами і підписниками. Публікатори надсилають повідомлення на конкретні теми, а підписники отримують ці повідомлення, підписавшись на відповідні теми. Брокер відіграє роль посередника, який керує з'єднаннями та доставкою повідомлень, а також гарантує надійність доставки.

MQTT підтримує кілька рівнів якості обслуговування (QoS), що дозволяє налаштовувати механізми доставки повідомлень:

– QoS 0 – повідомлення доставляються один раз, без підтвердження;

– QoS 1 – повідомлення доставляються принаймні один раз, з підтвердженням отримання;

– QoS 2 – повідомлення доставляються точно один раз, з підтвердженням отримання і додатковим механізмом контролю.

Це дає змогу вибирати між високою продуктивністю і надійністю залежно від вимог до конкретної системи або додатка.

MQTT також підтримує механізм збережених повідомлень (retained messages), що дозволяє зберігати останнє повідомлення для кожної теми на брокері. Це забезпечує, що нові підписники на тему одразу отримають останнє доступне повідомлення, навіть якщо вони підключаються після його публікації.

Завдяки своїй простоті, надійності та низькому споживанню ресурсів MQTT є популярним вибором для використання в системах IoT, домашніх автоматизаціях, а також в промислових додатках, де потрібно забезпечити ефективну передачу даних між великою кількістю пристроїв у реальному часі.

REST – мігрував в IOT як модель проєктування Web API. Архітектура REST умовно складається з клієнтів і серверів. Клієнти ініціюють запити до серверів, сервери обробляють запити та повертають відповідні відповіді.

Запити та відповіді будуються навколо передачі представлених ресурсів (рис. 1.8).

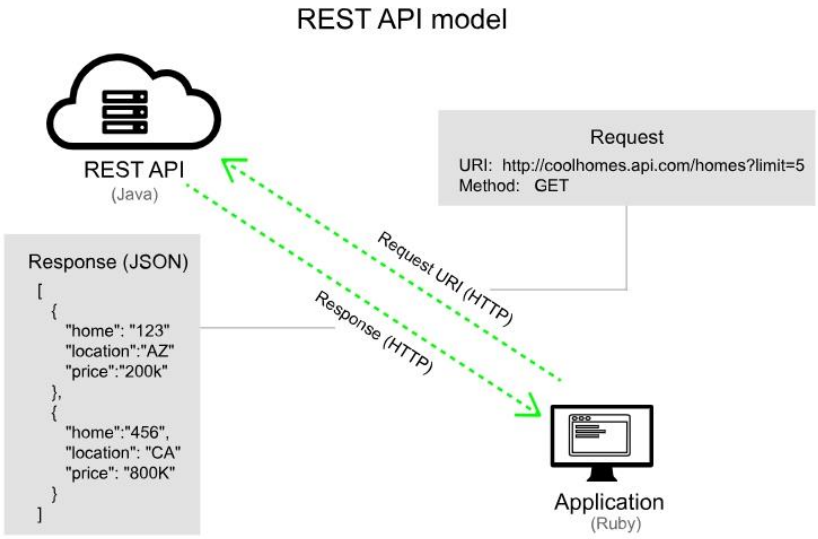


Рисунок 1.8 – Принцип обміну повідомлення за протоколом REST

API, що використовують протокол HTTP для передачі запитів та відповідей, розглядаються як веб-сервіси. У разі веб-сервісів клієнт, який запитує на ресурс, і сервер API, що надає відповідь, можуть використовувати будь-яку мову програмування або платформу. Не має значення, яка мова програмування, або платформа будуть використовуватися, тому що запит повідомлення та відповідь зроблено через загальний веб-протокол HTTP.

Веб-протокол є частиною веб-сервісів: вони незалежні від мови і тому сумісні між різними платформами та системами. При документуванні REST API не має значення, чи будують інженери API за допомогою Java, Ruby, Python або іншої мови. Запити виконуються через HTTP і відповіді повертаються через HTTP.

Constrained Application Protocol (CoAP) – це протокол передачі даних, розроблений для обміну повідомленнями між пристроями в умовах обмежених ресурсів, таких як пристрої Інтернету речей. CoAP є легким протоколом, який спеціально орієнтований на мережі з низькою пропускною

здатністю, високими затримками та обмеженими обчислювальними ресурсами, що робить його ідеальним для використання в таких сценаріях, як розумні міста, промислова автоматизація або домашні пристрої, де потрібна комунікація між великою кількістю маленьких пристроїв.

Архітектура CoAP складається з декількох компонентів (рис. 2.9):

– клієнт (Client) – пристрій, який ініціює запити до сервера;

– сервер (Server) – пристрій, який надає відповіді на запити від клієнта;

– Proxy – пристрій, який здійснює пересилання запитів від клієнта до сервера або від сервера до клієнта;

– Resource Directory – директорія ресурсів, яка зберігає інформацію про доступні ресурси на серверах.

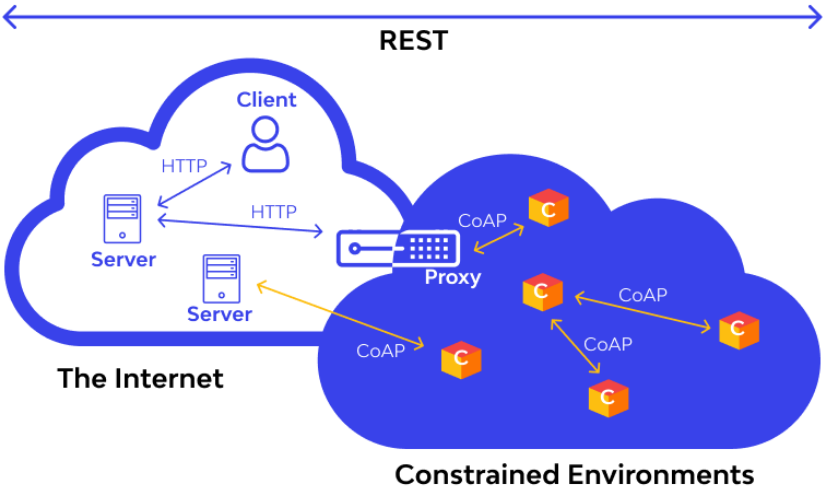


Рисунок 1.9 – Архітектура CoAP

CoAP працює на основі принципу клієнт-сервер і використовує принцип ресурсно-орієнтованої архітектури. Він дозволяє пристроям запитувати або передавати ресурси через HTTP-подібні методи (GET, POST, PUT, DELETE), але з більш легким та ефективним підходом. Для зменшення навантаження і збереження енергоспоживання CoAP працює з компактними заголовками повідомлень і підтримує асинхронний режим обміну даними, що дозволяє зменшити кількість необхідних з'єднань та оптимізувати передачу даних.

Однією з основних особливостей CoAP є використання UDP (User Datagram Protocol) замість TCP, що зменшує затримки і витрати на встановлення з'єднання. Завдяки цьому CoAP особливо підходить для використання в реальному часі, де важлива швидка доставка повідомлень, навіть в умовах нестабільних мереж або мереж з низькою пропускною здатністю. Проте для забезпечення надійності, CoAP включає механізми підтвердження доставки повідомлень, а також можливість повторної передачі повідомлень у разі втрати даних.

CoAP також підтримує multicast (мультиканальний трансляційний режим), що дозволяє одному пристрою одночасно відправляти повідомлення кільком пристроям, що значно підвищує ефективність комунікації в великих мережах пристроїв.

Протокол також передбачає безпеку через використання DTLS (Datagram Transport Layer Security), що забезпечує шифрування та автентифікацію даних під час передачі, запобігаючи їх модифікації або несанкціонованому доступу.

CoAP активно використовується в таких сферах, як розумні будинки, системи моніторингу здоров'я, енергетичні мережі та автоматизація, де важлива комунікація між великою кількістю пристроїв з обмеженими ресурсами.

XMPP (Extensible Messaging and Presence Protocol) – це відкритий протокол для обміну повідомленнями та присутності в реальному часі, який дозволяє різним користувачам і системам взаємодіяти через текстові повідомлення, сповіщення про статус і присутність. XMPP був розроблений як масштабований, розширюваний протокол для миттєвих повідомлень і використовується в численних додатках, таких як чати, голосові та відео дзвінки, а також в реальному часі для обміну даними в різних розподілених системах.

Архітектура XMPP ґрунтується на розподіленій архітектурі клієнт-сервер (Client-Server Architecture), що дозволяє розробляти масштабовані системи з безліччю користувачів та серверів (рис. 1.10).

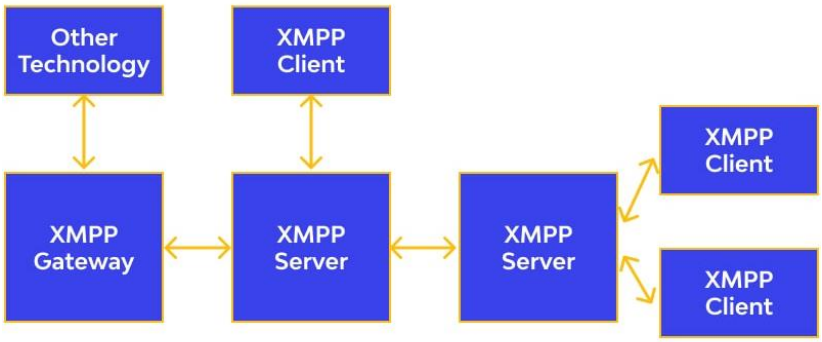


Рисунок 1.10 – Архітектура XMPP

XMPP використовує модель клієнт-сервер, де користувачі підключаються до серверів, які обробляють їх запити та доставляють повідомлення іншим користувачам. Протокол забезпечує високу гнучкість завдяки своїй розширюваності, що дозволяє додавати нові функціональності без змін в основному протоколі. Однією з головних переваг XMPP є підтримка присутності (presence), що дозволяє користувачам бачити, хто онлайн, а хто офлайн, а також отримувати сповіщення про зміни в статусі контактів.

Протокол базується на XML (Extensible Markup Language), що робить його надзвичайно гнучким і дозволяє передавати різноманітні типи даних, включаючи текстові повідомлення, файли, сповіщення про стан і метадані. XMPP дозволяє реалізовувати різні механізми маршрутизації повідомлень і розширювати базові функції через додаткові розширення (XMPP Extensions, XEP), що дозволяє користувачам реалізовувати складнішу логіку для чатів, групових повідомлень, мультимедійного обміну та інших сервісів.

XMPP дозволяє додавати нові функції за допомогою стандартних розширень, що забезпечує підтримку різних типів контенту і взаємодії.

Протокол підтримує шифрування даних через TLS (Transport Layer Security), що забезпечує конфіденційність і цілісність переданих повідомлень. Окрім цього, існують механізми автентифікації користувачів і серверів.

XMPP підтримує одночасне обслуговування мільйонів користувачів, забезпечуючи можливість створення великих розподілених мереж.

Протокол підтримує передачу даних про стан (онлайн/офлайн) і зміни статусу користувачів, що є важливим елементом для сучасних чат-систем.

XMPP також підтримує реальний обмін медіа-даними (аудіо, відео), а також дозволяє інтегрувати додаткові сервіси, як-от конференції, гри або інші інтерактивні функції.

XMPP знайшов широке застосування в різноманітних сферах, включаючи мобільні додатки для чатів, корпоративні комунікаційні платформи, а також в якості бази для створення інтерактивних сервісів в Інтернеті речей (IoT). Він є основою багатьох популярних месенджерів, таких як Jabber, Google Talk (до його закриття), а також використовуються для інтеграції різних сервісів у реальному часі.

## 1.4 Особливості та принцип використання протоколу MQTT для доступу до хмарного середовища

Комунікаційна стратегія публікації/передплати (pub/sub) MQTT, спрямована на максимізацію використання пропускної здатності, є заміною традиційної споживчої архітектури, яка безпосередньо взаємодіє з кінцевою точкою. Однак у парадигмі pub/sub клієнт, який передає новини (видавець), відокремлений від клієнтів, які отримують інформацію (або передплатників). Оскільки ані автори, ані клієнти не спілкуються один з одним одразу, їхньою взаємодією в них керують треті сторони, які називаються брокерами (рис. 1.11).

MQTT надає спосіб створення ієрархії каналів зв'язку – свого роду гілка з листям. Щоразу, коли видавець має нові дані для поширення серед клієнтів, повідомлення супроводжується приміткою контролю доставки. Клієнти вищого рівня можуть отримувати кожне повідомлення, тоді як клієнти нижчого рівня можуть отримувати повідомлення, які стосуються лише одному чи двом базовим каналам, «відгалуженим» у нижній частині ієрархії. Це полегшує обмін інформацією розміром від двох байт до 256 Мб.

Основні риси протоколу MQTT:

– обмін повідомленнями відбувається за принципом видавець-передплатник (Pub-Sub);

– розмір заголовка повідомлення становить 2 байти, а корисне навантаження може змінюватись від 1 байта до 256 Мб;

– у протоколі закладено можливість вибору одного із трьох рівнів обслуговування.

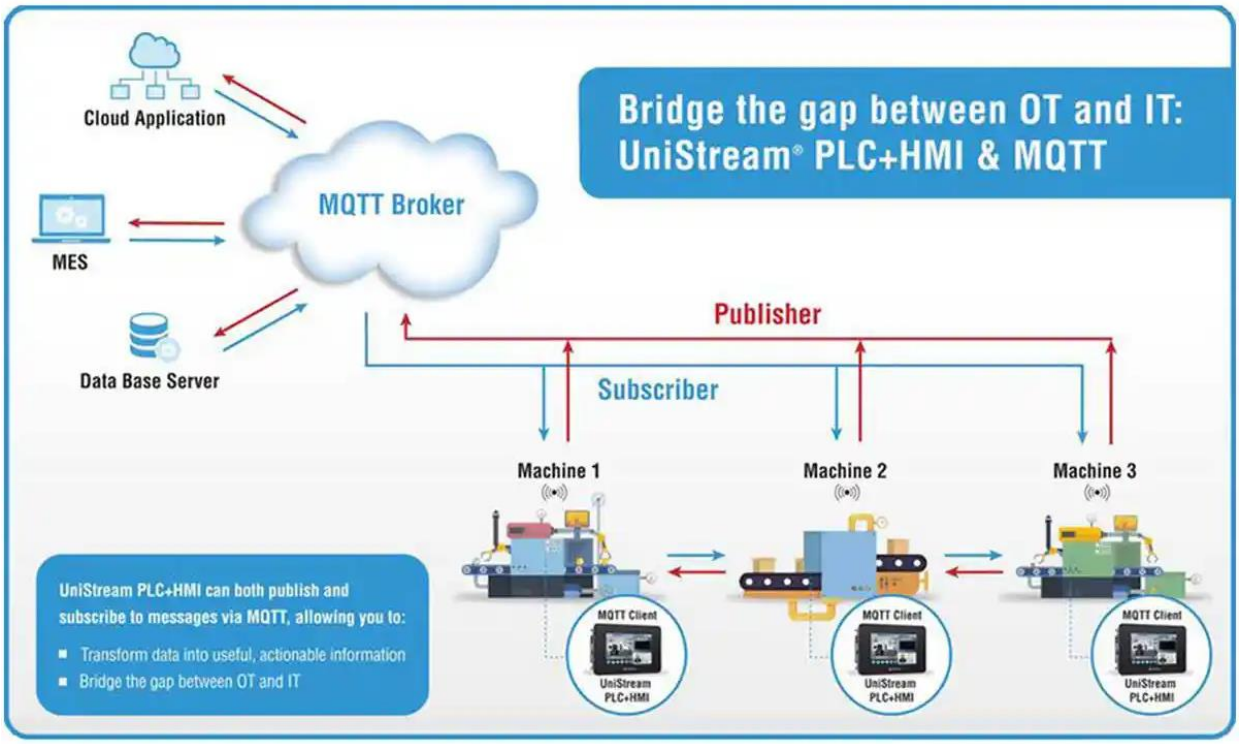


Рисунок 1.11 – MQTT, архітектура «видавець-передплатник»

Відмінною особливістю принципу «видавець-передплатник» від клієнт-серверного підходу є те, що клієнти, які надсилають повідомлення (видавці, Publisher), та клієнти, які приймають повідомлення (підписники, Subscriber), як правило, розділені. Розподіл може бути організований у трьох площинах:

– простір – видавець і передплатник нічого не повинні знати один про одного;

– час – видавець і передплатник не повинні бути увімкнені в один і той самий час;

– синхронізація – операції на обох сторонах не повинні припинятися протягом публікації чи отримання інформації.

Видавець та передплатник не передають одне одному повідомлення безпосередньо, не встановлюють прямий контакт, можуть не знати про існування одне одного. Координує та керує передачею повідомлень від видавця до передплатника та від передплатника до видавця брокер (Broker).

Паралельне виконання операцій на брокері є другою важливою особливістю принципу взаємодії «видавець-передплатник».

MQTT-клієнт – це, зазвичай, пристрій, оснащений мікроконтролером, який підтримує стек TCP/IP. Клієнтські бібліотеки MQTT доступні для багатьох мов програмування, наприклад Android, Arduino, C, C++, C#, Go, iOS, Java, JavaScript, NET.

Брокер є основним елементом системи «видавець-передплатник». Він відповідає за прийом усіх повідомлень, їх фільтрацію, ухвалення рішення про те, кому цікаві ці повідомлення, і, зрештою, за пересилання повідомлень усім клієнтам-передплатникам.

Серед серверних реалізацій брокера можна назвати:

– IBM WebSphere MQ;

– відкрите програмне забезпечення Mosquitto;

– рішення, що базується на хмарному сервісі Eurotech Everywhere Device Cloud;

– легко масштабований та високопродуктивний відкритий сервер emqttd, остання версія (0,17) дозволяє обслуговувати 1,3 мільйони з’єднань;

– брокер HiveMQ, що забезпечує корпоративну безпеку та максимальну масштабованість.

Спрощений процес обміну інформацією можна описати так:

– видавець передає повідомлення з даними (наприклад, інформація з датчиків температури) на брокер, вказуючи при цьому тему (Topic), до якої ці дані належать (наприклад, «Temp»);

– брокер аналізує, які з передплатників мають передплату на певні теми, в даному випадку – на тему «Temp»;

– передплатникам, які підписані на тему «Temp», брокером буде надіслано повідомлення з інформацією від датчиків температури.

Таким чином, безліч передплатників може бути підписано на різноманітні теми та залежно від цих передплат отримувати необхідну їм інформацію, не спілкуючись безпосередньо з видавцем.

## 1.5 Висновки по першому розділу

В результаті виконання першого розділу кваліфікаційної роботи проведено аналіз предметної області, а саме, аналіз актуальності дослідження. Розроблення програми на основі хмарного середовища для дистанційного навчання студентів є актуальним завданням, яке відповідає сучасним викликам цифрової трансформації освіти.

Проведено аналіз протоколів Інтернету речей, описані рівні архітектури IOT для поєднання хмарного середовища з віртуальними макетами. Розглянуті особливості побудови даної мережі в концепції Інтернету речей, враховуючи процеси розробки засобів автоматизації навчання здобувачів освіти, що використовуються в процесі дистанційного навчання з використанням віртуальних макетів.

# 2 ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ НА ОСНОВІ ХМАРНОГО СЕРЕДОВИЩА ДЛЯ ДИСТАНЦІЙНОГО НАВЧАННЯ СТУДЕНТІВ

## 2.1 Розробка архітектури автоматизованої системи

Загальна архітектура автоматизованої системи на основі хмарного середовища для дистанційного навчання студентів показана на рисунку 2.1.

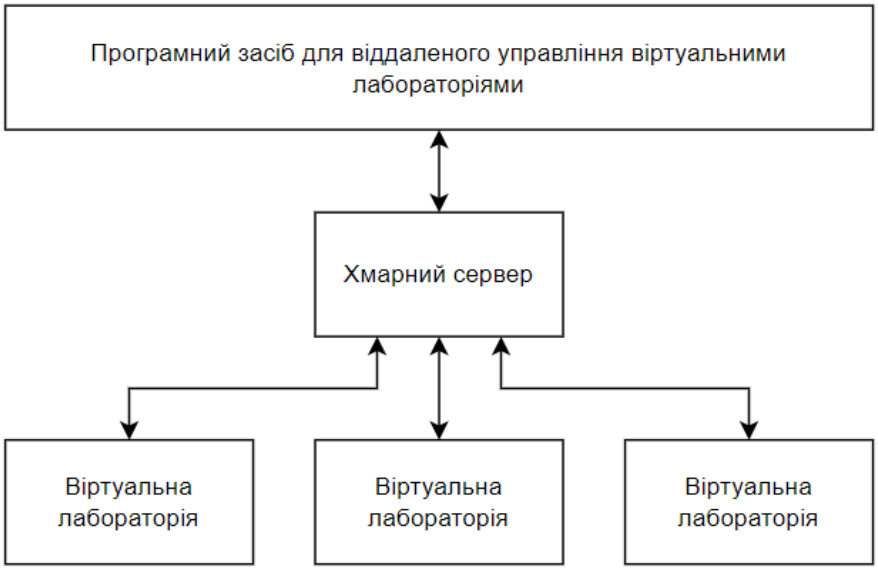


Рисунок 2.1 – Загальна архітектура автоматизованої системи

Архітектура хмарного середовища для дистанційного навчання студентів складається з трьох основних компонентів:

– програмного засобу для віддаленого управління віртуальними лабораторіями;

– хмарного сервера;

– віртуальної лабораторії.

Зазначені компоненти взаємодіють між собою для забезпечення ефективної організації навчального процесу, що дозволяє студентам виконувати лабораторні завдання дистанційно, використовуючи ресурси хмарних технологій.

Хмарний сервер є центральним елементом цієї архітектури. Він відповідає за зберігання і обробку даних, забезпечує управління всіма підключеними компонентами та дозволяє користувачам взаємодіяти з системою в режимі реального часу. На хмарному сервері функціонують кілька програмних модулів, включаючи сервер управління базою даних (наприклад, PostgreSQL), що зберігає інформацію про користувачів, лабораторні завдання, результати та інші дані. Також на сервері встановлено програмний модуль моніторингу процесів, який забезпечує стабільність роботи системи, відстежує її продуктивність і контролює активність користувачів та виконання завдань.

Програмний засіб для віддаленого управління віртуальними лабораторіями відіграє важливу роль в управлінні освітнім процесом. Він дозволяє адміністраторам і викладачам створювати і налаштовувати лабораторні завдання для студентів, а також керувати користувачами і відслідковувати їхню діяльність. До складу цього засобу входить кілька модулів.

Програмний модуль управління користувачами, який відповідає за автентифікацію, реєстрацію користувачів та управління їхніми правами доступу. Цей засіб дозволяє проводити контроль за виконанням завдань студентами, управляти ресурсами віртуальних лабораторій та обробляти дані, що надходять від користувачів.

Віртуальна лабораторія є інтегрованим середовищем, яке імітує реальні лабораторні установки або процеси і дозволяє студентам виконувати експерименти в дистанційному режимі. Віртуальна лабораторія містить цифрові двійники лабораторних пристроїв, які взаємодіють з програмним модулем збору даних і програмними інструментами. Через хмарний сервер студенти можуть виконувати різноманітні завдання, отримувати доступ до

лабораторного обладнання та аналізувати результати в реальному часі. Програмний модуль управління віртуальною лабораторією контролює всі процеси, що відбуваються в лабораторії, і передає дані до хмарного сервера для подальшої обробки.

Структура віртуальної лабораторії показана на рисунку 2.2.

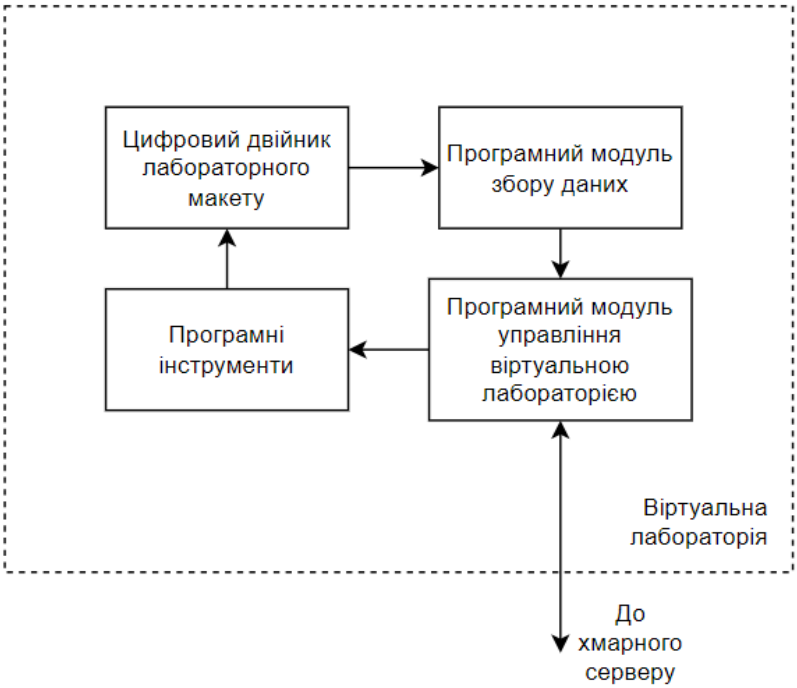


Рисунок 2.2 – Структура віртуальної лабораторії

Дана структурна схема віртуальної лабораторії для дистанційного навчання студентів ілюструє взаємодію між компонентами, які забезпечують функціонування лабораторії на основі хмарного середовища.

Цифровий двійник лабораторного макету представляє віртуальну модель реального лабораторного обладнання або процесу. Ця модель взаємодіє з програмним модулем збору даних і програмними інструментами.

Програмний модуль збирає дані з цифрового двійника лабораторного макету та передає їх до програмного модуля управління віртуальною лабораторією. Також він може отримувати дані від хмарного сервера для обробки.

Програмний модуль управління віртуальною лабораторіє керує всіма процесами, пов'язаними з віртуальною лабораторією, включаючи взаємодію з користувачами, управління даними і підключення до хмарного сервера. Він отримує дані від модуля збору даних і програмних інструментів.

Програмні інструменти – це засоби, які використовуються для обробки або модифікації цифрового двійника. Вони також надають можливість управляти процесами у віртуальній лабораторії та аналізувати результати.

Відповідно до загальної архітектури віртуальна лабораторія підключається до хмарного сервера для зберігання і обробки даних, що дозволяє забезпечити дистанційне навчання студентів. Це також дозволяє користувачам отримувати доступ до лабораторії з різних пристроїв.

Основна ідея даної схеми полягає в інтеграції хмарних технологій для забезпечення безперебійного доступу до навчальних лабораторій, що дозволяє студентам взаємодіяти з цифровими двійниками реальних лабораторних процесів і даними через віртуальне середовище.

Структура хмарного сервера показана на рисунку 2.3. Дана схема ілюструє архітектуру хмарного серверу, який використовується для управління віртуальною операційною системою, базою даних PostgreSQL і моніторингом процесів управління.

Віртуальна операційна система – це програмне середовище, що виконує роль операційної системи на віртуальному сервері. Віртуальна ОС взаємодіє з клієнтами хмарного серверу, забезпечуючи їм доступ до різних ресурсів і послуг серверу. Також вона забезпечує з'єднання з іншими компонентами, такими як сервер управління базою даних і програмний модуль моніторингу процесів управління.

Сервер управління базою даних PostgreSQL: серверна частина, яка відповідає за зберігання та управління даними в системі побудована на основі СКБД PostgreSQL.

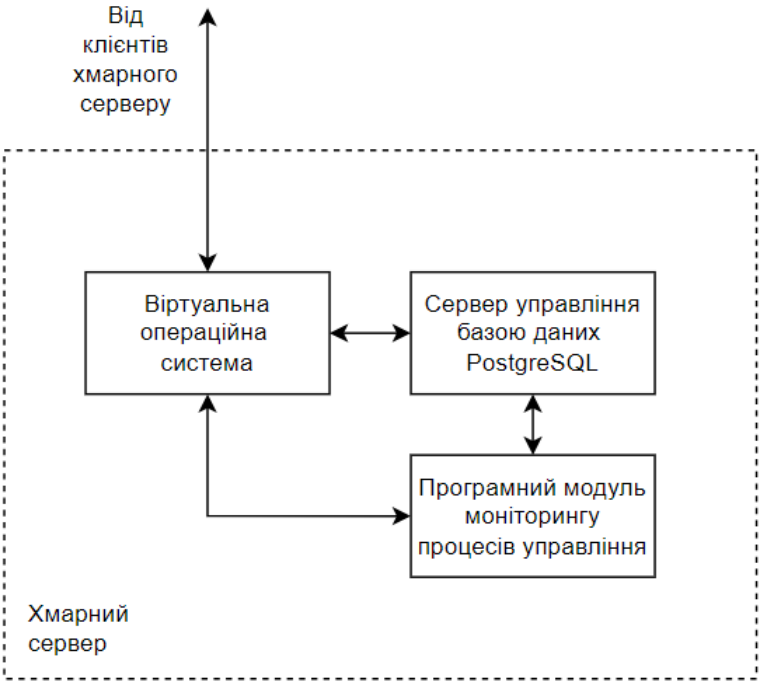


Рисунок 2.3 – Структура хмарного сервера

PostgreSQL є потужною реляційною базою даних з підтримкою складних запитів і масштабованою архітектурою. Він взаємодіє з віртуальною операційною системою, обробляючи запити на зберігання або отримання даних, що поступають від клієнтів, а також забезпечує зв’язок з програмним модулем моніторингу процесів управління.

Програмний модуль моніторингу процесів управління відповідає за моніторинг і контроль усіх процесів, що відбуваються на сервері. Він збирає інформацію про стан системи, працездатність сервісів, навантаження на сервер, та інші ключові параметри. Взаємодіє з сервером бази даних PostgreSQL для зберігання і обробки даних, а також передає інформацію віртуальній операційній системі, яка потім може бути передана клієнтам або використана для оптимізації роботи сервера.

Клієнтами хмарного серверу є користувачі або інші програми, які підключаються до хмарного серверу, щоб отримати доступ до його ресурсів. Віртуальна операційна система є головним інтерфейсом для взаємодії клієнтів

із сервером. Клієнти можуть здійснювати запити на отримання даних, виконувати обчислювальні задачі або інші операції через віртуальне середовище.

У даній схемі показана інтеграція між віртуальною операційною системою, сервером бази даних і моніторингом процесів. Основна мета такої архітектури полягає в забезпеченні стабільного управління ресурсами, безперервного моніторингу стану системи та швидкого реагування на запити клієнтів. PostgreSQL виступає інструментом для зберігання і керування великими обсягами даних, а програмний модуль моніторингу забезпечує стабільність роботи серверу, вчасно інформуючи про будь-які проблеми або потреби в оптимізації.

Завдяки хмарним технологіям, така система забезпечує доступ до ресурсів і обробку даних у режимі реального часу з будь-якого місця, що робить її важливим інструментом для дистанційного навчання, великих підприємств або інших організацій, що потребують гнучкого та масштабованого управління даними та обчислювальними ресурсами.

Структура програмного засобу для управління віртуальними лабораторіями показана на рисунку 2.4.

Дана схема відображає структуру управління віртуальними лабораторіями, яка базується на взаємодії кількох програмних модулів. Мета цієї архітектури – забезпечити ефективну роботу з хмарним сервером та керувати користувачами й задачами віртуальних лабораторій.

Програмний модуль підтримки з'єднання з хмарним сервером відповідає за встановлення і підтримку зв'язку між системою віртуальних лабораторій та хмарним сервером. Він забезпечує безперебійну передачу даних до і від хмари, дозволяючи користувачам виконувати лабораторні роботи в режимі реального часу. Цей модуль є ключовим для взаємодії всіх компонентів системи з хмарним середовищем.

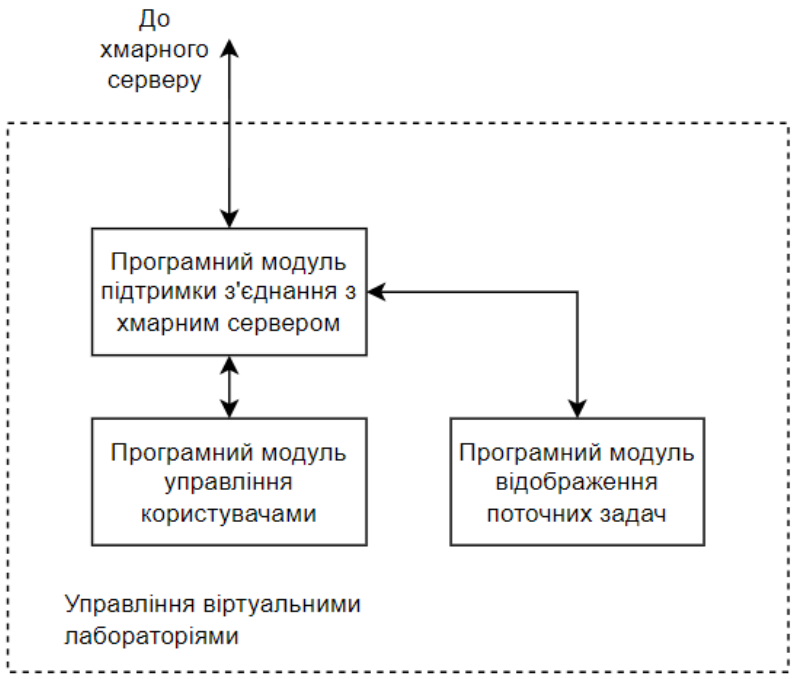


Рисунок 2.4 – Структура програмного засобу для управління віртуальними лабораторіями

Програмний модуль управління користувачами відповідає за обробку інформації про користувачів системи. Він дозволяє керувати правами доступу, реєстрацією, автентифікацією і підтримкою сесій користувачів. Також модуль може фіксувати активність користувачів, надавати доступ до необхідних ресурсів лабораторії та слідкувати за їхніми діями під час виконання завдань.

Програмний модуль відображення поточних задач призначений для надання користувачам інформації про їхні поточні задачі. Він відображає прогрес лабораторних робіт, поточні активності, можливі помилки та повідомлення системи. Завдяки цьому модулю користувачі можуть вчасно отримувати зворотний зв'язок щодо виконання завдань, що підвищує ефективність навчального процесу.

Взаємодія між модулями є критично важливою для роботи системи управління віртуальними лабораторіями. Модуль управління користувачами працює у тісній співпраці з модулем підтримки з'єднання з хмарним сервером,

щоб забезпечити правильне керування доступом до ресурсів і завдань. Програмний модуль відображення поточних задач також використовує дані, отримані через хмарне з'єднання, щоб інформувати користувачів про їхні дії.

Таким чином, дана система забезпечує безперервну роботу віртуальних лабораторій, надаючи користувачам можливість взаємодіяти з лабораторними інструментами та виконувати завдання віддалено через хмарне середовище. Це рішення робить процес навчання більш гнучким та доступним для широкого кола користувачів, забезпечуючи ефективний обмін інформацією та ресурсами через хмарний сервер.

## 2.2 Розробка алгоритму роботи програми

Перша частина алгоритму роботи програмного модуля управління віртуальною лабораторією показано на рисунку 2.5.

На початку роботи програми виконується функція, що генерує унікальний ідентифікатор апаратного забезпечення на якому відбувся її запуск. Результатом генерації є UserKey який однозначно описує властивості операційної системи та апаратури.

Наступним кроком є формування запиту до серверу бази даних, що розташовано в хмарному середовищу. В запиті вказується сформований UserKey за яким шукається користувач системи. Кожен користувач при реєстрації передає на сервер дану інформацію і вона додається до унікального запису в якості одно з атрибутів, що характеризує персону.

Якщо запис в базі даних не знайдено, запускається процес реєстрації нового користувача. Для цього формується відповідне вікно в яке нова персона вводить свої дані (прізвище, ім’я, по батькові, електронну адресу). При зберіганні нових даних UserKey автоматично додається до атрибутів користувача.

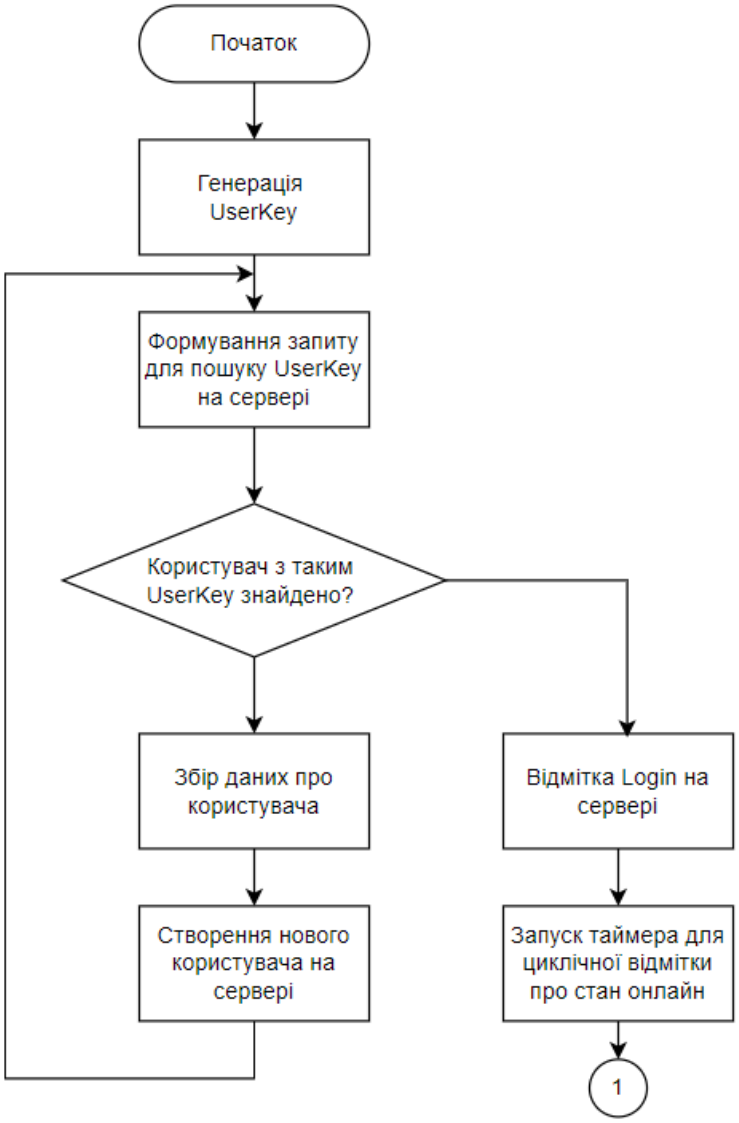


Рисунок 2.5 – Перша частина алгоритму роботи програмного модуля управління віртуальною лабораторією

Після завершення реєстрації процедура перевірки наявності UserKey в базі даних повторюється.

Якщо відповідний запис знайдено, то програма переходить до основного режиму роботи. На початку роботи в даному режимі програма формує запит на сервер з командою Update, яка встановлює атрибут Login в стан «true». Це означає, що користувач знаходиться в активному режимі.

За алгоритмом роботи серверу, атрибут Login скидається до стану «false» кожну хвилину. Алгоритм роботи серверу в режимі опитування записів користувачів показано на рисунку 2.6.

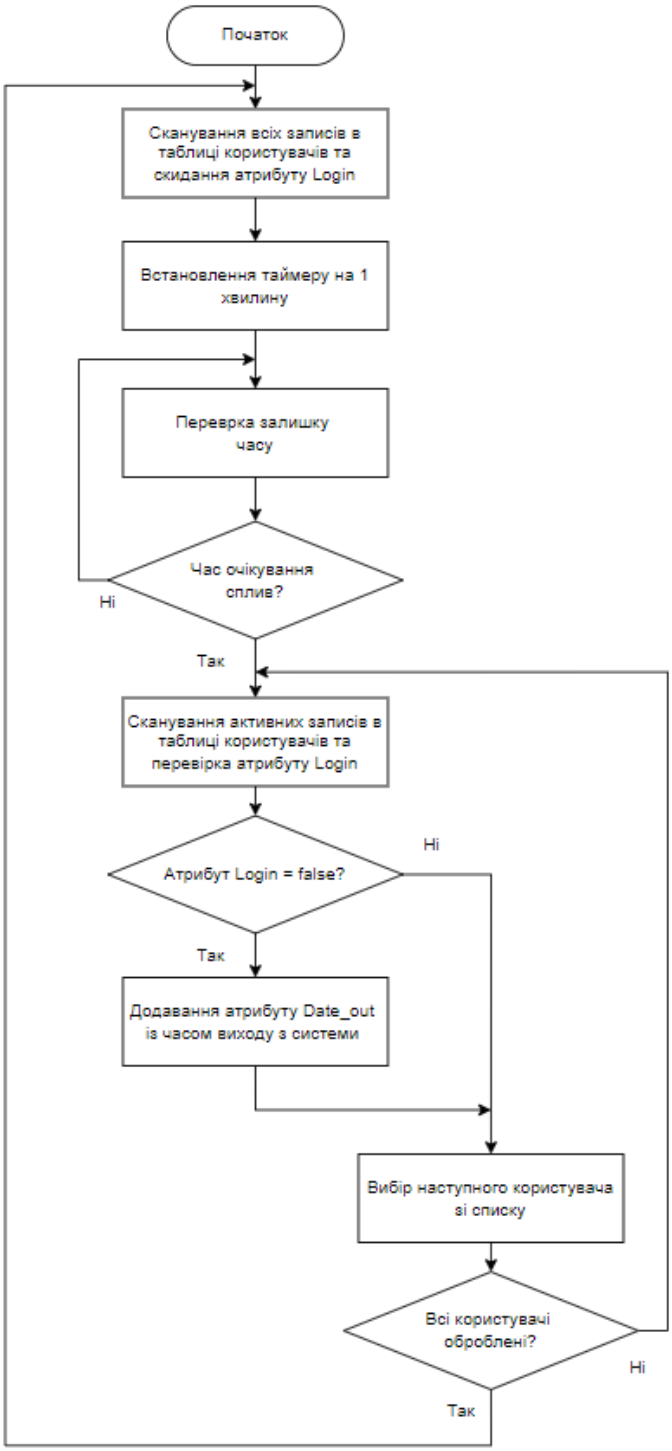


Рисунок 2.6 – Алгоритм роботи серверу в режимі опитування записів користувачів

В даному режимі відбувається сканування всіх записів в таблиці користувачів кожну хвилину та скидання атрибуту Login. Якщо при скануванні виявиться, що атрибут Login = «false» то це означає, що користувач не в мережі і програма змінює атрибут Date_out. В даний атрибут записується поточний час, що є часом фіксації виходу користувача із системи.

Цикл повторюється для всіх записів, де поле Date_out присутнє, але порожне, або має значення «0».

Для постійного оновлення стану користувача програма з його бобку запускає свій таймер та встановлює час оновлення на 30 секунд. Даний алгоритм показаний на рисунку 2.7.


Рисунок 2.7 – Алгоритм роботи програмного модуля управління віртуальною лабораторією в режимі оновлення стану користувача

Після спливу 30 секунд формується запит, що оновлює атрибут Login до стану «true».

На рисунку 2.8 показано другу частину алгоритму роботи програмного модуля управління віртуальною лабораторією.

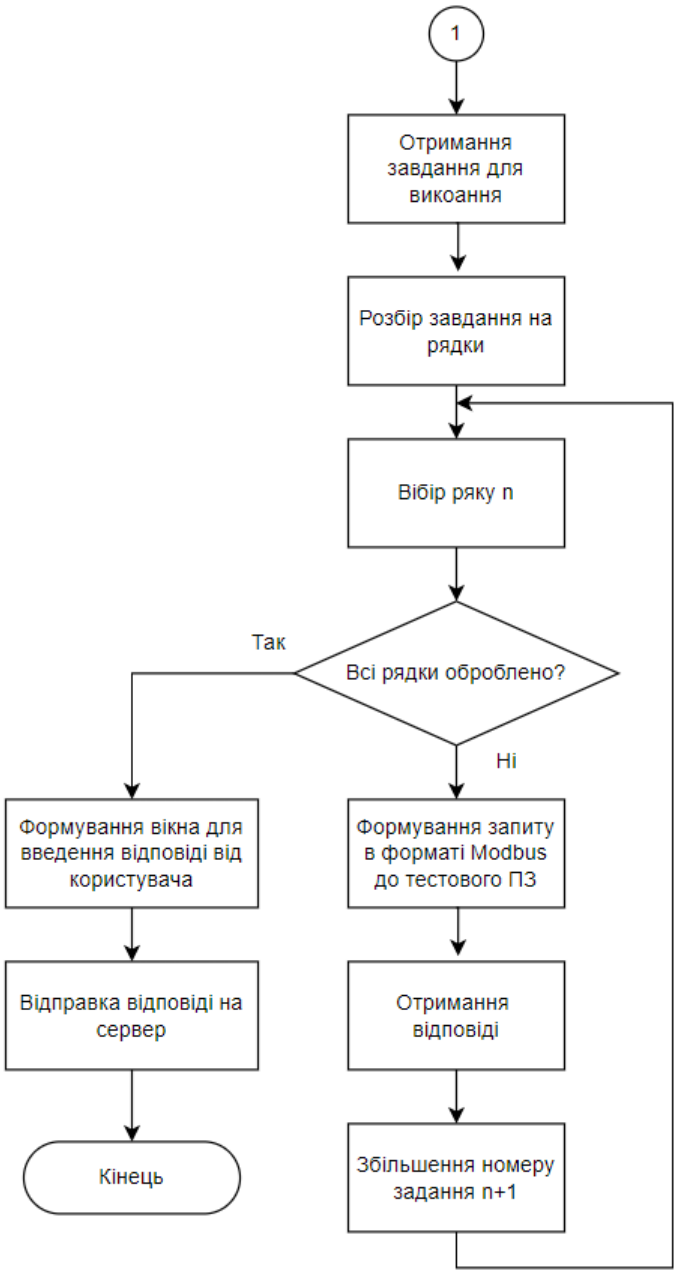


Рисунок 2.8 – Друга частина алгоритму роботи програмного модуля управління віртуальною лабораторією

В даному алгоритмі описується процес автоматичного запуску тестових запитів (завдань, що сформовані викладачем). Задача користувача –

опрацювати тестові завдання, сформувати відповідь та надіслати її на перевірку. Розглянемо принцип роботи програми на прикладі автоматичного опрацювання завдань, що пов’язані з моделюванням ситуації кібератаки на засоби автоматизації АСУТП.

На початку даного режиму роботи програма формує запит до БД та отримує список завдань для виконання. В завданнях вказуються команди для впливу на механізми віртуального макету. Завдання користувача – виявити втручання в роботу технічного засобу автоматизації та розпізнати всі команди, що були надіслані від особи тестового кіберзлочинця.

Загальний список завдань розбивається на окремі рядки та додається в чергу на виконання. На кожній ітерації обирається черговий рядок, формується запит в форматі протоколу Modbus та надсилається на керований віртуальний макет.

Пакети надсилаються через версію протоколу Modbus TCP/IP. Всі пакети в мережі перехоплюються та аналізуються за допомогою спеціального програмного інструменту WireShark. За допомогою даного програмного засобу користувач аналізує відповідні пакети та формує відповідь про виявлені атаки. Відповідь записується в спеціальне вікно програми та надсилається до хмарного серверу для перевірки та оцінювання викладачем.

## 2.3 Висновки по другому розділу

В результаті виконання даного розділу кваліфікаційної роботи побудована архітектура автоматизованої системи. Розроблені структурні схеми віртуальної лабораторії, хмарного серверу, програми управління віртуальними лабораторіями.

Розроблені необхідні алгоритми роботи програмних компонентів автоматизованої системи та описано принцип її роботи.

Загалом, архітектура хмарного середовища для дистанційного навчання студентів забезпечує гнучкість і масштабованість навчального процесу. Використання хмарних серверів дозволяє зберігати і обробляти великі обсяги даних, надавати студентам віддалений доступ до віртуальних лабораторій і дозволяє викладачам ефективно керувати навчальним процесом. Це середовище не тільки дозволяє студентам взаємодіяти з цифровими моделями реальних лабораторних процесів, але і робить можливим навчання незалежно від місця перебування користувача, що значно підвищує доступність і якість освіти.

Розроблюваний програмний засіб дозволяє автоматизувати процес навчання та зробити його прозорим та об’єктивним для оцінювання.

# 3 РОЗРОБКА АРХІТЕКТУРИ БАЗИ ДАНИХ

## 3.1 Опис структури бази даних

В результаті проведеного попереднього аналізу предметної області, та розробленої архітектури роботи системи визначені основні об’єкти, що підлягають зберіганню в базі даних:

– користувач (здобувач освіти);

– список завдань;

– список вправ;

– список відповідей;

– таблиця обліку відвідувань занять.

PostgreSQL є однією з найпопулярніших систем керування базами даних (СКБД), особливо в контексті хмарних рішень для дистанційного навчання, де важлива ефективна обробка та збереження великих обсягів даних. PostgreSQL – це об’єктно-реляційна СКБД з відкритим кодом, яка підтримує стандарти SQL і пропонує широкі можливості для розробки та керування базами даних [17].

У завданні дистанційного навчання PostgreSQL відіграє ключову роль як сервер керування базами даних, забезпечуючи обробку інформації про користувачів, завдання, вправи, відповіді, а також моніторинг доступу до системи. В архітектурі хмарного середовища для віртуальних лабораторій PostgreSQL забезпечує централізоване збереження даних, що дозволяє всім компонентам системи взаємодіяти через єдиний надійний сервер баз даних.

Однією з основних переваг PostgreSQL є її здатність масштабуватися для обробки великої кількості одночасних запитів, що є критично важливим у дистанційному навчанні, де велика кількість студентів може одночасно виконувати завдання та отримувати доступ до віртуальних лабораторій. Завдяки функціям транзакцій і підтримці високого рівня цілісності даних,

PostgreSQL гарантує, що всі операції над даними виконуються без помилок, навіть у разі збоїв системи або мережі.

PostgreSQL також забезпечує високу продуктивність завдяки підтримці індексів, ефективного кешування, а також паралельного виконання запитів. Для нашого проєкту, де необхідно швидко обробляти дані користувачів, завдання та відповіді, це є вагомою перевагою. Крім того, PostgreSQL має потужні інструменти для адміністрування баз даних, що дозволяє легко керувати доступом до системи, розподіляти права користувачів і забезпечувати безпеку збережених даних.

Ще однією важливою особливістю PostgreSQL є його можливості інтеграції з іншими технологіями, зокрема хмарними серверами. Це забезпечує легке налаштування хмарної інфраструктури, де всі компоненти системи можуть взаємодіяти з сервером бази даних через мережу. У нашому випадку PostgreSQL використовується для зберігання даних про користувачів, завдання, відповіді, а також ведення журналів активності користувачів. Це дозволяє легко масштабувати систему для роботи з великою кількістю студентів та зберігати всі необхідні дані в одній центральній базі.

Використання PostgreSQL у нашому проєкті хмарного середовища для дистанційного навчання дозволяє забезпечити надійність, масштабованість і високу продуктивність системи. Завдяки цьому рішення стає інструментом для управління віртуальними лабораторіями, що дозволяє студентам працювати з завданнями в інтерактивному режимі, зберігаючи всі результати в централізованій базі даних.

На рисунку 3.1 показана структура бази даних хмарного сховища. На основі даної діаграми бази даних можна описати структуру таблиць та їх взаємозв’язки в системі дистанційного навчання. В системі визначені такі основні таблиці:

– User (користувач);

– Task_list (список завдань);

– Exercise (список вправ);

– Answer (список відповідей);

– Login (таблиця обліку відвідувань).

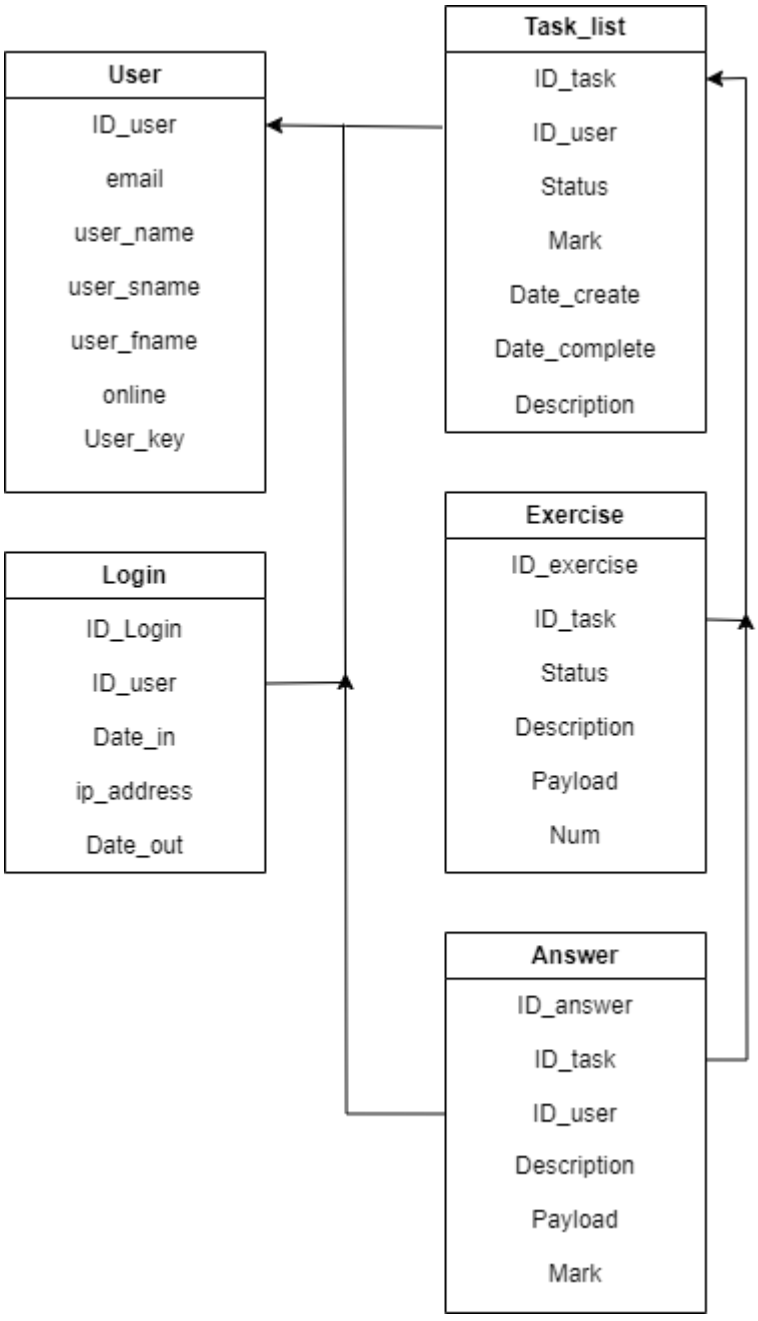


Рисунок 3.1 – Структура бази даних

Таблиця User містить інформацію про користувачів системи, зокрема здобувачів освіти. Вона має такі поля:

– id_user – унікальний ідентифікатор користувача;

– user_name – ім'я користувача;

– user_sname – прізвище користувача;

– user_fname – по батькові користувача;

– email – електронна адреса для ідентифікації користувача;

– User_key – зашифрований ключ, який є ідентифікатором системи, де працює здобувач освіти.

Для опису даного об’єкта створено програмний клас UserMember:

```
public class UserMember
{
   public string ID_user { get; set; }
   public string email { get; set; }
   public string user_name { get; set; }
   public string user_sname { get; set; }
   public string user_fname { get; set; }
   public string online { get; set; }
   public string User_key { get; set; }
}
```

Даний клас використовується як модель даних при доступі до таблиці з основної програми.

Таблиця Task_list зберігає перелік завдань, призначених для користувачів. Вона включає такі поля:

– id_task – унікальний ідентифікатор завдання;

– description – опис завдання, що пояснює, яке завдання має бути виконане;

– mark – поле для виставлення оцінки за завдання;

– Status – показує в якому стані знаходиться завдання (отримане для виконання або виконане);

– Date_create – дата та час додавання завдання до бази даних;

– Date_complete – дата та час завершення виконання завдання здобувачем освіти;

– id_user – зовнішній ключ, що посилається на користувача з таблиці "User", якому призначене це завдання.

Для опису даного об’єкта створено програмний клас Task_listMember:

```
public class Task_listMember
{
   public string ID_task { get; set; }
   public string ID_user { get; set; }
   public string Status { get; set; }
   public string Mark { get; set; }
   public string Date_create { get; set; }
   public string Date_complete { get; set; }
   public string Description { get; set; }
}
```

Таблиця Exercise містить перелік вправ, які студенти виконують у рамках завдань. Поля цієї таблиці:

– id_exercise – унікальний ідентифікатор вправи;

– Status – показує в якому стані знаходиться вправа (отримана для виконання або виконана);

– Description – коротке пояснення або опис вправи;

– Payload – безпосередньо вміст завдання;

– id_task – зовнішній ключ, що посилається на завдання з таблиці "Task_list", до якого належить ця вправа.

Для опису даного об’єкта створено програмний клас ExerciseMember:

```
public class ExerciseMember
{
   public string ID_exercise { get; set; }
   public string ID_task { get; set; }
   public string Status { get; set; }
   public string Description { get; set; }
   public string Payload { get; set; }
```

```
    public string Num { get; set; }
}
```

Таблиця Answer зберігає інформацію про відповіді студентів на вправи та завдання. Її структура включає такі поля:

– id_answer – унікальний ідентифікатор відповіді;

– description – текст відповіді, наданої студентом;

– id_exercise – зовнішній ключ, що посилається на вправу з таблиці "Exercise", до якої надана відповідь;

– id_user – зовнішній ключ, що посилається на користувача з таблиці "User", який надав відповідь.

Для опису даного об’єкта створено програмний клас AnswerMember:

```
public class AnswerMember
{
    public string ID_answer { get; set; }
    public string ID_task { get; set; }
    public string ID_user { get; set; }
    public string Description { get; set; }
    public string Payload { get; set; }
    public string Mark { get; set; }
}
```

Останньою є таблиця Login, яка відображає інформацію про сеанси доступу користувачів до системи, тобто коли і як часто вони входять у систему. Вона містить такі поля:

– id_login – унікальний ідентифікатор входу;

– date_in – час входу користувача в систему;

– Date_out – час виходу користувача із системи;

– id_user – зовнішній ключ, що посилається на користувача з таблиці "User", який здійснив вхід;

– ip_address – адреса з якої заходив студент на заняття.

Для опису даного об’єкта створено програмний клас LoginMember:

```
public class LoginMember
{
   public string ID_Login { get; set; }
   public string ID_user { get; set; }
   public string Date_in { get; set; }
   public string ip_address { get; set; }
   public string Date_out { get; set; }
}
```

## 3.2 Висновки по третьому розділу

В даному розділі кваліфікаційної роботи виконано опис структури бази даних для зберігання робочої інформації хмарного середовища для дистанційного навчання здобувачів освіти. Визначені основні об’єкти, що підлягають зберіганню в базі даних

Розглянута структура бази дозволяє зберігати дані про користувачів, їхні завдання, вправи, відповіді та облік відвідувань. Всі таблиці мають зв'язки через зовнішні ключі, що забезпечує інтеграцію даних і цілісність бази даних.

# 4 РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ХМАРНОГО СХОВИЩА

## 4.1 Розроблення програми для управління віртуальними лабораторіями

Зовнішній вигляд інтерфейсу програми для управління віртуальними лабораторіями показано на рисунку 4.1.

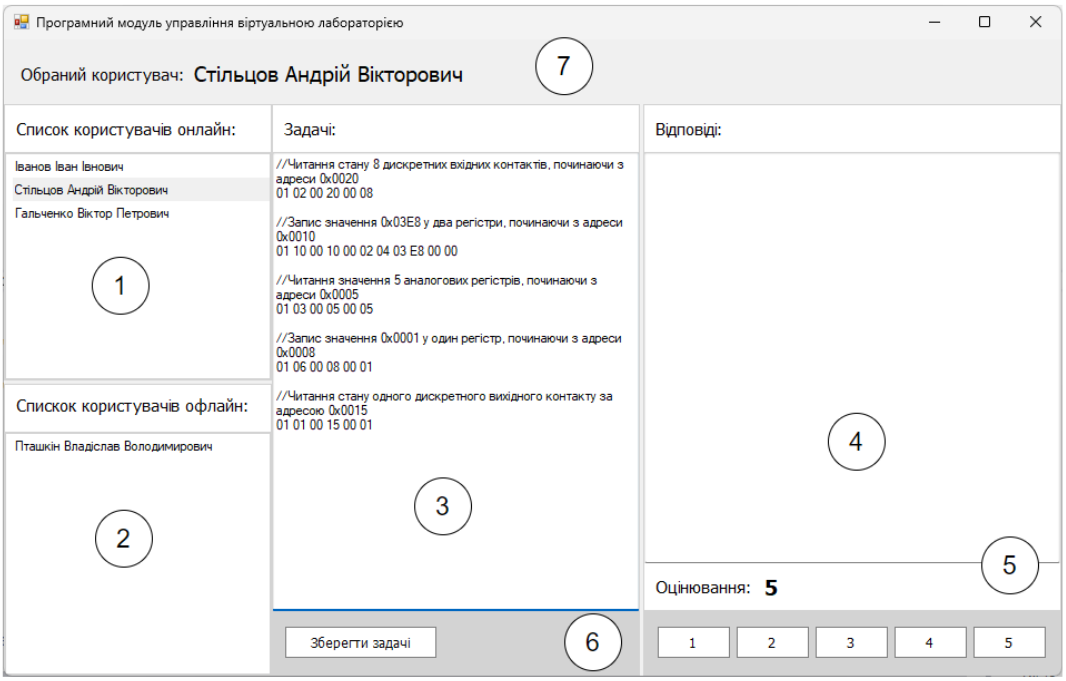


Рисунок 4.1 – Зовнішній вигляд інтерфейсу програми для управління віртуальними лабораторіями

В програмі передбачено декілька робочих областей. В області 1 відображається список студентів, що зараз знаходяться в режимі онлайн. Програма, що працює з боку віртуальних макетів згідно алгоритму з рисунку 2.7, постійно оновлює стан користувача.

В області 2 показано список студентів, що зареєстровані в хмарному середовище, але які зараз знаходяться в режимі офлайн (не підключені до автоматизованої системи).

На рисунку 4.2 показано приклад функції, що заповнює список студентів онлайн.

```
1 reference
private void FillUserOnline()
{
    Control_ViewTable cv = control_ViewUserOnline;
    cv.RowCount = 1;
    string sql = String.Format("Select U.*, (U.user_sname || ' ' " +
        "|| U.user_name || ' ' || U.user_fname) " +
        "as PIB from tUser as U where online = 'true';");

    cv.ID_Field_Name = "ID_User";
    cv.tableStructure.Clear();
    cv.tableStructure.Add(new Control_ViewTable.TableStructure("PIB",
        "string", "PIB", 150, HorizontalAlignment.Left, 0, true));
    cv.createStructure();

    cv.sql_d = sql;
    cv.FillDataList();
}
```

Рисунок 4.2 – Приклад функції, що заповнює список студентів онлайн

Функція FillUserOnline() відповідає за відображення користувачів, які перебувають онлайн, у вигляді таблиці. Перший крок полягає в ініціалізації змінної «cv», яка представляє об'єкт таблиці «control_ViewUserOnline», що відповідає за візуалізацію даних користувачів у форматі таблиці.

Далі формується SQL-запит для вибору даних з таблиці «tUser», яка містить інформацію про користувачів. Запит отримує всі записи користувачів, у яких поле «online» має значення «true», тобто ті, хто перебуває в мережі. Також у запиті формується додаткове поле «PIB», яке об'єднує прізвище, ім'я та по батькові користувача через пробіли, що забезпечує відображення повного імені користувача в таблиці.

Після цього встановлюється параметр «ID_Field_Name», який визначає ідентифікатор користувача в таблиці, в цьому випадку це поле «ID_User». Потім очищується структура таблиці, видаляючи всі попередні налаштування, після чого до структури додається нова колонка «PIB», яка буде містити повне ім'я користувача. Ця колонка має тип даних «string» та ширину 150 пікселів, а текст у колонці вирівнюється по лівому краю.

Наступним кроком викликається метод «createStructure()», який створює структуру таблиці на основі заданих параметрів. SQL-запит, створений на початку функції, присвоюється змінній «sql_d», що дозволяє функції виконати цей запит під час заповнення таблиці.

Останнім кроком функції є виклик методу «FillDataList()», який виконує SQL-запит, отримує дані з бази та заповнює таблицю відповідною інформацією. Таким чином, результатом виконання функції є таблиця, в якій відображаються всі користувачі, що наразі перебувають онлайн, із зазначенням їх повного імені.

Після редагування завдання кожному студенту, його можна зберегти до бази даних. Для цього передбачена область 6 з кнопкою «Зберегти задачі». Фрагмент функції збереження задач показано на рисунку 4.3.

```
1 reference
private void button10_Click(object sender, EventArgs e)
{
    if (SelectTask == "0")
    {
        String sql = String.Format("Insert into Task_list " +
            "(ID_user, Status, Mark, Description) values ('{0}', " +
            "'uncomplete', '0', '{1}');", SelectUser, textBox1.Text);
        DataBase.Exec_SQL(sql);
    }
    else
    {
        String sql = String.Format("Update Task_list " +
            "set Description = '{0}' where ID_Task = '{1}';",
            textBox1.Text, SelectTask);
        DataBase.Exec_SQL(sql);
    }
}
```

Рисунок 4.3 – Фрагмент функції збереження задач

В області 3 відображаються поточні задачі для обраного студента. Задачі мають пояснення та безпосередньо рядок запиту до віртуального макету в форматі Modbus протоколу.

Функція button10_Click виконує операції з базою даних залежно від значення змінної SelectTask, коли користувач натискає на кнопку.

Якщо значення SelectTask дорівнює «0», то це означає, що завдання ще не було створено. У цьому випадку формується SQL-запит для додавання нового запису в таблицю Task_list. Запит вставляє значення полів ID_user (ідентифікатор користувача), Status (статус завдання, який за замовчуванням встановлюється як «uncomplete»), Mark (оцінка, що встановлюється на «0»), та Description (опис завдання, взятий з текстового поля textBox1). Після цього метод Exec_SQL викликається для виконання цього SQL-запиту, що вставляє нове завдання в базу даних.

Якщо значення SelectTask не дорівнює «0», це означає, що завдання вже існує. У такому випадку формується інший SQL-запит, який оновлює опис існуючого завдання в таблиці Task_list. Запит встановлює нове значення Description для завдання, яке відповідає ідентифікатору ID_Task (ідентифікатор завдання). Значення опису також береться з текстового поля textBox1. Цей SQL-запит також виконується за допомогою методу Exec_SQL.

В полі 4 (рис. 4.1) відображається відповідь від студента. Коли завдання виконане, студент завантажує свою відповідь у відповідне поле програми.

В полі 5 є можливість виставити оцінку та зберегти її у відповідне поле в базі даних.

В полі 7 відображається прізвище, ім’я по батькові обраного студента.

## 4.2 Розроблення програми для дистанційного запуску завдань

Програма для дистанційного запуску завдань виконується на боці користувача хмарного сервісу. Завдання даної програми встановити з’єднання

з хмарним сервером та отримати персональний набір вправ, що необхідно виконати.

Зовнішній вигляд основного вікна програми показано на рисунку 4.4.

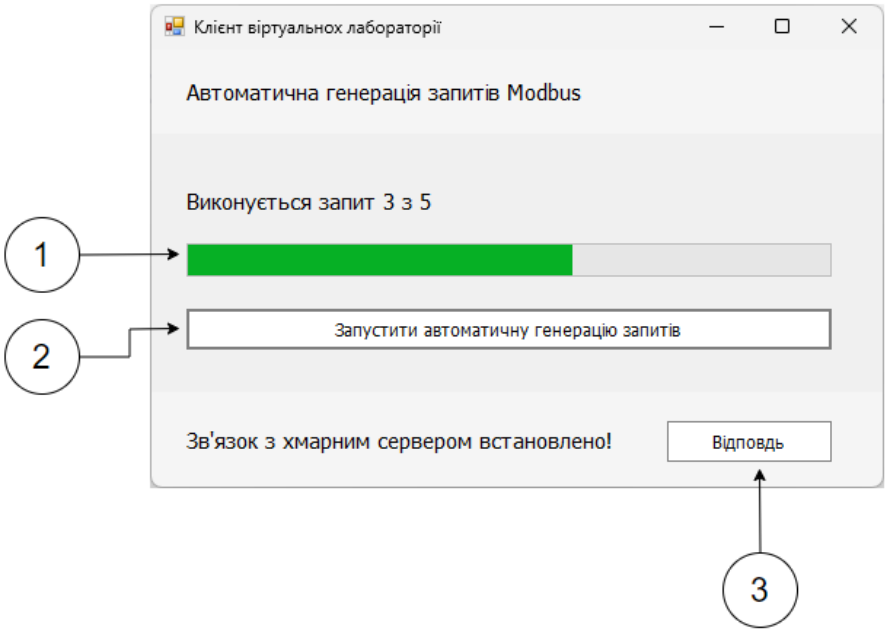


Рисунок 4.4 – Зовнішній вигляд основного вікна програми

В даному вікні відображається прогрес виконання вправ (1) та розташовуються кнопки керування (2 та 3).

Кнопка 2 виконує функцію запуску програмного таймеру для автоматичного виконання команд. На рисунку 4.5 показано приклад даної функції.

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    //Отримання завдання
    String task = "";
    string sql = String.Format("Select TL.* from Task_list as TL where TL.ID_user = '{0}';", SelectUser);
    List<System.Object> rows_user = (List<System.Object>)DataBase.LoadList("DistanceWork.Task_listMember", sql);
    foreach (System.Object ob in rows_user)
    {
        string ID_Task = (ob as Task_listMember).ID_task.ToString();
        string Description = (ob as Task_listMember).Description.ToString();
        task = Description;
        SelectTask = ID_Task;

        //Отримуємо масив завдань
        task_list = SplitTextIntoTasks(task);

        //Запускаємо задачі а автоматичному режимі
        curTask = 0;
        progressBar1.Value = 0;
        progressBar1.Maximum = task_list.Count;
        timer1.Enabled = true;
        label2.Text = String.Format("Виконується запит {0} з {1}", progressBar1.Value, progressBar1.Maximum);
    }
}
```

Рисунок 4.5 – Приклад функції для запуску автоматичного виконання команд

Функція button2_Click виконується після натискання кнопки і має на меті отримати список завдань для обраного користувача та запустити їх в автоматичному режимі. Спочатку ініціалізується порожній рядок task, який буде використовуватися для збереження опису завдання. Потім формується SQL-запит для вибору всіх записів з таблиці Task_list для конкретного користувача з ідентифікатором SelectUser. Цей SQL-запит використовує значення змінної SelectUser для ідентифікації обраного користувача.

Запит виконується за допомогою методу DataBase.LoadList, і результат зберігається у змінну rows_user, яка представляє собою список об’єктів типу Task_listMember. Далі для кожного об’єкта з цього списку (тобто для кожного завдання користувача) здійснюється обробка: отримується ідентифікатор завдання ID_Task, а опис завдання Description зберігається в змінну task. Оновлюється також змінна SelectTask, яка тепер містить ідентифікатор поточного завдання.

Опис завдання, що міститься у змінній task, передається функції SplitTextIntoTasks, яка розбиває текст завдання на окремі частини (задачі). Ці частини зберігаються у списку task_list. Після цього починається автоматичне виконання завдань. Змінна curTask (індекс поточного завдання)

встановлюється на значення 0, значення прогрес-бару progressBar1.Value також встановлюється в 0, а максимальне значення прогрес-бару progressBar1.Maximum задається відповідно до кількості завдань у списку task_list.

Включається таймер (timer1.Enabled = true), який, ймовірно, відповідає за автоматичне виконання завдань у фоновому режимі. Одночасно оновлюється текст метки label2, яка відображає поточний стан виконання завдань, наприклад, "Виконується запит 0 з n". Таким чином, дана функція завантажує завдання для конкретного користувача, розбиває їх на окремі задачі та починає їх автоматичне виконання з використанням таймера, при цьому оновлюючи прогрес виконання завдань в інтерфейсі.

Кнопка 3 (рис. 4.4) відкриває вікно для завантаження відповіді від студента. На рисунку 4.6 показано приклад даного вікна.

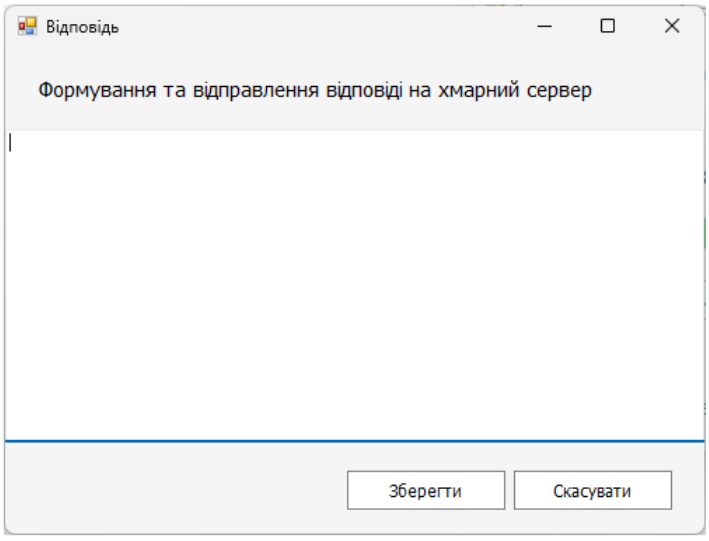


Рисунок 4.6 – Приклад вікна відправки відповіді на хмарний сервер

При першому запуску програми перевіряється наявність на сервері. Якщо відповідного запису не знайдено, то користувачеві висвічується вікно для введення персональних даних. На рисунку 4.7 показано приклад заповнення персональних даних.

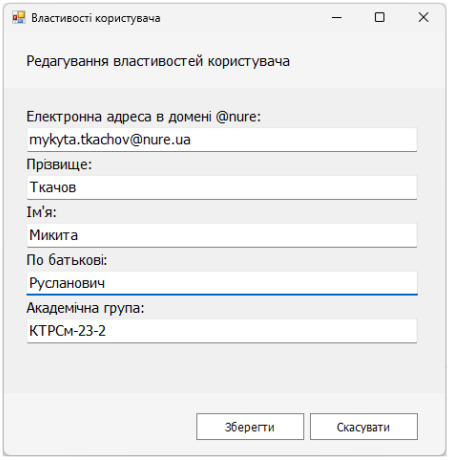


Рисунок 4.7 – Приклад заповнення персональних даних

Після натискання на кнопку зберегти дані записується в базу даних і наступного разу дане вікно вже не з’являється. На рисунку 4.8 показано приклад додавання нового запису до бази даних.

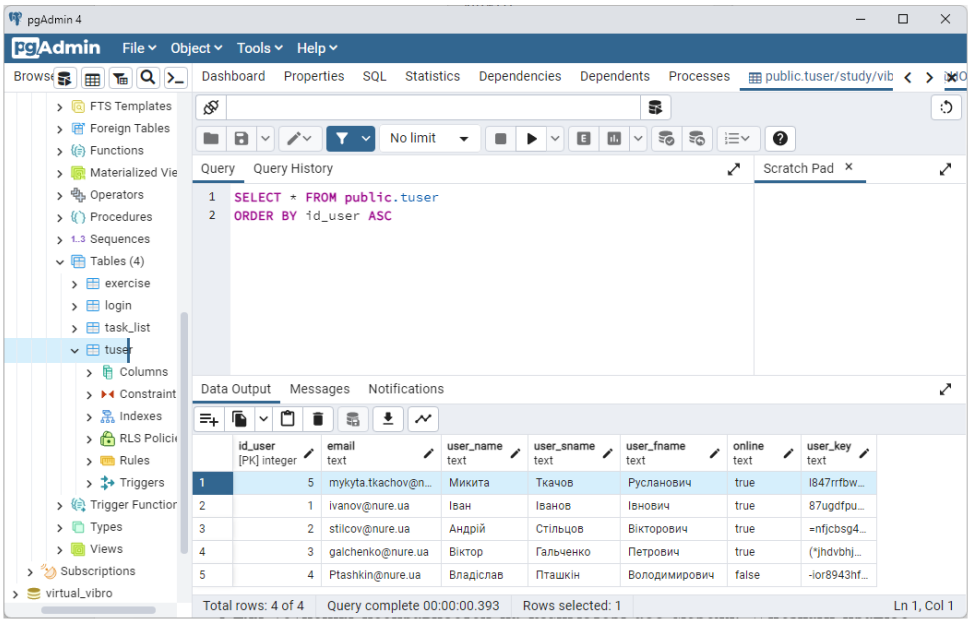


Рисунок 4.8 – Приклад додавання нового запису до бази даних

## 4.3 Експериментальні дослідження

### 4.3.1 Загальний опис та підготовка до проведення експерименту

Проведемо експериментальні дослідження для підтвердження правильності роботи хмарного середовища. Для цього сформуємо завдання, що матиме п’ять різних команд до віртуального макету. Ці команди будуть в автоматичному режимі надсилатись до підлеглого пристрою від програми дистанційного запуску завдань в рамках дослідження впливу випадкових загроз з боку кіберзлочинців до АСУТП. Завданням студента буде виявити загрози та розпізнати надіслані команди за допомогою спеціального програмного інструменті моніторингу мережі Wireshark.

Wireshark – це широко поширений інструмент для захоплення та аналізу мережного трафіку, який активно використовується як для освітніх цілей, так і для усунення несправностей на комп'ютері або мережі. Wireshark працює практично з усіма протоколами моделі OSI, має зрозумілий для звичайного користувача інтерфейс і зручну систему фільтрації даних. Крім того, програма є кросплатформною і підтримує наступні операційні системи: Windows, Linux, Mac OS X, Solaris, FreeBSD, NetBSD, OpenBSD.

Wireshark дозволяє захоплювати і аналізувати трафік на всіх рівнях Інтернет-моделі TCP/IP (канальний, мережевий, транспортний і прикладний) для великої кількості протоколів (мультимедіа, мобільні мережі і т.д.). Це дозволяє локалізувати проблему: проблема з мережею, конфігурацією ПЗ чи помилка ПЗ.

Мережевий інтерфейс – це програмне забезпечення, яке взаємодіє з мережним драйвером та з рівнем IP. Він забезпечує рівню IP доступ до всіх наявних мережних адаптерів, трафік яких ми будемо перехоплювати. Найчастіше у програмі Wireshark можна зустріти мережний інтерфейс бездротової (Wi-Fi) та кабельний (Ethernet).

На рисунку 4.9 показано приклад перехоплення загального мережевого трафіку.

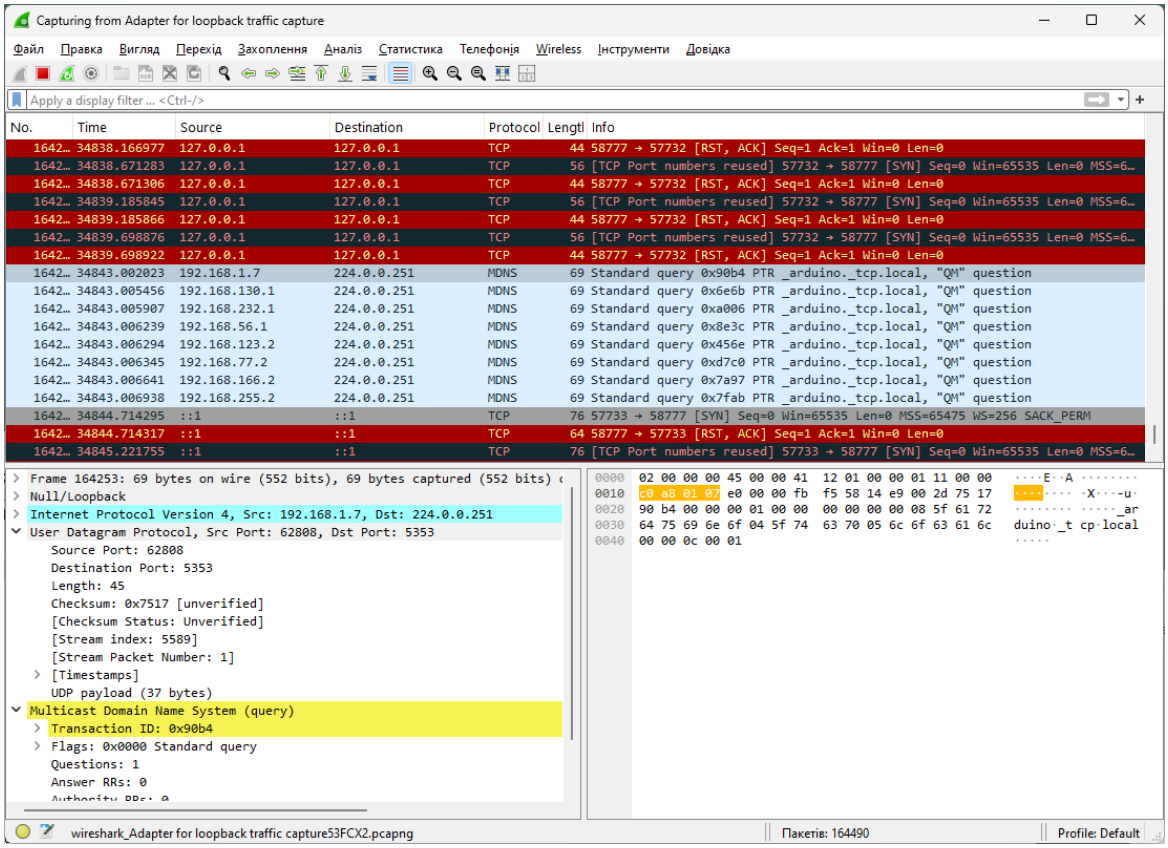


Рисунок 4.9 – Приклад перехоплення загального мережевого трафіку.

4.3.2 Підготовка до проведення експерименту

Виконаємо аналіз протоколу Modbus за допомогою програмного засобу Wireshark. Для проведення експерименту зберемо віртуальний лабораторний стенд, що буде включати наступні компоненти (рис. 4.10):

– аналізатор мережевого трафіку Wireshark (1);

– симулятор підлеглого пристрою Modbus Simulator (2);

– розроблений програмний інструмент для генерації запитів до підлеглого пристрою (3).

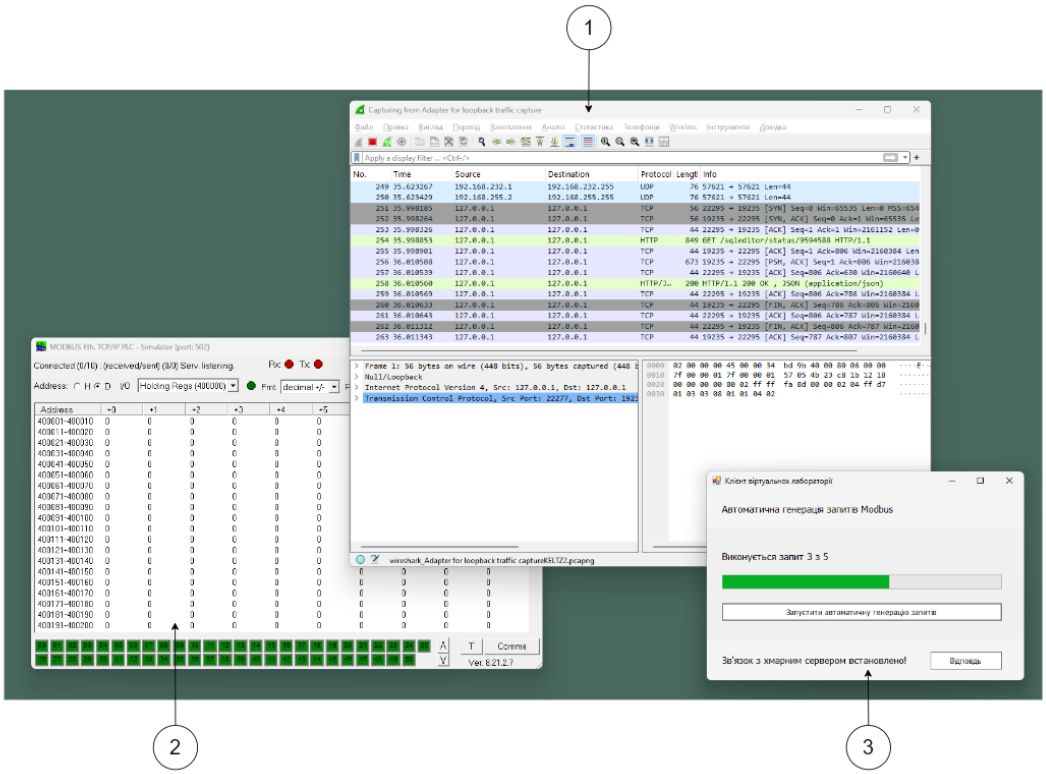


Рисунок 4.10 – Лабораторний стенд для дослідження методів аналізу мережевого трафіку

Програма ModRSsim2 – це симулятор протоколу Modbus для RS-232 та TCP/IP. Програма підтримує наступні реалізації протоколу:

– симуляція підлеглого пристрою з доступом за протоколом Modbus TCP/IP;

– симуляція підлеглого пристрою з доступом за протоколом Modbus RTU.

Середовище надає можливості тестування з використанням різних сценаріїв автоматизації.

Після завантаження програми ModRSsim2 необхідно провести початкові налаштування для роботи з протоколом Modbus. Для цього потрібно обрати відповідний тип протоколу в правому верхньому меню (рис. 4.11).

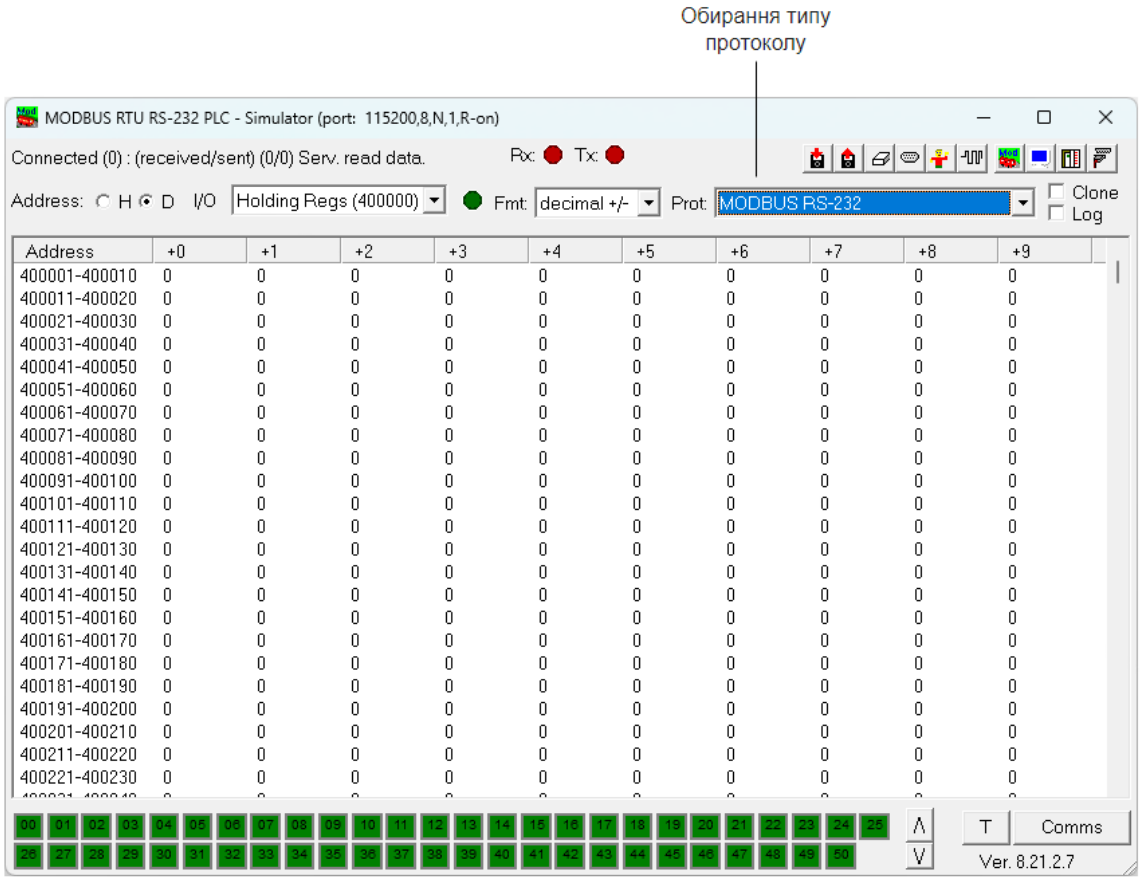


Рисунок 4.11 – Обирання типу протоколу

За допомогою вказівника миші виставимо наступну комбінацію на вході симулятора «01010101» (рис. 4.12).

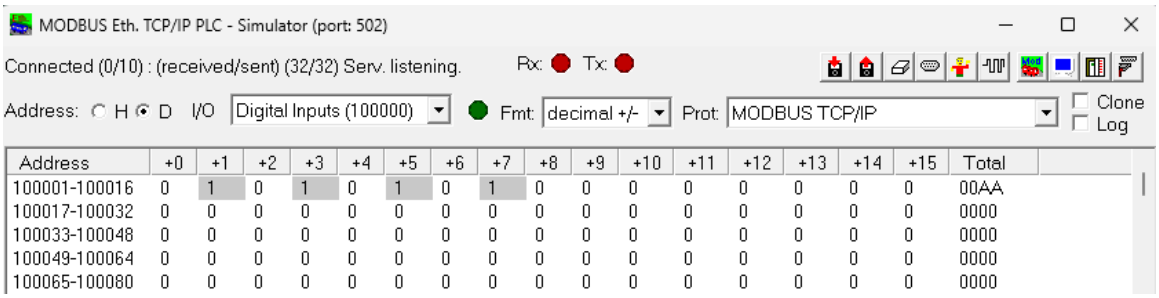


Рисунок 4.12 – Зміна стану вхідних контактів в симуляторі

Для читання стану вхідних контактів використовується функція «02». Щоб прочитати стан восьми вхідних контактів, починаючи з нульової адреси сформую наступний рядок байтів:

01 02 00 00 00 08

Значення обраних байтів описано в таблиці 4.1.

Таблиця 4.1 – Опис байтів тестового запиту читання стану вхідних контактів

| Байт | Опис полів запита |
|---|---|
| 01 | Адреса підлеглого пристрою |
| 02 | Код функції |
| 00 | Адреса початкового біта, Hi байт |
| 00 | Адреса початкового біта, Lo байт |
| 00 | Кількість дискретних входів, Hi байт |
| 08 | Кількість дискретних входів, Lo байт |

Перед виконанням запиту потрібно налаштувати програму Wireshark на перехоплення пакетів, що передаються через порт 502, який є стандартним для протоколу Modbus.

За допомогою програми Wireshark виконаємо аналіз трафіку через порт 502 (рис. 4.13).

В перехопленому трафіку можна виділити пакети, що були направлені від майстра до Modbus симулятора (область 1 на рис. 4.13) та ті, що послані у відповідь від симулятора до розробленої програми (область 2 на рис. 4.13).

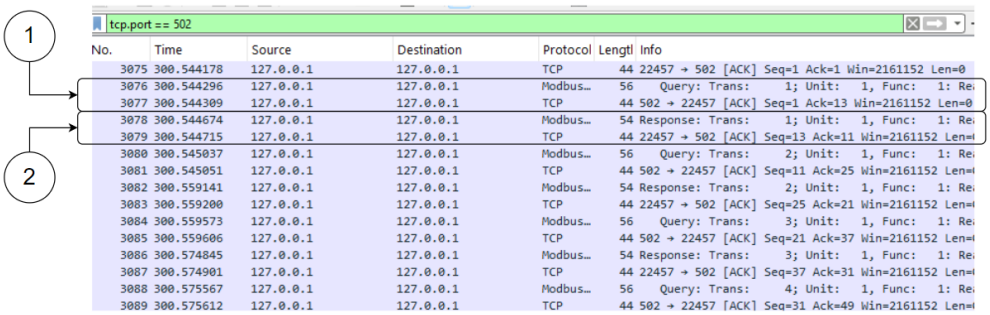


Рисунок 4.13 – Перехоплений трафік через порт 502

Для докладного аналізу пакетів спершу натиснемо на запит від майстра до підлеглого з позначкою «Modbus» в головному вікні та відкриємо панель рівнів моделі OSI (рис. 4.14).

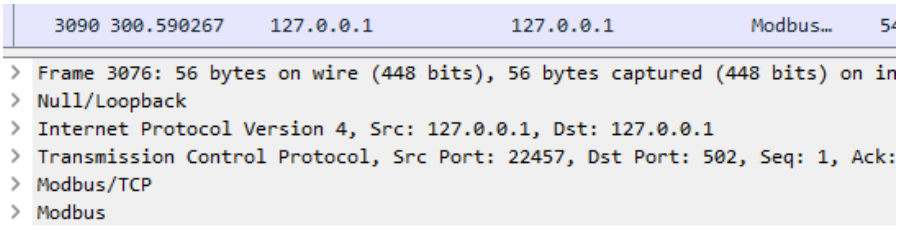


Рисунок 4.14 – Вміст панелі рівнів моделі OSI

На рисунку 4.15 показані основні параметри протоколу, що аналізується.

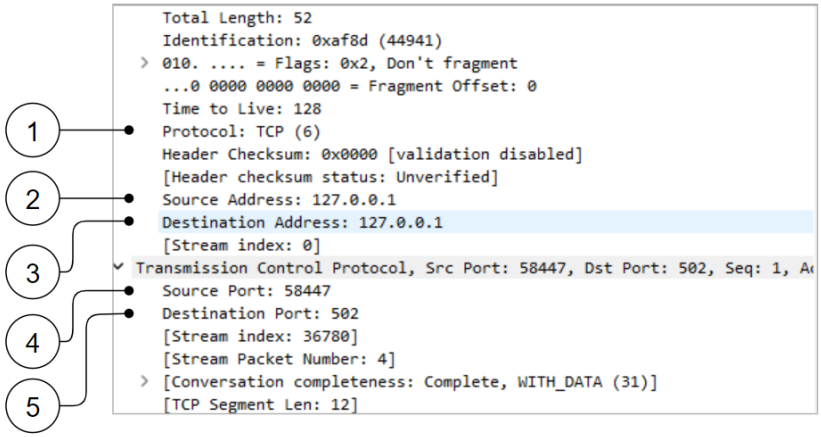


Рисунок 4.15 – Основні параметри протоколу, що аналізується

В результаті аналізу можна виділити наступні характеристики:

– тип протоколу (1) – TCP;

– IP адреса пристрою з якого надіслано повідомлення (2) – 127.0.0.1;

– IP адреса пристрою до якого надійшло повідомлення (3) – 127.0.0.1;

– порт джерела сигналу (4) – 58447;

– порт призначення TCP (5) – 502.

Далі пакет показує рівень Modbus/TCP. Це ADU, який Modbus використовує для зв’язку між пристроями. Wireshark чудово справляється з аналізом окремих полів протоколу Modbus. Ми можемо побачити чотири поля заголовка MBAP в ADU (рис. 4.16):

– ідентифікатор транзакції (1): 1;

– ідентифікатор протоколу (2): 0;

– довжина залишку пакету (3): 6;

– ідентифікатор складової одиниці пакету даних (4): 1.

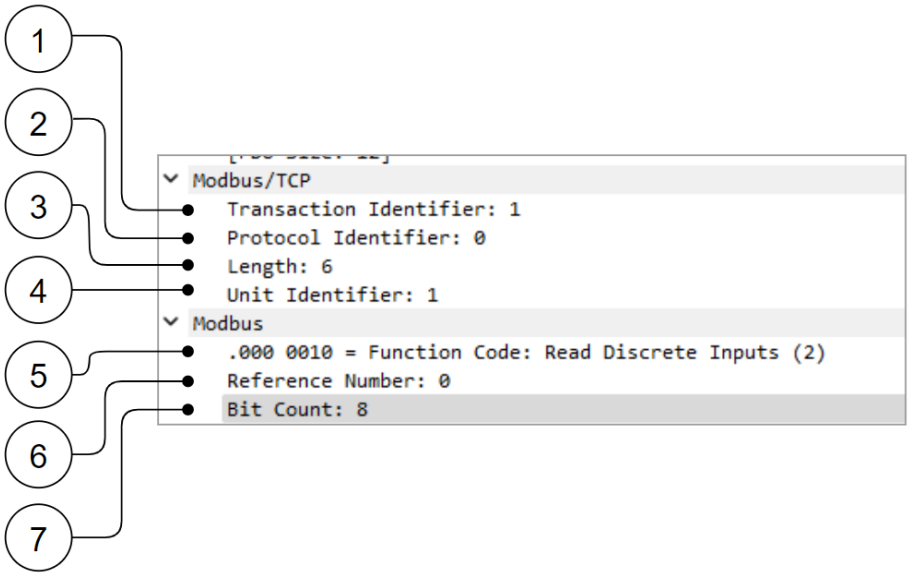


Рисунок 4.16 – Аналіз протоколу Modbus

Крім того також можна побачити поля для PDU кадру Modbus, що описують наступне:

– код функції: 2 (читання дискретних входів);

– номер зміщення відповідно початкової адреси: 0 (початок з дискретного входу з номером 0);

– кількість дискретних входів - 8 (читати 8 дискретних входів поспіль).

Натискаючи на відповідні поля протоколу в панелі рівнів OSI в панелі метаданих можна докладно подивитись на вміст пакету даних. Кольором буде підсвічено відповідні байти, що становлять обраний параметр протоколу Modbus (рис. 4.17).

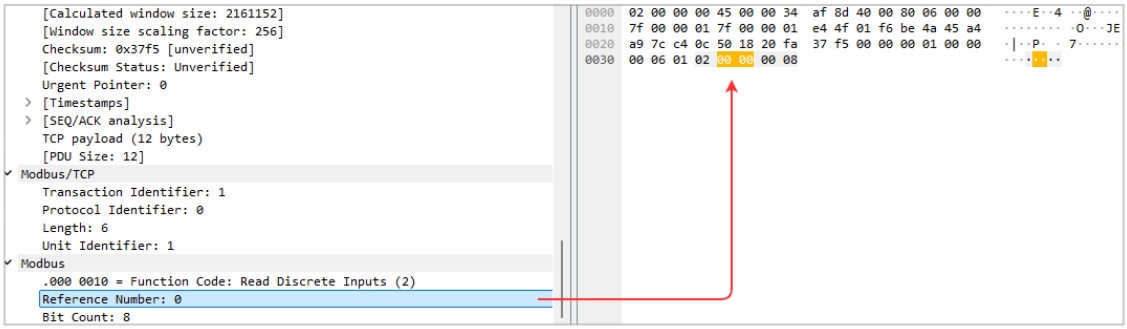


Рисунок 4.17 – Аналіз вмісту полів протоколу в режимі запиту

Реалізація частини даних PDU залежить від запитуваної функції. У цьому випадку за допомогою функції 2 (Read Descrete Inputs), частина даних PDU інтерпретується як окремі біти (рис. 4.17). Для функції 1, Read Coils, дані також будуть інтерпретуватися як біти, а для інших функцій частина даних PDU може містити додаткову інформацію разом, щоб підтримувати запит функції.

## 4.4 Виробнича санітарія в лабораторії

Робота в приміщенні проводиться сидячи і не вимагає фізичної напруги. Тому вона відноситься до категорії Іа (легкі фізичні роботи, енерговитрати до 120 ккал/ч). З метою забезпечення комфортних умов для працівників та

відповідно до ДСН 3.3.6.042-99 в приміщенні встановлені наступні оптимальні показники мікроклімату:

– для холодного періоду: температура повітря від 22 °С до 24 °С; вологість повітря від 40 % до 60 %; швидкість руху повітря оптимальна 0,1 м/с;

– для теплого періоду року: температура повітря від 23 °С до 25 °С; вологість повітря від 40 % до 60 %; швидкість руху повітря оптимальна 0,1 м/с, допустима 0,2 м/с.

Пайка металевих виводів радіоелементів при їх монтажі на друковану плату модуля свинцево-олов'яними припоями (Chemet Sn60Pb40 Кар-Т) супроводжується виділенням в атмосферу аерозолів свинцю, олова і парів розчинників.

Для розрахунку необхідного повітрообміну необхідно в кожному випадку визначати кількість шкідливих речовин, що виділяються у повітря приміщення.

Пайка здійснюється свинцево-олов'яним припоєм Chemet Sn60Pb40 Кар-Т, який містить 0,39 частки обсягу свинцю (с) і 61 % олова.

Найбільш отруйні аерозолі (пари) свинцю. В процесі пайки з припою випаровується до В = 0,1% свинцю, а на 1 пайку витрачається 10 мг припою. При числі пайок N, кількість виділених парів свинцю визначається як:

$$G = c \cdot B \cdot m_{пр} \cdot N\,, \qquad (4.1)$$

де с – частка обсягу свинцю в припої;

$B$ – частка виділених парів шкідливої речовини;

$m_{пр}$ – маса припою на одну пайку, мг;

$N$ – кількість контактів за годину.

В приміщенні об'ємом $V_n$ = 138,88 м$^3$ три людини здійснюють паяння припоєм Chemet Sn60Pb40 Kap-T продуктивністю по 50 контактів в годину. Визначаємо кількість аерозолів свинцю, що виділяються в повітря:

$$G = 039 \cdot 0,001 \cdot 10 \cdot 50 = 0,195 \text{ мг/ч.}$$

Необхідний повітрообмін визначається за формулою:

$$L = \frac{G \cdot 1000}{x_в - x_н}, \quad (4.2)$$

де $L$ –необхідний повітряобмін, м$^3$/год;

$G$ – кількість речовин, що виділяються у повітря приміщення, г/ч;

$x_в$ – гранично допустима концентрація шкідливих речовин в повітрі робочої зони приміщення, згідно СНБ 3.02.03-03, мг/м$^3$;

$x_н$ – максимально можлива концентрація тієї ж шкідливості в зовнішньому повітрі, відповідно до ДСН 3.3.6.042-99, мг/м$^3$

$$L = \frac{0,195}{0,01 - 0} = 19,5 \text{м}^3/\text{год.}$$

Визначаємо кількість повітря, яке необхідно подати в робочу зону для того, щоб забезпечити необхідну кількість повітря на працюючого

$$L_1 = n \cdot L_{люд}, \quad (4.3)$$

де $L_{люд}$ – норма подачі припливного повітря на 1 людину (згідно СніП 2.04.05-91 $L$ = 60 м$^3$/год).

$$L_1 = 2 \cdot 60 = 120 \text{ м}^3/\text{год.}$$

Порівнюючи норми подачі $L_{люд}$ і $L$ для подальших розрахунків приймаємо найбільше значення. Знайдемо кратність повітрообміну $n$, яке показує, скільки разів упродовж однієї години повітря повністю змінюється в приміщенні:

$$n = \frac{L}{V_n}, \tag{4.3}$$

де $V_n$ – внутрішній об'єм приміщення, м$^3$

$$n = \frac{120}{138,88} = 0,86 \text{ ч}^{-1}.$$

Таким чином, для розрахованих параметрів обрана енергозберігаюча припливно-витяжна установка Dantex DV 250E - DV1200E з продуктивністю 150 - 1200 м$^3$/год, що забезпечить необхідну кратність повітрообміну.

Приміщення лабораторії має природне та штучне освітлення відповідно до ДБН Ст. 2.5-28-2006.

Природне світло проникає через бічні світлопроєми, зорієнтовані на північ, і забезпечують коефіцієнт природної освітленості (КПО) не нижче 1,2%.

Необхідна освітленість приміщення 200-500лк.

Згідно ДСН 3.3.6.037-99 рівень шуму в лабораторії не перевищує 50 дБА.

## 4.5 Висновки по четвертому розділу

В результаті виконання четвертого розділу кваліфікаційної роботи виконане розроблення хмарного середовища для дистанційного навчання студентів. Хмарне середовище включає в себе:

– програмний засіб для віддаленого управління віртуальними лабораторіями;

– хмарний сервер з сервером баз даних PostgreSQL;

– віртуальної лабораторії до складу якої входить програма для автоматичного виконання завдань та віртуальні макети для відпрацювання цих завдань.

Описані графічні інтерфейси розроблених програм та наведені приклади основних функцій. Проведено експериментальні дослідження для перевірки правильності виконання реалізованих алгоритмів.

Розглянуто приклад роботи програми на основі автоматичного опрацювання завдань, що пов’язані з моделюванням ситуації кібератаки на засоби автоматизації АСУТП. Завданням користувача було виявити втручання в роботу технічного засобу автоматизації та розпізнати всі команди, що були надіслані від особи тестового кіберзлочинця.

Проведені експерименти показали працездатність запропонованого рішення.

Також було описано норми виробничої санітарії в лабораторії.

# ВИСНОВКИ

В результаті виконання кваліфікаційної роботи розроблено хмарне середовища для дистанційного навчання студентів. Розглянуто приклад роботи програми на основі автоматичного опрацювання завдань, що пов’язані з моделюванням ситуації кібератаки на засоби автоматизації АСУТП. Завданням користувача було виявити втручання в роботу технічного засобу автоматизації та розпізнати всі команди, що були надіслані від особи тестового кіберзлочинця.

Проведено аналіз предметної області, а саме, аналіз актуальності дослідження. Розроблення програми на основі хмарного середовища для дистанційного навчання студентів є актуальним завданням, яке відповідає сучасним викликам цифрової трансформації освіти.

Проведено аналіз протоколів Інтернету речей, описані рівні архітектури IOT для поєднання хмарного середовища з віртуальними макетами. Розглянуті особливості побудови даної мережі в концепції Інтернету речей, враховуючи процеси розробки засобів автоматизації навчання здобувачів освіти, що використовуються в процесі дистанційного навчання з використанням віртуальних макетів.

Надано відомості про особливості основних протоколів організації зв’язку в IOT та доступу до хмарного сховища.

Розроблені структурні схеми віртуальної лабораторії, хмарного серверу, програми управління віртуальними лабораторіями. Розроблені необхідні алгоритми роботи програмних компонентів автоматизованої системи та описано принцип її роботи.

Виконано опис структури бази даних для зберігання робочої інформації хмарного середовища для дистанційного навчання здобувачів освіти. Визначені основні об’єкти, що підлягають зберіганню в базі даних

Розглянута структура бази дозволяє зберігати дані про користувачів, їхні завдання, вправи, відповіді та облік відвідувань. Всі таблиці мають зв'язки через зовнішні ключі, що забезпечує інтеграцію даних і цілісність бази даних.

Проведені експерименти показали працездатність запропонованого рішення.

Таким чином, розроблена архітектура хмарного середовища для дистанційного навчання студентів забезпечує гнучкість і масштабованість навчального процесу. Використання хмарних серверів дозволяє зберігати і обробляти великі обсяги даних, надавати студентам віддалений доступ до віртуальних лабораторій і дозволяє викладачам ефективно керувати навчальним процесом. Це середовище не тільки дозволяє студентам взаємодіяти з цифровими моделями реальних лабораторних процесів, але і робить можливим навчання незалежно від місця перебування користувача, що значно підвищує доступність і якість освіти.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. ДСТУ 3008–2015. Звіти у сфері науки і техніки. Структура і правила оформлення. Документація. – Введ. 2015-06-22. - К.: Держстандарт України, 2015. - 31 с.

2. Методичні вказівки з підготовки та захисту кваліфікаційної роботи здобувачами другого (магістерського) рівня вищої освіти спеціальності 174 Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітньо-професійних програм: «Комп'ютерно-інтегровані технологічні процеси і виробництва», «Комп'ютеризовані та робототехнічні системи» / Упоряд. І. Ш. Невлюдов, Р. В. Артюх, В. В. Безкоровайний, Н. П. Демська, В. В. Євсєєв, О. І. Филипенко, О. М. Цимбал. Харків: ХНУРЕ, 2023. 55 с..

. Кафедра технології та автоматизації виробництва РЕЗ та ЕОЗ [Електронний ресурс] – Режим доступу: www/ URL: https://tapr.nure.ua/wp-

4. Основи наукових досліджень : підручник / І. Ш. Невлюдов, Ю. М. Олександров, А. О. Андрусевич, О. О. Чала ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Prague : OKTAN PRINT, 2024. – 468 с.

5. Положення про кваліфікаційну роботу здобувача вищої освіти на другому (магістерському) рівні [Електронний ресурс] : Наказ ХНУРЕ від 06 травня 2021 р. № 143. – Режим доступу: https://nure.ua/wp-content/uploads/Main_Docs_NURE/143-vid-06.05.2021-pro-vvedennja-v-diju-rishennja-vchenoi-radi-universitetu.pdf.

6. Положення про протидію академічному плагіату в Харківському національному університеті радіоелектроніки [Електронний ресурс] : наказ ректора ХНУРЕ від 28.04.2017 р. № 290 // Нормативно-правова база ХНУРЕ : офіційний веб-портал. – Режим доступу: https://nure.ua/universytet/normativno-pravova-baza#id13. – Станом на 18.09.2024. – Назва з екрану.

7. Невлюдов І. Ш. Застосування цифрових двійників технічних засобів автоматизації для розроблення програмно-технічних комплексів АСУ ТП : Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова. – Харків: Видавництво Іванченка І. С., 2023. – 267 с.

8. Node-RED та технологія промислового Інтернету речей : Навчальний посібник / І. Ш. Невлюдов, С. П. Новоселов, О. В. Сичова. – Харків: Видавництво Іванченка І. С., 2024. – 207 с.

9. Digital Twins for Industrial Applications [Електронний ресурс] : Industrial Internet Consortium, a program of Object Management Group, Inc. (“OMG”). – Режим доступу: https://www.iiconsortium.org/pdf/IIC_Digital_Twins_Industrial_ Apps_White_ Paper_2020-02-18.pdf. – Станом на 01.10.2024. – Назва з екрану.

10. Internet of Things, IoT. IT-Enterprise – your one-stop platform for digital transformation | www.it.ua. URL: https://www.it.ua/knowledge-base/technology-innovation/internet-veschej-internet-of-things-iot (date of access: 31.12.2024).

11. Three Layer Architecture in the IoT. Specifications and Security Threats. KITRUM. URL: https://kitrum.com/blog/three-layer-architecture-in-the-internet-of-things/ (date of access: 31.12.2024).

12. Understanding IoT edge gateways and their benefits | Compulab. Compulab | System on Modules, IoT Gateways, Industrial Computers. URL: https://www.compulab.com/blog/understanding-iot-edge-gateways-and-their-benefits/ (date of access: 31.12.2024).

13. Огляд мережевих протоколів бездротового Інтернету речей і способи вибору. Dusun IoT: Embedded Hardware Vendor/Manufacturer | IoT Gateway Expert. URL: https://www.dusuniot.com/uk/blog/best-wireless-protocol-for-your-iot-project/ (дата звернення: 31.12.2024).

14. Data Distribution Service | IoT Applications | ADLINK. ADLINK Technology. URL: https://www.adlinktech.com/en/data-distribution-service (date of access: 31.12.2024).

15. AMQP 0-9-1 Model Explained | RabbitMQ. RabbitMQ: One broker to queue them all | RabbitMQ. URL: https://www.rabbitmq.com/tutorials/amqp-concepts (date of access: 31.12.2024).

16. MQTT - The Standard for IoT Messaging. MQTT - The Standard for IoT Messaging. URL: https://mqtt.org/ (date of access: 31.12.2024).

17. PostgreSQL. PostgreSQL. URL: https://www.postgresql.org/ (date of access: 30.12.2024).