НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО	**ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ**
Декан факультету (Директор ННІ)	**Завідувач кафедри**
Інформаційних технологій (назва факультету(ННІ))	Комп'ютерних систем, мереж та кібербезпеки (назва кафедри)
_____ Болбот І.М., д.т.н, проф. (підпис) (ПІБ, вчене звання і ступінь)	_____ Касаткін Д.Ю., к. пед.н., доц. (підпис) (ПІБ, вчене звання і ступінь)
«___»_____2025 р.	«___»_____2025 р.

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка автоматизованої системи контролю доступу з функцією трекінгу відвідуваності»

Спеціальність 123 «Комп’ютерна інженерія»
(код і найменування)

Освітня програма Комп’ютерні системи захисту інформації
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми
д.пед.н., професор _____ Мамченко С.М.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи
д.т.н., професор _____ Лахно В.А.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Таран Е.М.
(підпис) (ПІБ)

КИЇВ-2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ
Завідувач кафедри
комп'ютерних систем, мереж та кібербезпеки
к.пед.н., доц. _____ Касаткін Д.Ю.
(вчене звання і ступінь) (підпис) (ПІБ)
«__» _____ 20__ р.

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ ЗДОБУВАЧУ

Таран Едуард Михайлович
(прізвище, ім'я, по батькові)

Спеціальність 123 «Комп'ютерна інженерія»
(код і найменування)

Освітня програма Комп'ютерні системи захисту інформації
(назва)

Орієнтація освітньої програми Освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи: «Розробка автоматизованої системи контролю доступу з функцією трекінгу відвідуваності»

затверджена наказом ректора НУБіП України від "29" жовтня 2024р. № 1941 «С»

Термін подання завершеної роботи на кафедру 14 листопада 2025 р.

Вихідні дані до магістерської кваліфікаційної роботи вихідними даними є вимоги до створення емулятора подій АСКД, специфікація модулів серверної логіки, параметри роботи контролерів ESP32, сценарії навантаження та формат обміну даними між MQTT-, HTTP- та DB-компонентами.

Перелік питань, що підлягають дослідженню:

1. дослідження архітектури емулятора та впливу параметрів навантаження на точність моделювання подій.
2. Оцінювання роботи алгоритмів авторизації й обробки подій: швидкодія, затримки, стійкість.
3. Аналіз інтеграції з MQTT-брокером і базою даних: стабільність, логування, відтворюваність експериментів.

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання " 29 **"** жовтня 2024 р.

Керівник магістерської кваліфікаційної роботи _____ Лахно В.А.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Таран Е.М.
(підпис) (прізвище та ініціали)

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

1. ACCS – Access Control and Counting System, система контролю доступу та трекінгу відвідуваності.
2. API – Application Programming Interface, програмний інтерфейс взаємодії.
3. DB – Database, база даних.
4. ESP32 – Embedded System Platform 32-bit, мікроконтролер з Wi-Fi/BLE модулем.
5. HTTP(S) – HyperText Transfer Protocol (Secure), протокол передавання гіпертексту (захищений).
6. JSON – JavaScript Object Notation, формат представлення структурованих даних.
7. KPI – Key Performance Indicator, ключовий показник ефективності.
8. LAN – Local Area Network, локальна обчислювальна мережа.
9. MQTT – Message Queuing Telemetry Transport, телеметричний протокол обміну повідомленнями.
10. MQTT QoS1 – Quality of Service Level 1, гарантія доставки «принаймні один раз».
11. p95 / p99 – перцентилі затримок обробки (95-й і 99-й відповідно).
12. REST – Representational State Transfer, архітектурний стиль побудови веб-API.
13. RSSI – Received Signal Strength Indicator, рівень потужності прийнятого сигналу.
14. SQL – Structured Query Language, мова структурованих запитів.
15. TLS – Transport Layer Security, протокол захисту транспортного рівня.
16. UI – User Interface, інтерфейс користувача.
17. UID – User Identifier, унікальний ідентифікатор користувача.

18. UUID – Universally Unique Identifier, глобальний унікальний ідентифікатор.

19. WAN – Wide Area Network, глобальна мережа.

20. ЗЦД – зона контролю доступу (логічна область з політиками авторизації).

21. ПС – програмна система.

22. СУБД – система управління базами даних.

23. ТЗ – технічне завдання.

24. ТЛС-канал – захищений канал передавання даних за протоколом TLS.

ВСТУП

Необхідність забезпечення високого рівня контрольованості, прозорості та інтегрованості процесів доступу до об’єктів з розгалуженою інфраструктурою — такими як навчальні заклади, адміністративно-офісні комплекси, виробничі приміщення та технічні зони - обумовлює актуальність створення інтелектуальних систем контролю фізичного доступу. Більшість традиційних рішень, що базуються на автономних контролерах і фіксованих схемах авторизації, не дозволяють повноцінно враховувати часові інтервали перебування користувачів, адаптуватися до динамічних політик безпеки або синхронізуватися з зовнішніми інформаційними системами обліку кадрів, пропускного режиму чи обліку робочого часу. У сучасних умовах цифрової трансформації, де ключову роль відіграють гнучкість, масштабованість і міжсистемна сумісність, виникає потреба у впровадженні автоматизованих систем контролю доступу нового покоління, які функціонують у режимі реального часу, підтримують подієво-орієнтовану архітектуру, дозволяють реєструвати, обробляти й аналізувати події проходу з повною історією дій користувачів. Особливе значення має використання захищених стандартів взаємодії (MQTT, HTTP, WebSocket), дотримання вимог до обробки персональних даних та підтримка централізованої логіки реагування на події доступу.

Метою роботи є розробка програмно-апаратної системи контролю доступу з функцією трекінгу відвідуваності, яка об'єднує контролери зчитування подій, серверну логіку обробки, журналювання та візуалізацію результатів, і забезпечує достовірну реєстрацію доступів, гнучке керування правами користувачів, обробку виняткових ситуацій і формування звітності за часовими інтервалами, зонами та профілями доступу.

Для досягнення поставленої мети необхідно вирішити наступні **завдання:**

– провести огляд сучасних рішень у сфері АСКД та визначити їхні функціональні обмеження;

– формалізувати вимоги до архітектури, функціоналу, безпеки та відмовостійкості системи;

– побудувати структурну модель системи з виділенням ключових модулів: зчитувачів, контролерів, каналу зв’язку, серверної частини, бази даних та інтерфейсу адміністратора;

– розробити прошивку для периферійного пристрою (наприклад, на базі ESP32) із підтримкою подієвої генерації повідомлень;

– реалізувати серверний модуль обробки подій на Python із функціями логування, перевірки прав доступу, виявлення аномалій і формування звітності;

– забезпечити інтеграцію з базою даних (наприклад, SQLite/PostgreSQL) для трекінгу та аналітики;

– провести тестування системи на відповідність нефункціональним вимогам (продуктивність, затримки, стійкість до втрати з’єднання);

– проаналізувати результати експериментів, визначити ключові технічні параметри та сформулювати рекомендації щодо масштабування та розгортання.

Об’єктом дослідження є автоматизовані системи контролю доступу з фіксацією подій входу/виходу.

Предметом дослідження є методи побудови архітектури та алгоритмів обробки подій, що реалізуються в рамках подієво-орієнтованої АСКД з функцією трекінгу.

Методами дослідження виступають системний аналіз архітектур безпеки, структурно-функціональне моделювання програмних і апаратних компонентів, імітаційне тестування потоків подій, вимірювання метрик продуктивності, побудова сценаріїв стрес-навантаження та аналіз логів подій.

Практична цінність роботи полягає у створенні універсальної, гнучкої, легко масштабованої системи, яка може бути розгорнута у навчальних закладах, бізнес-центрах, медичних установах або виробничих зонах без потреби у

складному впровадженні — з підтримкою локальної або хмарної інфраструктури, простого конфігурування політик доступу та перегляду звітності.

Наукова новизна дослідження полягає у реалізації подієво-орієнтованої моделі серверної обробки з динамічною маршрутизацією подій доступу, виявленням аномалій та механізмом трекінгу відвідуваності в реальному часі, що поєднує мікроконтролерну периферію та програмну обробку з використанням MQTT-протоколу, структурованого журналювання та аналітичного формування звітів.

Структура роботи включає вступ, чотири розділи, висновки, список використаних джерел та додатки. У першому розділі викладено аналіз предметної області, класифікацію існуючих систем контролю доступу та методів трекінгу; у другому — розроблено архітектурну модель, визначено вимоги, побудовано діаграми структурної та функціональної взаємодії компонентів; третій розділ присвячено реалізації апаратної та програмної частин, з описом API, логіки обробки та сценаріїв роботи; у четвертому розділі наведено результати тестування, порівняльний аналіз метрик та рекомендації щодо впровадження й подальшого розвитку. Висновки узагальнюють отримані результати та окреслюють перспективи практичного застосування системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Автоматизовані системи контролю доступу (АСКД) є критично важливими компонентами в екосистемах безпеки об'єктів, де існує потреба в обмеженні доступу до фізичних зон, реєстрації подій входу/виходу та відстеженні присутності персоналу в реальному часі. У сучасних умовах зростаючих вимог до безпеки, відповідності регуляторним нормам, а також до інтегрованості з інформаційною інфраструктурою підприємств, АСКД еволюціонують у напрямку подієво-орієнтованих платформ з підтримкою трекінгу відвідуваності, аналітики навантаження та генерації оперативної звітності [1]. Класичні реалізації систем часто є монолітними, обмеженими в масштабуванні та інтеграції з HRM, ERP або SIEM-рішеннями, що унеможливлює оперативну адаптацію до змін у політиках безпеки чи бізнес-процесах.

На рисунку 1.1 наведено класифікаційну схему систем контролю доступу з функцією трекінгу відвідуваності, яка відображає основні структурні компоненти та взаємозв’язки між підсистемами. Зокрема, система поділяється на функціональні блоки за ознаками призначення, типів комунікацій, обробки даних, архітектурних рішень, аспектів безпеки, а також варіантів розгортання.

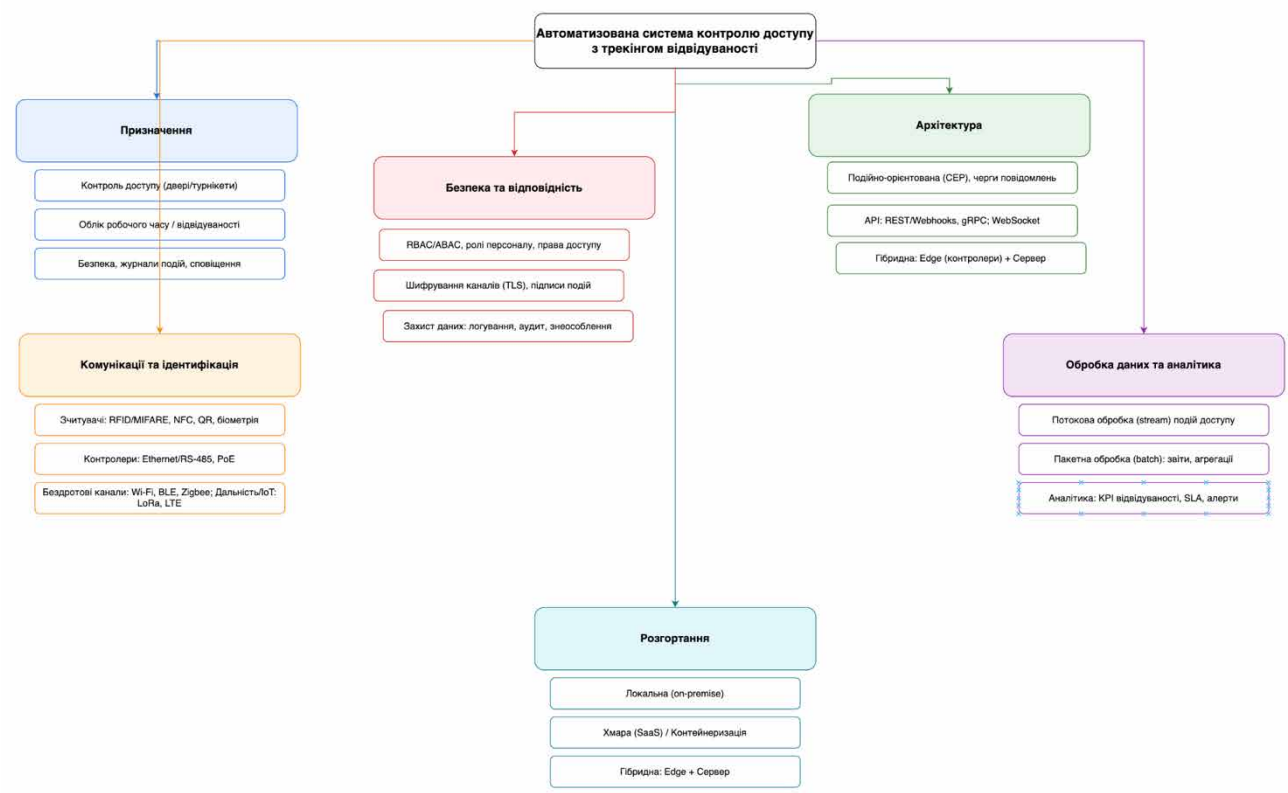

Рис. 1.1 – Класифікація автоматизованих систем контролю доступу з трекінгом відвідуваності

Призначення систем охоплює контроль фізичного доступу (двері, турнікети), облік робочого часу та виявлення аномалій у поведінці користувачів. Підсистема комунікації забезпечує зв’язок між зчитувачами (RFID/NFC, біометрія), контролерами (Ethernet, PoE) та сервером, використовуючи як провідні канали (RS-485), так і бездротові (Wi-Fi, LoRa, Zigbee). Архітектура системи базується на подієвій моделі з використанням CEP-механізмів, API (REST/Webhooks/gRPC) та гібридної обробки (Edge + Server), що дозволяє забезпечити низькі затримки та гнучке масштабування [2]. Підсистема безпеки охоплює RBAC/ABAC-моделі прав доступу, TLS-шифрування, аудит та логування подій.

Інтеграція обробки подій у форматах stream (реактивна обробка) та batch (агрегація звітів) дозволяє реалізувати гнучку аналітику: SLA, KPI відвідуваності, динаміку змін прав доступу тощо. Вибір платформи розгортання (on-premise, контейнеризоване середовище, хмара) визначається вимогами замовника до приватності, контролю над даними та надійності. Узагальнюючий

порівняльний аналіз основних характеристик сучасних АСКД представлено в таблиці 1.1.

Таблиця 1.1

Ключові характеристики АСКД нового покоління

Компонент	Характеристика
Призначення	Контроль доступу, трекінг відвідуваності, сигналізація
Канали зв’язку	Ethernet, RS-485, Wi-Fi, BLE, LoRa, LTE
Ідентифікація	RFID, NFC, QR, біометрія
Архітектура	Гібридна (контролер + сервер), подієва модель
Обробка подій	Stream (реактивна), Batch (звіти), CEP
Безпека	TLS, підписи подій, аудит, логування, RBAC/ABAC
Аналітика	KPI, SLA, звіти по зонах/часу/подіях
Розгортання	On-premise, контейнеризоване (Docker), хмарне (SaaS)

Метою дослідження в межах предметної області є розробка автоматизованої системи контролю доступу з трекінгом відвідуваності, яка забезпечує надійну ідентифікацію користувачів, реєстрацію подій проходу, централізовану обробку доступу та аналітику перебування. Система повинна поєднувати апаратні засоби зчитування, безпечну передачу даних, гнучку архітектуру обробки подій та адаптивні механізми формування звітності, з можливістю масштабування, інтеграції з зовнішніми системами й дотриманням вимог до збереження персональних даних.

1.2 Огляд існуючих апаратних і програмних рішень

Системи контролю доступу широко реалізуються у вигляді комерційних апаратно-програмних комплексів, а також відкритих або частково модульних конфігурацій для індивідуального застосування. З метою виявлення архітектурних патернів, типових комунікаційних інтерфейсів і методів підключення було проаналізовано низку реальних рішень, де доступна структурна або електрична схема. Це дозволило сформувати уявлення про логіку зчитування, передачу подій, схеми керування замками, комунікації між

контролерами та виконавчими пристроями, а також типову логіку роботи підсистем входу/виходу.

На рисунку 1.2 подано фрагмент силового підключення до блоку живлення змінного струму (115V AC), у якому ідентифіковано стандартну кольорову маркування: чорний (фаза), білий (нейтраль), зелений (заземлення). Цей підхід застосовується для внутрішнього підключення модулів живлення контролерів доступу в промислових установках.

Рис. 1.2 – Основне силове підключення 115V AC для контролера доступу

На рисунку 1.3 зображено схему підключення кнопок відкриття дверей (зовнішньої та внутрішньої) до SDC-контролера. Обидві кнопки працюють через нормально відкриті (N/O) реле, які замикаються при натисканні, подаючи імпульс на вхід контролера. Також показано інфрачервоний сенсор (SDC AUTO-IR), який активується при виявленні присутності, працюючи паралельно з кнопками.

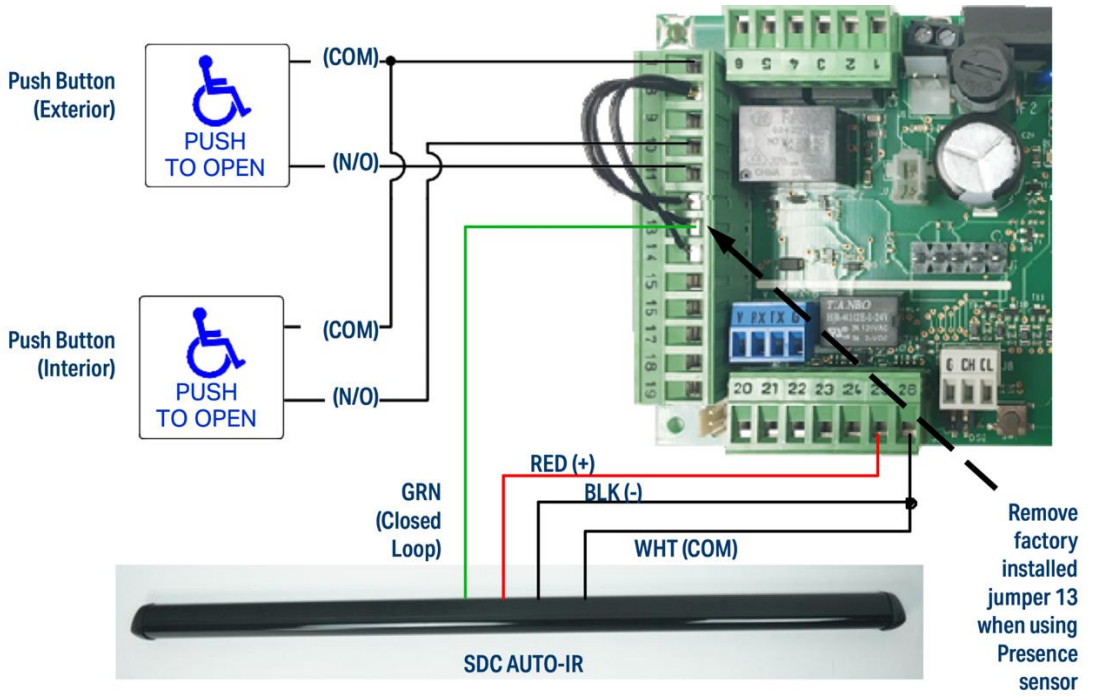

Рис. 1.3 – Двостороння схема підключення кнопок до SDC-контролера

На рисунку 1.4 подано приклад схеми підключення контролера до зовнішнього блоку живлення з використанням двох релейних виходів. Живлення 12/24VDC подається на клеми NO/C/NC, а замикання здійснюється за допомогою активного стану реле. Такий принцип використовується у випадках, коли керування замками відбувається через логіку в зовнішньому модулі.

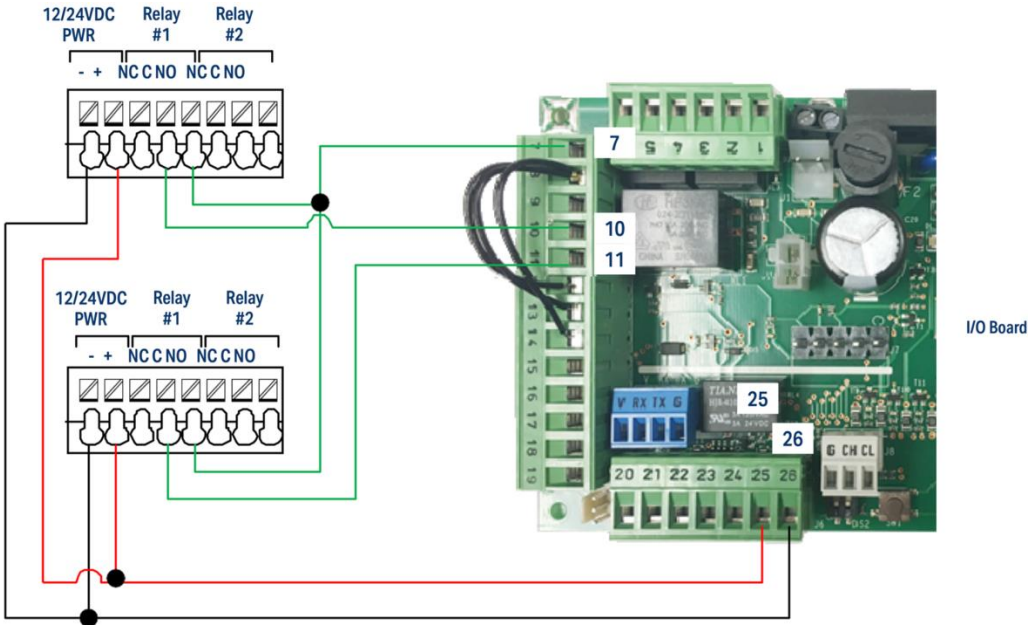

Рис. 1.4 – Живлення електрозамків через зовнішні реле 12/24VDC

На рисунку 1.5 проілюстровано повну систему, що включає сенсори безконтактного доступу, кнопку відкриття, електрозасувку (Fail Secure), захисні

елементи (MOV), і підключення до I/O плати контролера. Схема демонструє практичне рішення з урахуванням комутаційних перемичок, логіки живлення та сигналізації.

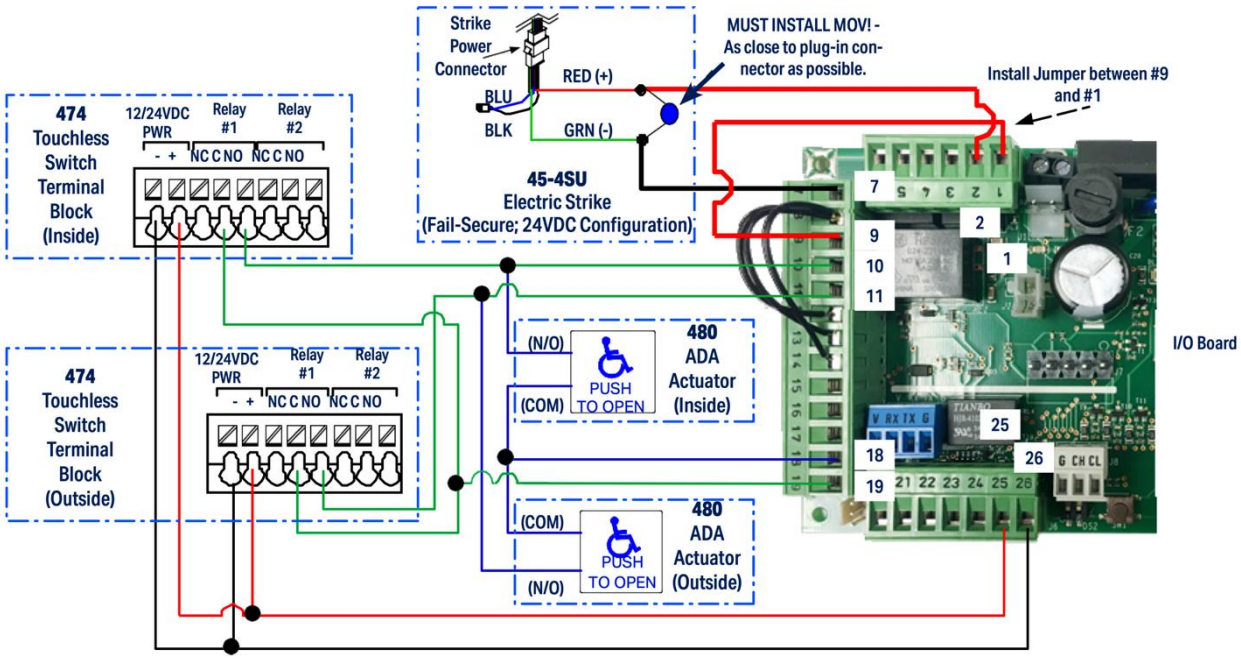

Рис. 1.5 – Інтеграція з електрозасувкою та безконтактними сенсорами

Реалізацію двох блоків Touchless Switch 474 (внутрішній і зовнішній), які керують відкриттям дверей. Передбачено подвійне підключення до джерела живлення, електрозасувки та контролера. Архітектура схеми побудована на логіці замикання двох незалежних каналів із подальшою маршрутизацією подій.

На рисунку 1.6 зображено на головній сторінці веб-сайту оглядову схему від ArcSite, що демонструє централізовану PoE-інфраструктуру: RFID-зчитувач підключений до контролера через CAT6-кабель, живлення та дані передаються по одному каналу. Схема ілюструє тренд на уніфікацію комунікаційної та енергетичної інфраструктури за допомогою стандартів Ethernet та IP-контролю.

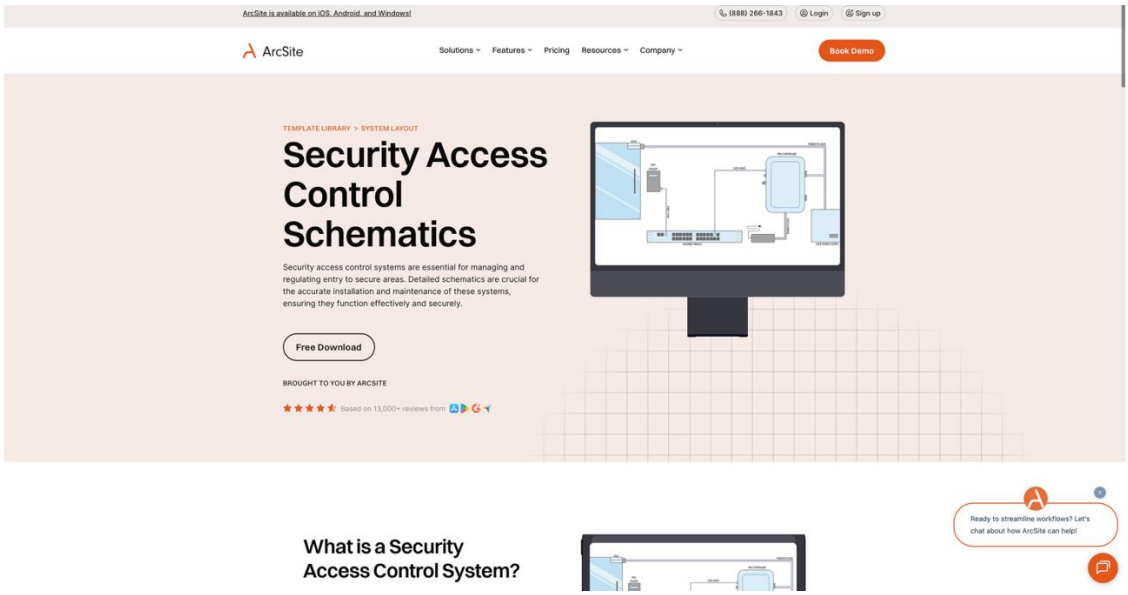

Рис. 1.6 – (Головна сторінка веб-сайту) PoE-схема керування доступом з використанням CAT6 Ethernet

На рисунку 1.7 наведено деталізовану схему ArcSite із візуалізацією контролера, блоків живлення, реле та механізму замка. Система охоплює повний цикл: від авторизації користувача до фізичного зняття замка з живленням через комутатор. Такий підхід дозволяє централізовано керувати доступом на рівні мережі.

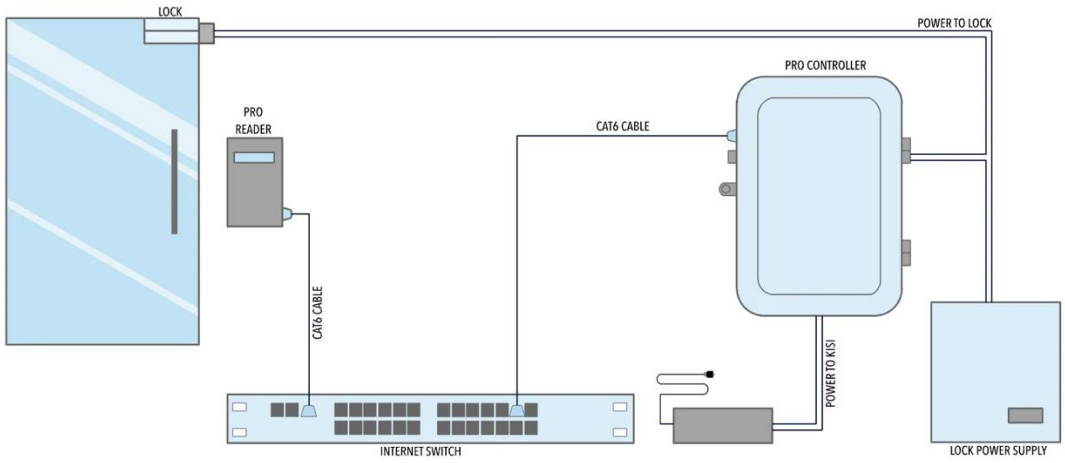

Рис. 1.7 – Повна структурна схема централізованої PoE-системи доступу

Для систематизації характеристик аналізованих рішень, включно з проектованою у роботі системою на базі ESP32 з асинхронною логікою та MQTT-комунікацією, у таблиці 1.2 наведено порівняння за п’ятьма критичними

критеріями: архітектура контролера, метод доступу, спосіб комунікації, журналювання та аналітика.

Таблиця 1.2

Порівняння апаратних і програмних рішень для АСКД

Система / Пристрій	Контролер / логіка	Метод доступу	Комунікація	Аналітика / Журналювання
SDC Auto Entry Control	Автономна плата SDC, реле, I/O	Кнопка, сенсор присутності	Провідна, замикання реле	Відсутня
474 Touchless Actuator System	Touchless логіка, 2 реле (внутрішній і зовнішній)	Бездотикові сенсори	Провідна з I/O-платою	Лише подієвий контакт
SDC Access Control + Relay	Гібридне керування з живленням 12/24VDC	Кнопки, сенсор + електрозамок	Провідна, зовнішнє живлення	Без історії подій
ArcSite Schematics (PoE)	PoE-контролер, API підключення	RFID + інтернет-контроль	PoE + CAT6	Історія подій через API
Наша система (ESP32)	ESP32, асинхронна обробка подій на Python	RFID/NFC/QR/Біометрія (модульно)	Wi-Fi + MQTT/HTTP + TLS	Повна база подій, аналітика, формування звітів

Як свідчить порівняльний аналіз, більшість комерційних рішень зосереджені на простому апаратному виконанні з мінімальним функціоналом аналітики. Водночас запропонована система вирізняється високим рівнем адаптивності, можливістю масштабування, веденням журналів, збором KPI та візуалізацією статистики в режимі реального часу. Це робить її придатною як для малих організацій, так і для середніх підприємств, що прагнуть реалізувати сучасний підхід до управління безпекою та обліком присутності.

1.3 Аналіз вимог системи контролю

Ефективність роботи автоматизованої системи контролю доступу з трекінгом відвідуваності безпосередньо залежить від повноти, узгодженості та реалізованості вимог, що формуються на етапі проєктування. У системах цього класу вимоги поділяються на чотири категорії: функціональні, нефункціональні, технічні та апаратні. Кожна з них визначає окремий аспект — від логіки обробки подій до фізичної сумісності компонентів. Формалізація вимог дозволяє здійснити модульне проєктування з подальшим поетапним тестуванням, уникаючи конфліктів між функціональністю, продуктивністю та безпекою.

Функціональні вимоги фокусуються на базових та розширених можливостях системи. Вона повинна підтримувати ідентифікацію користувача за допомогою RFID, NFC, QR-кодів або біометрії; забезпечувати миттєву реєстрацію подій входу та виходу з фіксацією часових міток; перевіряти права доступу відповідно до встановлених ролей і зон; зберігати події у базі даних і формувати аналітичні звіти за параметрами відвідуваності. У таблиці 1.3 наведено перелік функціональних вимог.

Таблиця 1.3

Функціональні вимоги до системи трекінгу

ID	Функціональна вимога
F1	Ідентифікація користувачів за RFID/NFC/QR/біометрією
F2	Реєстрація подій входу/виходу з фіксацією часу
F3	Перевірка прав доступу за ролями (RBAC/ABAC)
F4	Ведення журналу відвідувань у централізованій базі даних
F5	Генерація статистики трекінгу за періодами, локаціями, користувачами

Нефункціональні вимоги визначають показники якості системи, які не стосуються конкретної функціональності, але є критичними для продуктивності, безпеки та взаємодії з користувачем. Вони включають час реакції на подію (менше 200 мс), надійність реєстрації (відмовостійкість), захист трафіку між контролером і сервером (TLS-шифрування), а також сумісність з

веб-інтерфейсами для моніторингу та адміністрування. Деталізація наведена в таблиці 1.4.

Таблиця 1.4

Нефункціональні вимоги до системи

ID	Нефункціональна вимога
NF1	Максимальний час обробки події — не більше 200 мс
NF2	Гарантована реєстрація подій з вірогідністю ≥ 99.9%
NF3	Передача даних між компонентами виключно за допомогою TLS/HTTPS
NF4	Сумісність адміністративного інтерфейсу з сучасними браузерами

Технічні вимоги формалізують вибір протоколів, інтерфейсів та інфраструктурних компонентів, що забезпечують реалізацію системи. Система повинна підтримувати MQTT або HTTP REST як транспортні шини для подій, базу даних для зберігання історії (SQLite чи PostgreSQL), та реалізований API-доступ до сервера. Веб-інтерфейс адміністратора передбачається на Flask або FastAPI, з можливістю інтеграції через JSON-інтерфейси. Таблиця 1.5 подає перелік технічних вимог.

Таблиця 1.5

Технічні вимоги до серверної та програмної частини

ID	Технічна вимога
T1	Протокол обміну: MQTT (QoS 1) або HTTP REST API
T2	База даних: SQLite для локального розгортання, PostgreSQL для масштабованих версій
T3	Серверна логіка: Python 3.10+, Flask/FastAPI, асинхронна подієва модель
T4	JSON-формат обміну даними, підтримка CORS та авторизації

Окремо слід визначити апаратні вимоги, які обмежують вибір електронних компонентів для забезпечення повної сумісності та коректної роботи. Контролер повинен базуватись на мікросхемі ESP32 з підтримкою Wi-Fi та Bluetooth; зчитувачі мають бути сумісні з протоколами ISO/IEC 14443-A (наприклад, MFRC522); виконавчі механізми – електромагнітні замки або реле 5В; а також

стабілізоване живлення 5В/12В для периферії. Повний перелік наведено у таблиці 1.6.

Таблиця 1.6

Апаратні вимоги до системи трекінгу

ID	Апаратна вимога
H1	Контролер ESP32 з підтримкою Wi-Fi, BLE та GPIO
H2	RFID/NFC-зчитувач типу MFRC522 з підтримкою SPI
H3	Електромагнітний замок або реле 5V для керування замиканням дверей
H4	Джерело живлення 5V/12V з фільтрацією і захистом

Комплексний аналіз цих вимог дозволяє сформулювати архітектуру системи як подієво-орієнтовану розподілену модель, що забезпечує низьку затримку, високу надійність, підтримку аналітики й масштабованість. На відміну від автономних рішень з фіксованою логікою, спроєктована система буде підтримувати сценарії з кількома точками доступу, централізовану базу подій, журналювання та API-взаємодію з зовнішніми модулями, зберігаючи водночас простоту розгортання на доступному апаратному забезпеченні.

1.4 Постановка завдання

З урахуванням результатів аналізу предметної області та системних вимог сформульовано завдання на розробку архітектурно-організаційної моделі автоматизованої системи контролю доступу з функцією трекінгу відвідуваності. Метою проєкту є створення технічного рішення, здатного виконувати ідентифікацію користувачів, фіксацію подій доступу, верифікацію прав, а також накопичення, обробку та подальший аналіз інформації про відвідування визначених зон об’єкта.

Система повинна забезпечувати функціонування в умовах подієвого середовища з високою частотою подій, підтримувати модульну побудову, масштабування в розрізі точок доступу та централізовану обробку даних. Архітектура має передбачати мінімізацію затримок, ізоляцію критичних

підсистем, стійкість до збоїв комунікацій і сумісність із сучасними каналами передачі даних (MQTT, HTTP, RS-485, Ethernet).

Контролюючі пристрої мають забезпечувати базову логіку авторизації та передачі подій у середовище обробки, при цьому зберігаючи здатність функціонування у деградованому режимі при втраті з'єднання. Система повинна підтримувати єдиний формат подій з параметрами: тип дії, унікальний ідентифікатор користувача, часовий штамп, точка доступу, результат авторизації.

На рівні обробки подій система має забезпечувати перевірку ролей, формування журналу, контроль цілісності записів, сегментацію доступу до історичних даних, а також агрегування інформації для подальшого формування звітів про дисципліну відвідуваності, перевищення порогів, нетипові події, відхилення від нормативних шаблонів. Дані мають бути збережені в уніфікованій структурі з підтримкою реплікації та резервування.

З урахуванням зазначеного, сформульовано перелік основних завдань дослідження:

1. Провести систематизований аналіз сучасних апаратних і програмних рішень у сфері контролю доступу та трекінгу присутності.
2. Сформулювати вимоги до функціональності, надійності, масштабованості та захищеності системи.
3. Розробити структурну та функціональну модель системи з подієвою архітектурою.
4. Сформувати концепцію маршрутизації подій, контролю доступу та журналювання.
5. Побудувати модель потоків даних між контрольними пристроями, підсистемою авторизації та сховищем подій.
6. Визначити засоби формування та структурування звітів, критерії аналітики та моделі оцінки відхилень.
7. Запропонувати технічні засоби й архітектурні принципи для реалізації системи у фізичному середовищі або у вигляді симуляційного стенду.

Завдання полягає у побудові повнофункціональної архітектурної моделі системи з можливістю адаптації до конкретного середовища розгортання, підтримкою обробки подій у реальному часі, централізованого зберігання історії доступу та реалізації трекінгових сценаріїв із застосуванням уніфікованих стандартів даних та інтерфейсів.

1.5 Висновки до першого розділу

У першому розділі здійснено системний аналіз предметної області автоматизованих систем контролю доступу (АСКД) із функцією трекінгу відвідуваності, що дало змогу сформувати цілісне уявлення про структуру, принципи роботи та тенденції розвитку сучасних рішень у цій сфері. Визначено, що сучасні АСКД переходять від автономних, апаратно орієнтованих моделей до інтегрованих подієво-аналітичних систем, здатних взаємодіяти з корпоративною інфраструктурою підприємства, HRM- та ERP-модулями, а також із зовнішніми сервісами моніторингу. Проведена класифікація систем за функціональними, архітектурними та комунікаційними ознаками (рис. 1.1, табл. 1.1) дозволила виокремити основні блоки побудови сучасних рішень: підсистеми ідентифікації, обробки подій, аналітики, безпеки та управління доступом.

Порівняльний огляд існуючих апаратних і програмних платформ (SDC, ArcSite та ін.) показав, що більшість комерційних систем мають обмежену підтримку аналітики й відсутність розширених механізмів інтеграції. Запропонована у цій роботі концепція системи на базі контролера ESP32 із асинхронною логікою, протоколом MQTT і серверною обробкою подій у середовищі Python/FastAPI забезпечує суттєву перевагу у масштабованості, відмовостійкості та можливості централізованої обробки подій (табл. 1.2).

Формалізація вимог до системи (табл. 1.3–1.6) дала змогу визначити повний набір функціональних, нефункціональних, технічних та апаратних характеристик, що є критичними для її реалізації: швидкодію (≤ 200 мс),

надійність фіксації подій (≥ 99.9 %), безпечну передачу даних (TLS/HTTPS), модульність і підтримку кількох типів ідентифікації (RFID, NFC, QR, біометрія). Встановлено, що оптимальним рішенням для реалізації цілей є подієво-орієнтована гібридна архітектура з розподілом обробки між польовими пристроями (Edge) і серверною аналітикою (Server), що забезпечує гнучкість і надійність функціонування навіть у разі тимчасових збоїв мережі.

У результаті проведеного аналізу сформульовано завдання подальшого проєктування системи - створення інтелектуальної, масштабованої АСКД з трекінгом відвідуваності, яка об’єднує апаратні й програмні компоненти в єдине інформаційне середовище, підтримує реєстрацію та аналітику подій у реальному часі, інтегрується з корпоративними сервісами та забезпечує високий рівень безпеки даних. Отримані висновки стали науково-технічною основою для розробки моделі системи та побудови емуляційного стенду, що розглядається у другому розділі.

2 ПРОЄКТУВАННЯ ЕМУЛЯТОРА СЕРВЕРНОЇ ОБРОБКИ ПОДІЙ АВТОМАТИЗОВАНОЇ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ

2.1 Функціональна схема та логічна архітектура емулятора подій

Розроблення функціональної схеми емулятора серверної обробки подій у системі контролю доступу з функцією трекінгу відвідуваності має ключове значення для верифікації архітектурних рішень, оптимізації потоків даних та забезпечення узгодженості між апаратними і програмними модулями. На рисунку 2.1 подано узагальнену функціональну схему, що відображає послідовність обробки подій - від генерації телеметрії емулятором ESP32 до маршрутизації повідомлень через MQTT-брокер і подальшої інтерпретації серверною логікою. Система організована за багаторівневим принципом із виділенням шарів Edge, Network, Server Logic та Analytics, що забезпечує модульність і нормалізацію каналів обміну.

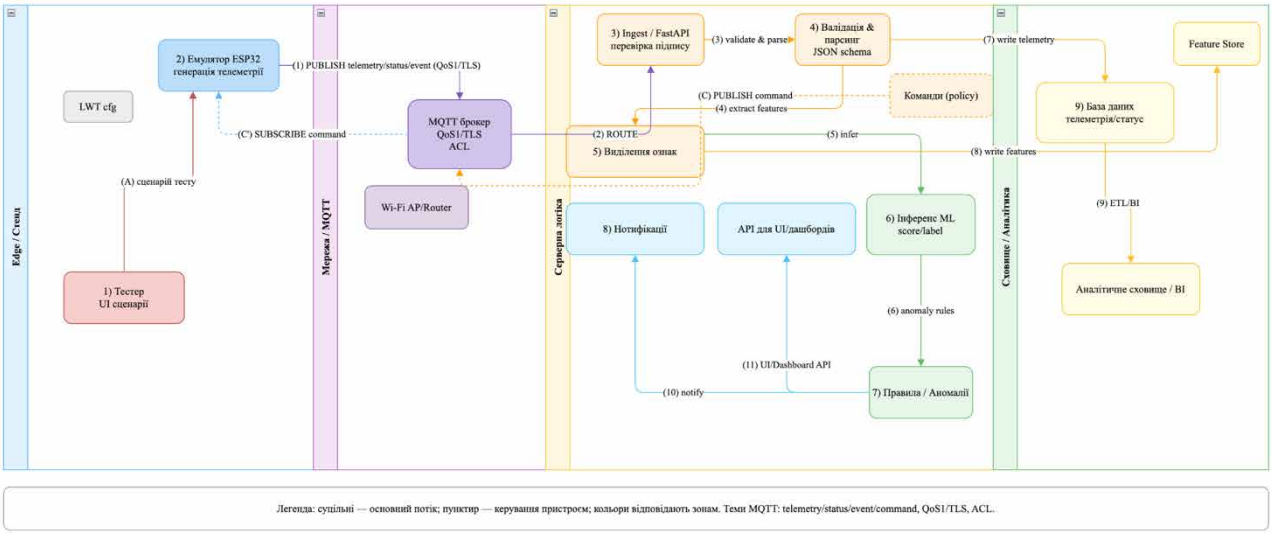

Рис. 2.1 – Функціональна схема емулятора серверної обробки подій

У межах архітектури реалізовано принцип подієво-орієнтованої взаємодії, де кожен вузол системи є джерелом або споживачем подій. Емулятор ESP32 формує тестові пакети телеметрії з атрибутами часу, ідентифікаторам користувача та статусом події, які передаються до MQTT-брокера з параметрами QoS 1 і TLS-шифруванням. Серверна частина, розроблена на Python

(Flask/FastAPI), приймає ці повідомлення, здійснює перевірку цілісності, парсинг JSON-структур, виділення ознак і збереження результатів у базі даних. Така побудова забезпечує розмежування потоків за призначенням і мінімізує ризики дублювання або втрати даних при масштабуванні.

Особливістю реалізації є нормалізація каналів передачі подій, що передбачає єдину структуру тем MQTT (telemetry/status/event/command) і уніфікований формат обміну даними. Це дозволяє незалежно тестувати модулі - контролери, сервер і брокер - без необхідності фізичної інтеграції всіх компонентів, що особливо важливо на етапі емуляції. На відміну від традиційних стендів із фіксованими з'єднаннями, запропонована схема підтримує сценарії динамічної підписки та маршрутизації, зберігаючи реалістичність мережевих затримок та протоколів авторизації.

Для узагальнення взаємозв'язків між основними модулями системи у таблиці 2.1 наведено характеристику рівнів обробки та типів передаваних повідомлень.

Таблиця 2.1

Логічна структура каналів обміну подіями в емуляторі системи

Рівень системи	Основні модулі	Тип повідомлень	Призначення каналу
Edge (Стенд)	Емулятор ESP32, тестер сценаріїв	telemetry, status	Генерація подій і відправка телеметрії
Network (MQTT)	MQTT-брокер, Wi-Fi AP	event, command	Маршрутизація та шифрований транспорт подій
Server Logic	FastAPI / Flask сервер, API	validate, route, log	Обробка, перевірка прав доступу, журналювання
Analytics	База даних, Feature Store, BI	telemetry, features	Збереження, аналітика та формування звітів

Завдяки такій структурі забезпечується узгодженість між каналами передачі, підтримка реактивної обробки подій і можливість адаптації системи під різні конфігурації контролерів. Реалізована схема дозволяє досліджувати поведінку системи в умовах високої частоти подій, визначати межі пропускної

здатності каналів і формувати базу даних для подальшого аналітичного моделювання. Таким чином, функціональна схема слугує основою для подальших розділів, де проводиться формалізація протоколів і реалізація програмного емулятора.

2.2 Принципова схема емулятора та організація передавання сигналів

Принципова схема дослідного емулятора (рис. 2.2) відображає взаємодію апаратних та програмних компонентів системи, які забезпечують повноцінне відтворення циклу події доступу - від моменту прикладання RFID-картки користувачем до запису телеметрії у базу даних. Обрана архітектура поєднує фізичний рівень контролера, сигнальні інтерфейси периферії та серверну інфраструктуру, що дозволяє створити узгоджене середовище для тестування, налагодження та валідації логіки контролю доступу.

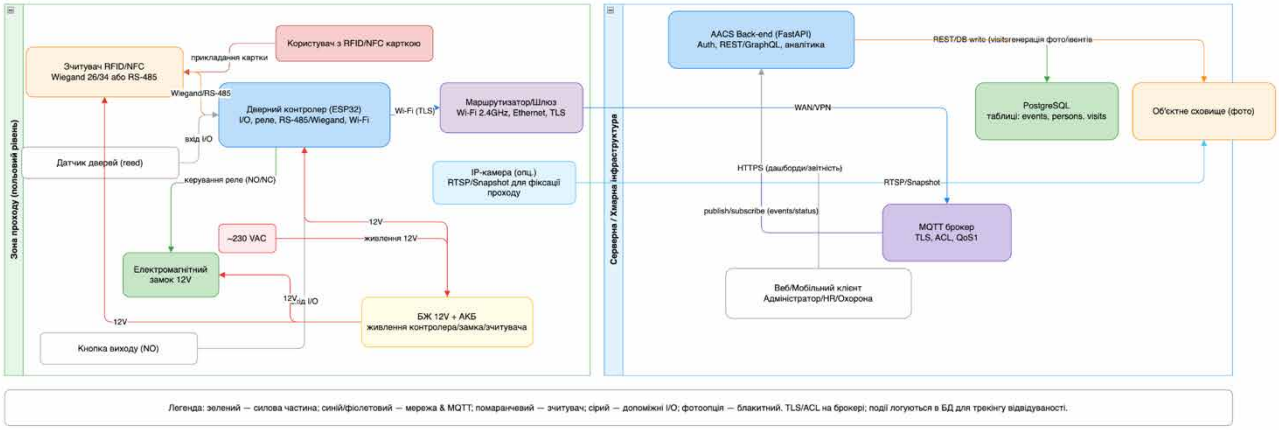

Рис. 2.2 – Принципова схема дослідного стенду та організація передавання сигналів

Побудова схеми базується на ідеї нормалізації інформаційних і силових каналів, що уможливлює уніфікацію форматів даних і мінімізацію кількості протоколів взаємодії між рівнями. Основою є контролер ESP32, який реалізує обробку подій на рівні поля та забезпечує комунікацію із сервером через Wi-Fi із захистом TLS. Сигнали зчитувача RFID/NFC, датчиків дверей, кнопок виходу та електромагнітних замків передаються до контролера через стандартизовані

інтерфейси - Wiegand, RS-485 та GPIO, що дозволяє використовувати як реальні, так і емуляційні модулі у межах тестового стенду.

На серверному боці реалізовано централізований обмін через MQTT-брокер із підтримкою ACL-списків і QoS1, а також аналітичну підсистему на базі PostgreSQL і FastAPI-сервісу, що забезпечує REST/GraphQL-взаємодію. Такий розподіл дозволяє розділити функції маршрутизації подій, авторизації, журналювання та аналітики, уникаючи конфліктів між потоками керування й даних. Передавання сигналів організовано за змішаною схемою - апаратна телеметрія надходить до брокера у вигляді MQTT-повідомлень, а результати обробки повертаються до вузлів у формі команд із підтвердженням отримання.

У таблиці 2.2 наведено систематизацію основних каналів та їхнього функціонального призначення, що відображає логіку інтеграції між польовим рівнем, серверною частиною та сховищем даних.

Таблиця 2.2

Канали взаємодії компонентів системи емулятора

Рівень взаємодії	Канал / Протокол	Призначення	Тип сигналу
Польовий рівень (Edge)	Wiegand / RS-485 / GPIO	Обмін між зчитувачем, контролером і замком	Дискретний / аналоговий
Комунікаційна мережа	Wi-Fi 2.4 GHz (TLS) / MQTT	Транспорт подій і команд	Цифровий (шифрований)
Серверна логіка	FastAPI / REST / GraphQL	Обробка запитів, аналітика, авторизація	Логічний
Сховище даних	PostgreSQL / Object Storage	Збереження подій, фотографій, відвідувань	Дані / медіа

Застосування принципу нормалізації потоків у цій схемі забезпечує гнучкість масштабування та спрощує заміну апаратних компонентів без зміни логіки програмної взаємодії. Така архітектура дозволяє проводити експерименти з реальними затримками, контролювати достовірність передавання сигналів і створює базу для подальшого моделювання поведінки системи у розподілених середовищах. У результаті отримано уніфіковану модель емулятора, здатну

відтворювати роботу повномасштабної АСКД із високим ступенем точності та відтворюваності результатів.

2.3 Електрична та монтажна схема дослідного стенду емулятора

Побудова електричної та монтажної схеми дослідного стенду (рис. 2.3–2.8) є ключовим етапом у створенні фізичної моделі системи контролю доступу з функцією трекінгу відвідуваності, що забезпечує перевірку працездатності апаратних компонентів, взаємодію вузлів та стабільність електроживлення. Зображені фрагменти схем демонструють реалізацію концепції нормалізованого живлення, уніфікованих інтерфейсів і безпечної комутації сигналів, яка була покладена в основу всієї архітектури системи.

На рисунку 2.3 подано приклад основного вузла керування – контролера ESP32, який виступає ядром системи, здійснюючи комунікацію з периферією та обробку подій на рівні стенду. Його використання зумовлене поєднанням бездротових інтерфейсів (Wi-Fi, BLE), підтримкою GPIO і низьким енергоспоживанням, що робить можливим моделювання як реальних, так і емуляційних потоків даних.

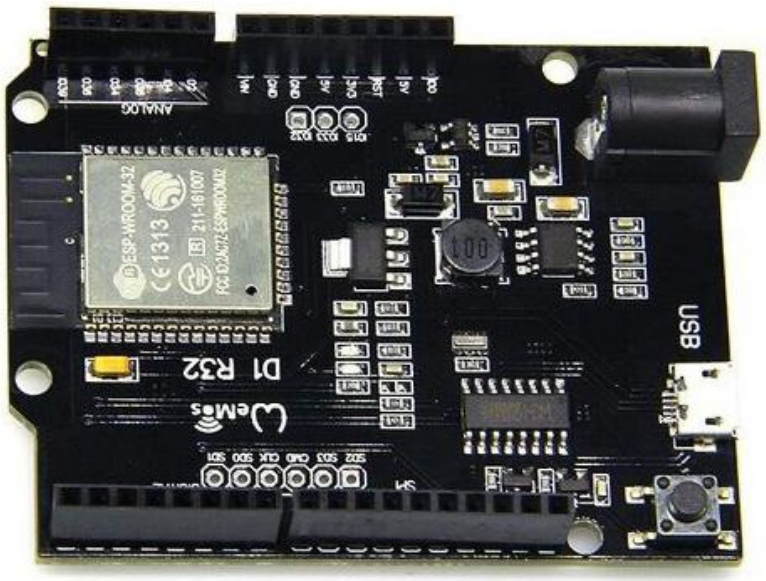

Рис. 2.3 – Контролер ESP32 як центральний елемент стенду

На рисунку 2.4 наведено модуль RFID-зчитувача MFRC522, який реалізує первинну ідентифікацію користувача. Використання цього модуля у стенді дозволяє перевіряти коректність обробки подій доступу, інтеграцію з Wiegand- або SPI-інтерфейсом та формування тестових подій для серверного ядра.

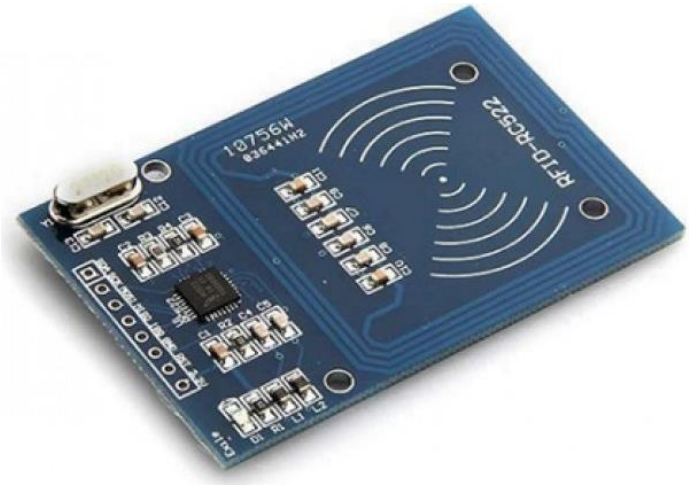

Рис. 2.4 – Модуль RFID/NFC-зчитувача MFRC522 для ідентифікації користувачів

Рисунок 2.5 ілюструє електромагнітний замок 12 В, який застосовується для імітації фізичного акту доступу. Він є частиною силової лінії системи, що дозволяє відпрацьовувати сценарії подачі та зняття напруги, перевіряти реакцію контролера на зміну станів реле та проводити стрес-тести із затримками сигналів.

Рис. 2.5 – Електромагнітний замок 12 В як виконавчий елемент системи

На рисунку 2.6 зображено датчик дверей (reed-сенсор), який забезпечує зворотний зв’язок для контролера щодо стану проходу. Його включення до схеми дозволяє фіксувати спроби несанкціонованого відкриття дверей або перевіряти логіку спрацьовування подій типу “open/close” у базі даних.

Рис. 2.6 – Датчик дверей типу reed для зворотного контролю стану проходу

На рисунку 2.7 показано блок живлення 12 В із функцією резервування (АКБ), який забезпечує живлення контролера, зчитувача та електромеханічних вузлів. Його вибір зумовлений потребою у стабільності напруги при переходах між навантаженнями, захистом від перенапруг (OVP/OCP) та можливістю моделювання аварійних режимів при втраті основного джерела живлення.

Рис. 2.7 – Блок живлення 12 В з АКБ-резервом для забезпечення безперервності роботи стенду

Електрична схема з’єднань між цими компонентами (рис. 2.8) демонструє узгодження рівнів логіки, захист від саміндукції (діод D1, TVS-захист) та реалізацію єдиної шини заземлення для всіх пристроїв. Схема побудована за принципом радіальної топології, де кожен пристрій має власну лінію підключення, що зменшує падіння напруги та електричні шуми.

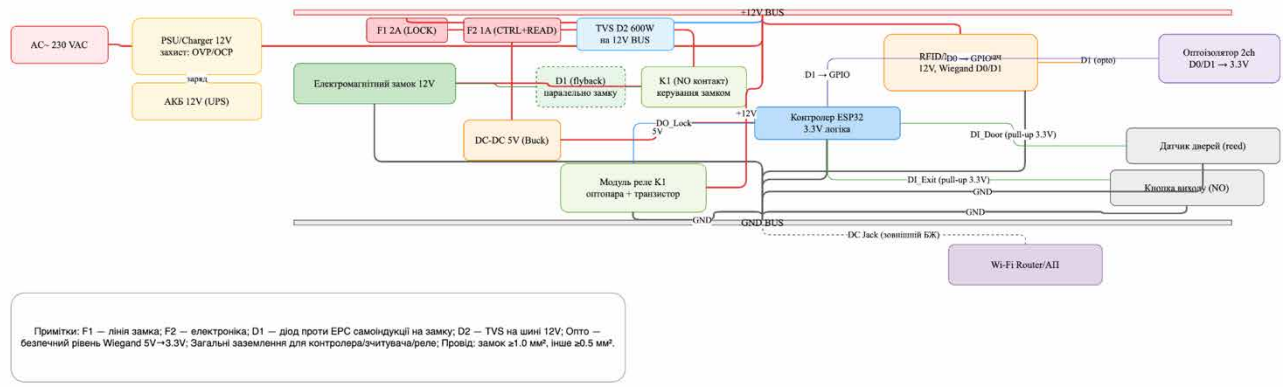

Рис. 2.8 – Електрична схема підключення компонентів дослідного стенду

Для систематизації характеристик апаратних компонентів і визначення їх ролі в стенді у таблиці 2.3 подано узагальнений перелік основних пристроїв, що забезпечують роботу електричної частини.

Таблиця 2.3

Основні апаратні компоненти дослідного стенду емулятора

№	Пристрій	Основна функція	Номінальні параметри
1	Контролер ESP32	Центральна логіка обробки подій, комунікація з сервером	3.3 В логіка, Wi-Fi/BLE, GPIO
2	Зчитувач RFID/NFC MFRC522	Ідентифікація користувача	13.56 МГц, SPI/Wiegand
3	Електромагнітний замок 12 В	Виконавчий механізм доступу	12 В DC, 450 мА
4	Датчик дверей (reed)	Контроль стану проходу	NO/NC, 3.3–5 В
5	Блок живлення 12 В + АКБ	Стабілізоване живлення та резервування	12 В 2 А, UPS-режим

На заключному етапі розроблено монтажну схему (рис. 2.9), яка визначає просторове розташування пристроїв, траси прокладки кабелів та висоти монтажу. Прийняті висоти - 1.0 м для кнопки виходу, 1.4 м для зчитувача, 2.1 м для замка та 2.2 м для контролера з блоком живлення - відповідають ергономічним і безпечним нормам експлуатації систем АСКД.

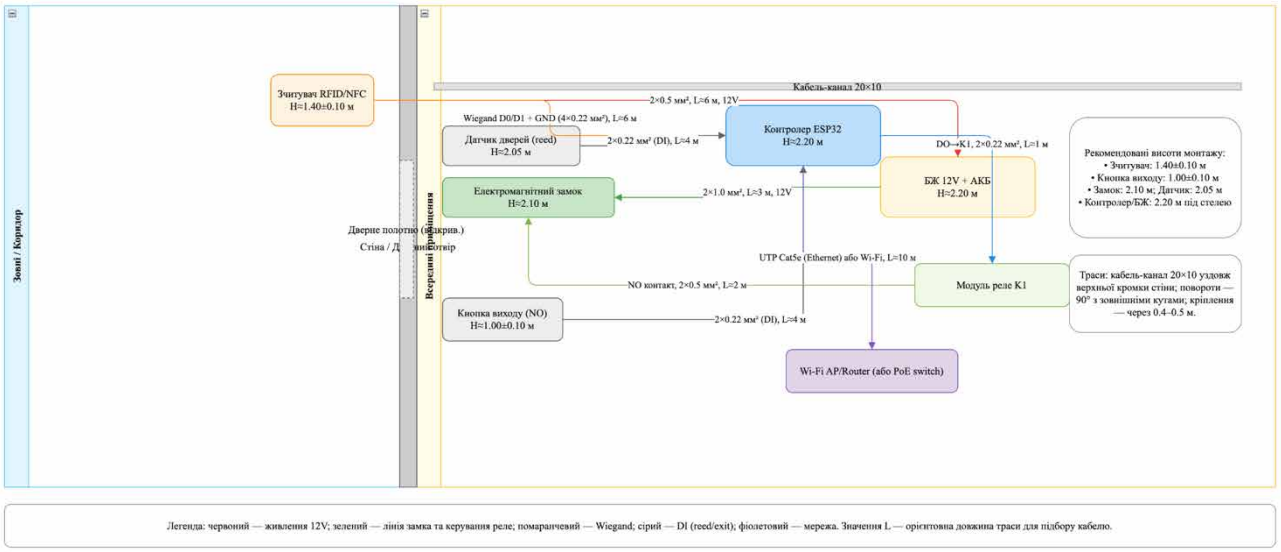

Рис. 2.9 – Монтажна схема розміщення компонентів дослідного стенду

Застосована архітектура електричної та монтажної частин забезпечує стабільність подачі живлення, безпечну комутацію сигналів, а також простоту масштабування й сервісного обслуговування. Використання нормалізованих інтерфейсів (GPIO, SPI, RS-485, Wiegand), уніфікованих напруг живлення (5 В/12 В) та ізольованих контурів керування підвищує надійність системи та дозволяє відтворювати у лабораторних умовах повний цикл роботи промислової АСКД.

2.4 Передумови створення програмного емулятора та вибір технологічного стеку

Необхідність створення програмного емулятора для системи контролю доступу з функцією трекінгу відвідуваності зумовлена потребою у тестуванні логіки серверної обробки подій без використання повного апаратного комплексу. Реальні АСКД містять багаторівневі взаємодії між мікроконтролерами, сенсорами, мережевими шлюзами та серверними службами, тому розробка стенду з програмною імітацією потоків подій дозволяє верифікувати архітектурні рішення, протестувати алгоритми маршрутизації та оцінити стабільність протоколів у різних сценаріях навантаження. Такий підхід знижує вартість і ризики на етапі впровадження, забезпечує контрольоване

середовище для дослідження ефективності обробки даних, журналювання й аналітики.

У межах дослідження реалізація програмного емулятора базується на принципі архітектурної еквівалентності - тобто логіка обробки подій, структура повідомлень і протоколи залишаються ідентичними до реальної системи, але виконуються в ізольованому середовищі. Це дозволяє забезпечити сумісність між модулем емулятора та виробничим сервером, використовуючи однакові формати MQTT-повідомлень і API-запитів. Застосування віртуалізованого середовища дає змогу імітувати багатопоточність, перевіряти поведінку системи при затримках або втраті з'єднання, а також відстежувати ефективність обробки у реальному часі.

Для реалізації системи було обрано стек технологій, який забезпечує оптимальне співвідношення між продуктивністю, масштабованістю, зручністю інтеграції та відповідністю стандартам безпеки. Основні компоненти програмного середовища наведено в таблиці 2.4.

Таблиця 2.4

Вибір технологічного стеку для реалізації програмного емулятора

№	Компонент	Призначення	Основні характеристики
1	Python 3.10+ (Flask / FastAPI)	Реалізація серверної логіки, обробка подій, REST / GraphQL API	Асинхронна обробка, модульність, висока продуктивність
2	MQTT (Eclipse Mosquitto)	Передача подій між емулятором та сервером	QoS1, TLS, ACL, підтримка багатопоточності
3	SQLite / PostgreSQL	Локальне або масштабоване зберігання історії подій	ACID-сумісність, реплікація, JSON-поля
4	Docker	Контейнеризація середовища та ізоляція процесів	Кросплатформність, швидке розгортання
5	WebSocket / REST / JSON	Реалізація двосторонньої взаємодії та моніторингу	Уніфікований формат обміну даними
6	Grafana / BI Dashboard	Візуалізація даних і контроль метрик	Підтримка MQTT Input та PostgreSQL Sources

Вибір стеку базується на критеріях відмовостійкості, відкритості стандартів і можливості масштабування без зміни базових протоколів. Python забезпечує високу швидкість розробки і сумісність з аналітичними бібліотеками, MQTT гарантує мінімальні затримки передачі подій і підтримує централізоване журналювання, а PostgreSQL створює основу для аналітичних запитів та історичних звітів. Контейнеризація в Docker спрощує процес розгортання, дозволяє ізолювати серверну частину від апаратної платформи та забезпечує однаковість середовищ у межах дослідів.

Застосування програмного емулятора з описаним стеком технологій дозволяє створити відтворюване, масштабоване та безпечне середовище для тестування логіки контролю доступу, перевірки стійкості протоколів MQTT/HTTP, аналітичної обробки подій і побудови звітності. Розроблена структура створює основу для подальшого моделювання потоків даних і інтеграції інтелектуальних алгоритмів аналізу IoT-подій у наступному розділі роботи.

2.5 Формалізація специфікації повідомлень і тем MQTT

Для забезпечення уніфікованої взаємодії між елементами дослідного стенду та серверною частиною системи розроблено специфікацію повідомлень і структуру тем протоколу MQTT, що визначає принципи публікації, маршрутизації та підписки на події. Використання MQTT як основного транспортного механізму обумовлене його асинхронністю, ефективним управлінням затримками і підтримкою різних рівнів якості доставки (QoS). Це дозволяє досягнути стійкої передачі IoT-подій навіть за умов часткової втрати з’єднання або нестабільності каналу.

На рисунку 2.10 подано узагальнену структурну схему потоків обміну, що демонструє логіку публікацій та підписок між емулятором ESP32, брокером MQTT, сервером обробки подій (FastAPI) та інтерфейсом аналітичної

візуалізації. В основі побудови лежить принцип нормалізації топіків, коли усі повідомлення мають спільну ієрархічну структуру aacs/<site>/<door>/*, що забезпечує єдність адресації та спрощує маршрутизацію потоків даних у багатозонному середовищі.

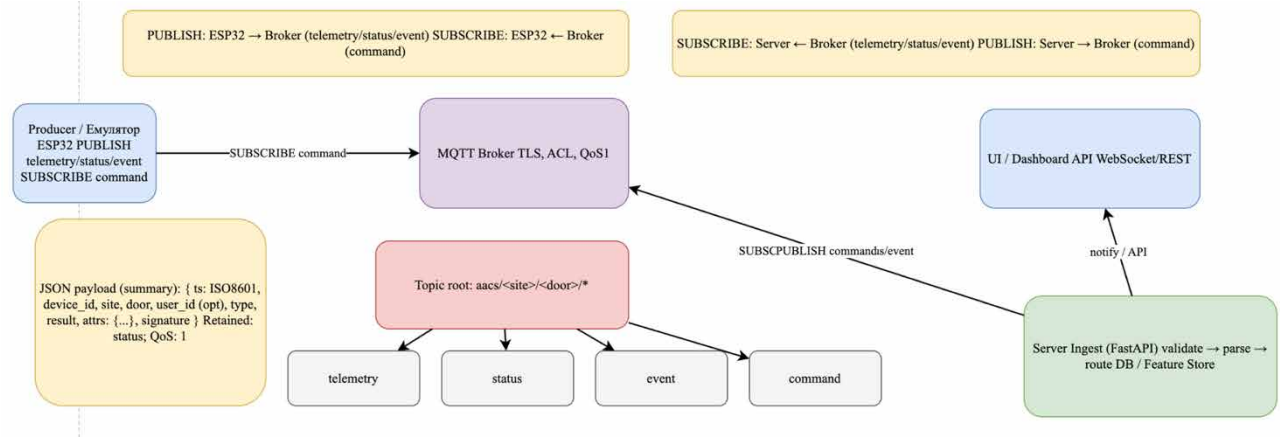

Рис. 2.10 – Структура тем та потоків повідомлень MQTT у системі емулятора

Основні типи повідомлень визначені для трьох класів функціональності:

– telemetry - періодичні дані стану контролера (напруга живлення, RSSI, стан реле);

– status - сигнали зміни стану дверей або виконавчих пристроїв;

– event - події взаємодії користувача (ідентифікація, помилка, доступ дозволено/заборонено);

– command - керуючі повідомлення, що передаються сервером до вузлів (оновлення прошивки, тест, зміна конфігурації).

Усі повідомлення формуються у форматі JSON, що забезпечує розширюваність структури та зручність у парсингу на боці серверної логіки. Формат включає часову позначку ISO 8601, унікальний ідентифікатор пристрою, параметри події, атрибути середовища та цифровий підпис для перевірки достовірності. У таблиці 2.5 наведено узагальнену структуру повідомлення та призначення його основних полів.

Таблиця 2.5

Формалізована структура MQTT-повідомлення

Поле	Тип даних	Призначення
ts	string (ISO 8601)	Час фіксації події
device_id	string	Унікальний ідентифікатор вузла ESP32
site / door	string	Локація та точка контролю
user_id	string (опц.)	Ідентифікатор користувача або картки
type	string	Тип події (telemetry/status/event/command)
result	string	Результат виконання або стан
attrs	object	Додаткові параметри (температура, напруга, RSSI тощо)
signature	string	Цифровий підпис для верифікації повідомлення

Застосування стандартизованої схеми тем MQTT з чіткою ієрархією і типізацією повідомлень спрощує логіку аналітичної обробки даних, підвищує рівень узгодженості між сервером та периферією і створює основу для розширення функціональності без зміни архітектури транспортного шару. Такий підхід забезпечує нормалізацію інформаційних потоків у межах системи, що особливо важливо при переході до промислової експлуатації та інтеграції з розподіленими сховищами подій.

У результаті розроблена формалізація дозволяє здійснювати централізоване керування темами, контролювати доступ через ACL-списки брокера, відслідковувати коректність формату повідомлень та гарантувати їх доставку із заданою якістю обслуговування (QoS = 1). Це формує надійну основу для подальшої інтеграції алгоритмів інтелектуального аналізу потоків IoT-подій у межах серверного ядра системи.

2.6 Висновки до другого розділу

У другому розділі виконано повний цикл проєктування дослідного стенду та програмного емулятора серверної обробки подій у системі з IoT-пристроями,

що дозволило сформувати цілісну архітектурну, апаратну та програмну модель системи. Розроблена функціональна та принципова схема забезпечує відтворюваність процесів контролю доступу, включно з передачею, маршрутизацією та обробкою сигналів на рівнях периферії, мережі та серверної логіки. Запропонована структуризація каналів обміну та нормалізація топології комунікаційних потоків дала змогу реалізувати єдину логіку обробки подій як у фізичному, так і в емуляційному середовищі.

Розроблені електрична та монтажна схеми підтвердили технічну життєздатність рішення, забезпечивши узгодження між силовими й логічними контурами системи. Використання у складі стенду типових вузлів - контролера ESP32, зчитувача RFID/NFC, електромагнітного замка, датчика стану дверей і резервованого блоку живлення - дозволило створити уніфіковану апаратну базу, придатну для відтворення типових сценаріїв реальної АСКД. Особливу увагу приділено ізоляції ліній живлення, стандартизації інтерфейсів (Wiegand, RS-485, GPIO) та захисту від перенапруг, що забезпечує безпечну роботу під час експериментів.

У частині програмного забезпечення обґрунтовано вибір стеку технологій Python / FastAPI, MQTT, PostgreSQLта Docker, який забезпечує асинхронну обробку подій, надійність передачі повідомлень і масштабованість серверної логіки. Формалізовано специфікацію MQTT-повідомлень із визначенням типів топіків, структури JSON-пакета та політик доступу ACL, що уможливлює централізоване керування потоками телеметрії, статусів і команд.

Таким чином, у межах другого розділу сформовано технічно й методично обґрунтовану основу для подальшого моделювання алгоритмів інтелектуального аналізу IoT-потоків, яке буде реалізовано в наступному розділі. Отримані результати засвідчують узгодженість апаратних, мережевих і програмних рішень, а також готовність системи до випробувань у реальному та емуляційному середовищах.

3 ПРОГРАМНА ІМПЛЕМЕНТАЦІЯ СИСТЕМИ КОНТРОЛЮ ДОСТУПУ З ФУНКЦІЄЮ ТРЕКІНГУ ВІДВІНУВАНОСТІ

3.1 Складання пристрою у Tinkercad, вибір компонентів системи

На даному етапі було реалізовано віртуальне складання макету системи контролю доступу у середовищі Tinkercad Circuits, що дозволило апробувати архітектурну концепцію пристрою, відпрацювати взаємодію між його основними модулями та перевірити алгоритм реакції на події до переходу до програмної емуляції. Використання Tinkercad дало змогу створити цілісний прототип без фізичного монтажу, що важливо для попередньої нормалізації електричних сигналів і тестування логічних сценаріїв роботи.

На рисунку 3.1 наведено загальний вигляд змодельованого пристрою, який відтворює базову логіку системи контролю присутності. Основу конструкції становить контролер Arduino UNO R3, який виконує функції оброблення подій і взаємодії з периферією. До нього підключено LCD-дисплей 16×2 для виведення текстових повідомлень, дві кнопки (pushbutton) для генерації подій користувачів, потенціометр 10 кΩ для регулювання контрастності дисплея, світлодіод (LED) з обмежувальним резистором 220 Ω для індикації активності системи та макетну плату для реалізації комутації. Така конфігурація дозволяє відтворити подієвий цикл: виявлення натискання, оброблення сигналу, оновлення інформації на дисплеї та візуальне підтвердження виконаної дії.

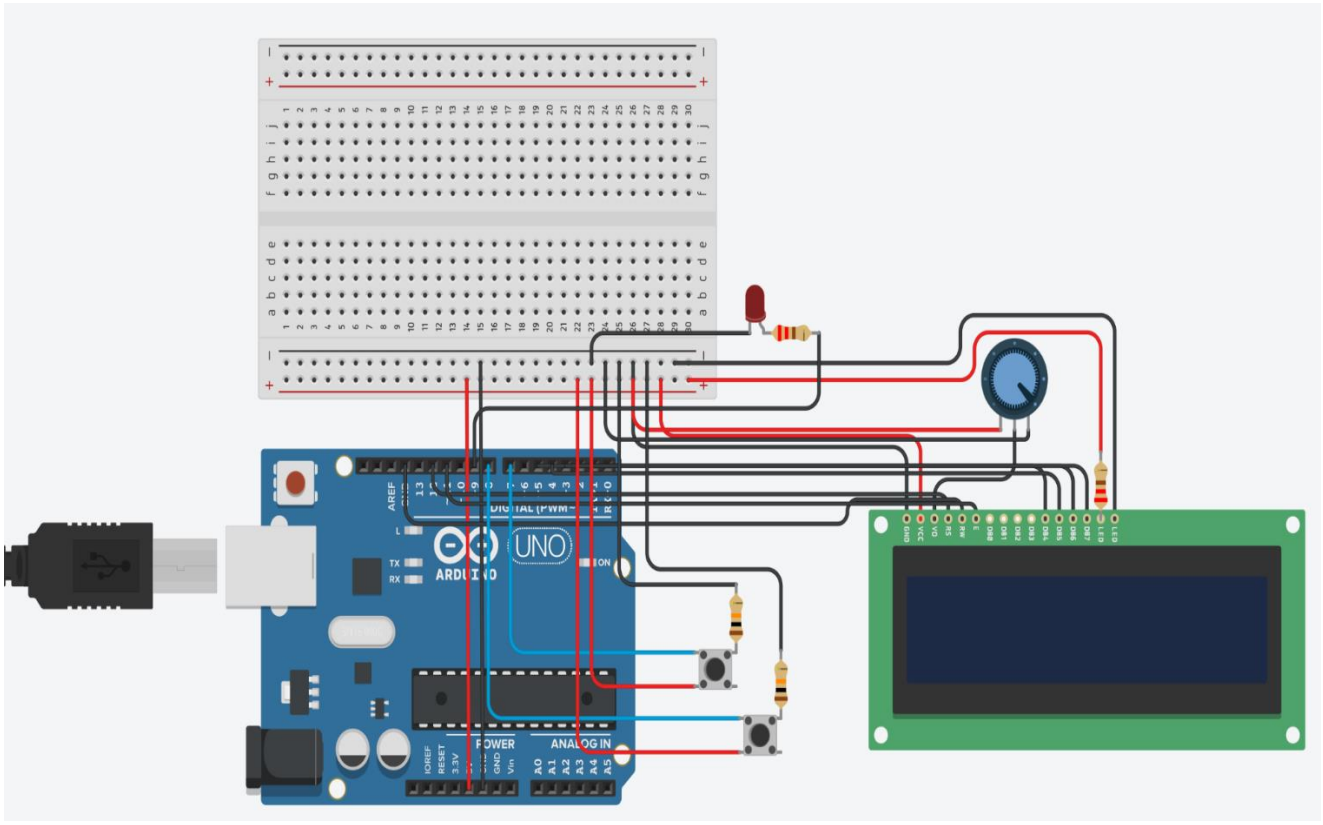

Рис. 3.1 – Зібраний макет системи у середовищі Tinkercad

На рисунку 3.2 представлено внутрішнє з’єднання компонентів моделі, реалізоване відповідно до інтерфейсу HD44780. Всі сигнальні лінії LCD підключено до цифрових пінів D2–D7 контролера Arduino, кнопки підключено до пінів D7 і D8 із підтягуванням через резистори 10 кΩ, а LED з’єднано з піком D9 через послідовний резистор 220 Ω. Потенціометр підключено до входу VO дисплея для регулювання контрастності. Такий підхід забезпечує стабільну роботу макету й коректну реакцію на зміни станів без паразитних перешкод.

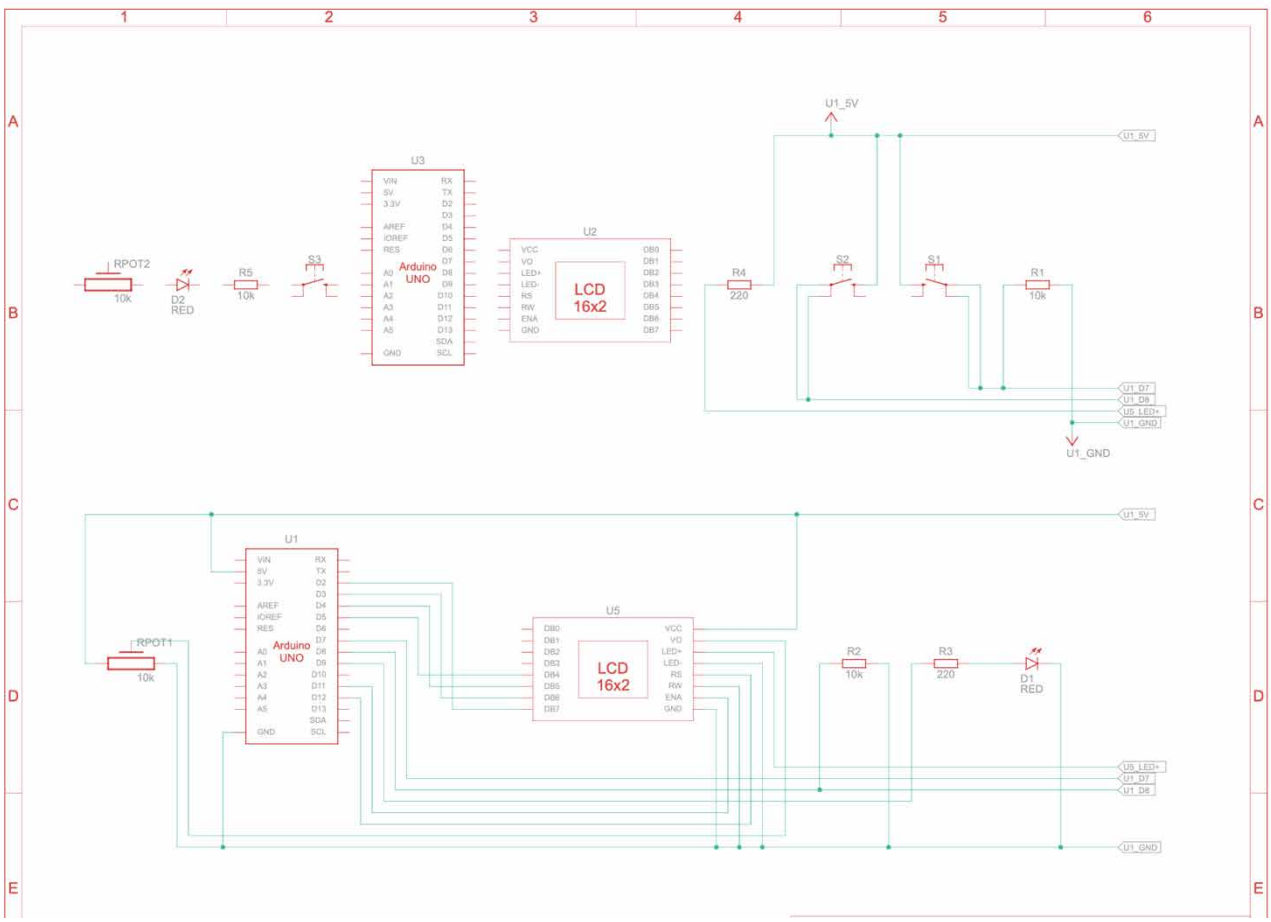

Рис. 3.2 – Електронна модель підключень у середовищі Tinkercad

На рисунку 3.3 показано програмний код, розроблений у середовищі Arduino IDE, який визначає поведінку пристрою. Програма ініціалізує бібліотеку LiquidCrystal, налаштовує порти введення-виведення та реалізує оброблення подій натискання кнопок. Після активації кнопки контролер виводить на дисплей ідентифікатор користувача (наприклад, Student A або Student B), подає короткий сигнал на світлодіод і повертає систему у стан Ready..., що моделює основний цикл подієвого обміну даними у системі контролю доступу.

```
#include <LiquidCrystal.h>

// LCD pins: RS, E, D4, D5, D6, D7
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int studentA = 7;
int studentB = 8;
int buzzer   = 9;

void setup() {
  pinMode(studentA, INPUT);
  pinMode(studentB, INPUT);
  pinMode(buzzer, OUTPUT);

  lcd.begin(16, 2);
  lcd.print("Attendance Sys");
  lcd.setCursor(0,1);
  lcd.print("Ready...");
}

void loop() {
  if (digitalRead(studentA) == HIGH) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Student A");
    lcd.setCursor(0,1);
    lcd.print("Present");
    tone(buzzer, 1000, 200);
    delay(1000);
    lcd.clear();
    lcd.print("Ready...");
  }

  if (digitalRead(studentB) == HIGH) {
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Student B");
    lcd.setCursor(0,1);
    lcd.print("Present");
    tone(buzzer, 1200, 200);
    delay(1000);
    lcd.clear();
    lcd.print("Ready...");
  }
}
```

Рис. 3.3 – Фрагмент програмного коду моделі пристрою у середовищі Arduino IDE

На рисунку 3.4 представлено роботу пристрою у момент події**:** на дисплеї відображається повідомлення “Student A – Present”, а світлодіод сигналізує про успішну ідентифікацію користувача. Це свідчить про правильність реалізації логічних переходів між станами, узгодженість програмної та апаратної частини, а також стабільність системи в умовах безперервного циклу читання сигналів.

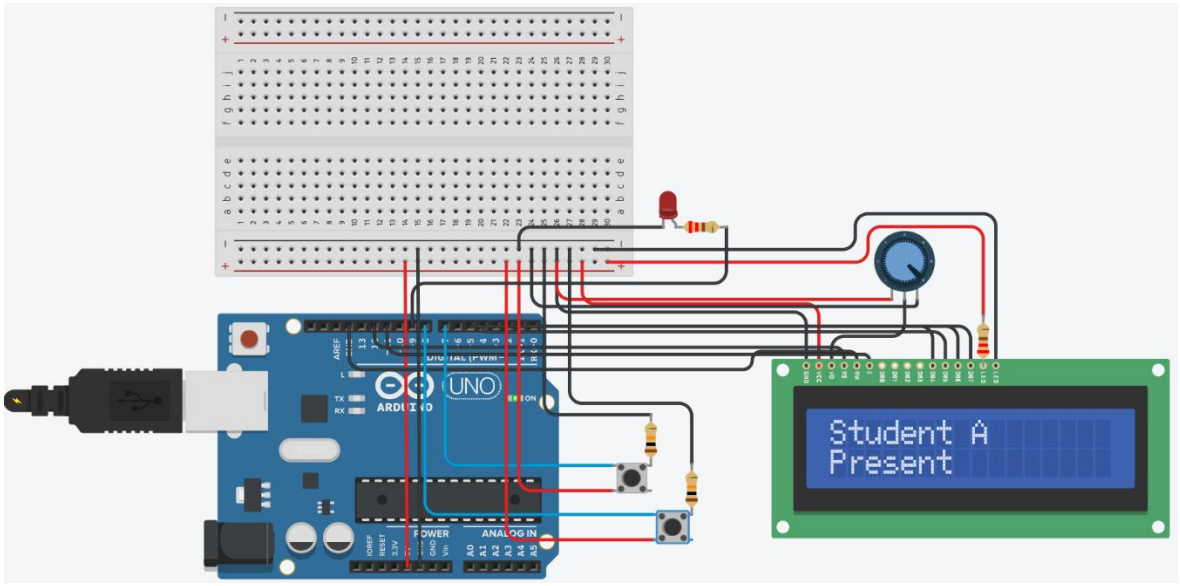

Рис. 3.4 – Візуалізація активного стану пристрою під час події “Student A”

На рисунку 3.5 показано режим очікування системи, коли події відсутні, і дисплей відображає повідомлення “Ready…”. Такий стан демонструє реалізацію базового циклу моніторингу подій і підтримання системи у готовності, що відповідає принципам подієво-орієнтованої архітектури.

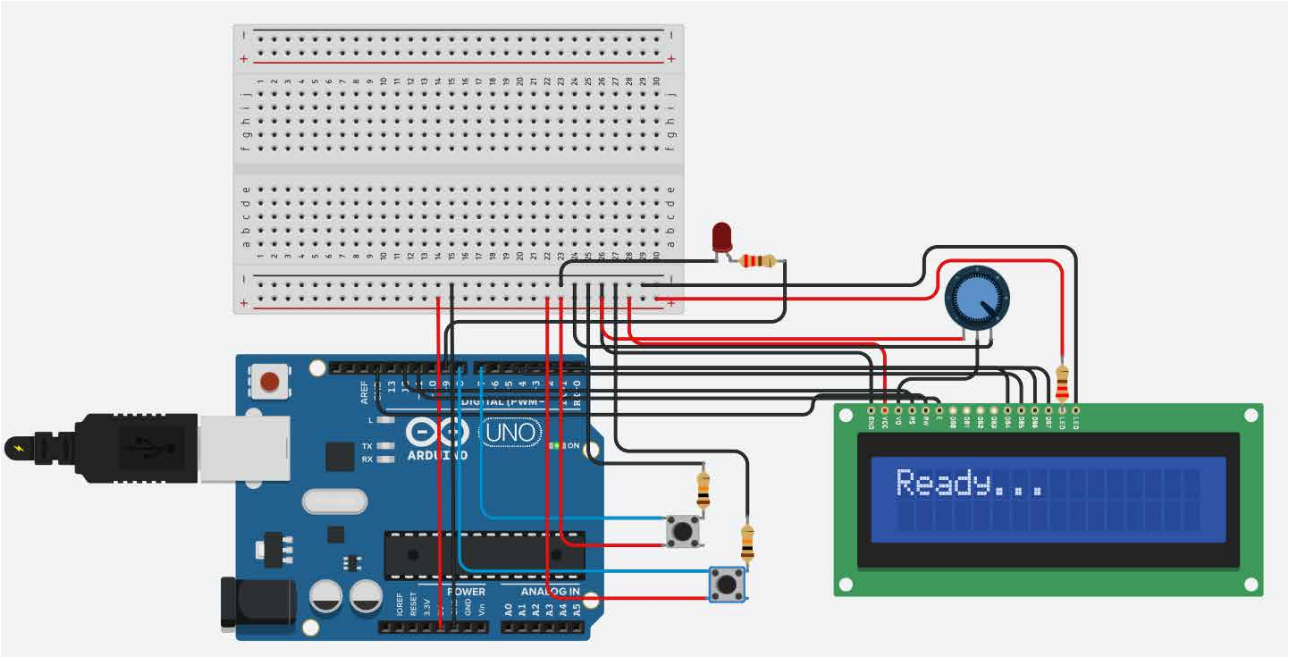

Рис. 3.5 – Режим очікування пристрою (система у стані готовності)

На рисунку 3.6 наведено перелік використаних у моделі компонентів, що формують апаратну основу дослідного стенду. Кожен елемент виконує окрему функцію у контурі взаємодії: контролер – логіку, кнопки – ініціацію подій, дисплей – візуалізацію, а резистори – нормалізацію рівнів напруги.

Name	Quantity	Component
U1	1	Arduino Uno R3
S1 S2	2	Pushbutton
U5	1	LCD 16 x 2
Rpot1	1	10 kΩ Potentiometer
R4 R3	2	220 Ω Resistor
R1 R2	2	10 kΩ Resistor
D1	1	Red LED

Рис. 3.6 – Таблиця основних компонентів дослідного макету

Склад і характеристика компонентів віртуального стенду представлені у таблиці 3.1.

Таблиця 3.1

Склад і характеристика компонентів віртуального стенду

Позначення	Кількість	Компонент
U1	1	Arduino UNO R3 – мікроконтролер для оброблення подій
S1, S2	2	Pushbutton – кнопки ініціації подій користувачів
U5	1	LCD 16×2 – символьний дисплей для виведення інформації
Rpot1	1	Потенціометр 10 кΩ – регулювання контрастності дисплея
R4, R3	2	Резистор 220 Ω – обмеження струму для LED
R1, R2	2	Резистор 10 кΩ – підтягування входів кнопок
D1	1	Red LED – індикатор події та стану системи

Отримана модель у середовищі Tinkercad Circuits дозволила протестувати електричну сумісність елементів, оптимізувати схему живлення та підтвердити правильність архітектурної побудови. Такий макет виконує роль проміжного етапу перед створенням емулятора серверної логіки, оскільки забезпечує достовірну симуляцію поведінки фізичного пристрою, нормалізацію рівнів сигналів і перевірку алгоритму взаємодії між апаратними та програмними компонентами системи.

3.2 Моделювання предметної області, діаграма компонентів, діаграма пакетів

Моделювання предметної області автоматизованої системи контролю доступу з функцією трекінгу відвідуваності є необхідним етапом для узгодження вимог, структури та логіки взаємодії між апаратними, програмними і аналітичними компонентами. На основі аналізу попередніх розділів побудовано три ключові UML-моделі: діаграму прецедентів, діаграму компонентів і діаграму пакетів, які в сукупності описують функціональну, архітектурну та структурну організацію системи.

На рисунку 3.7 подано діаграму прецедентів, яка відображає взаємодію між основними акторами (користувач, охорона/адміністратор, HR-менеджер, системний адміністратор та зовнішні HRM/ERP/BI-системи) і прецедентами системи. У центрі предметної області знаходиться процес проходження ідентифікації через RFID/NFC/QR-механізми та перевірка прав доступу за моделями RBAC/ABAC. Усі події фіксуються у базі даних з подальшим формуванням журналів і звітів відвідуваності. Передбачено сценарії сповіщення про аномалії у разі виявлення відхилень від політики безпеки. Інтеграція із зовнішніми HRM/ERP-системами забезпечує автоматичну синхронізацію користувачів та політик доступу, що підвищує узгодженість бізнес-процесів.

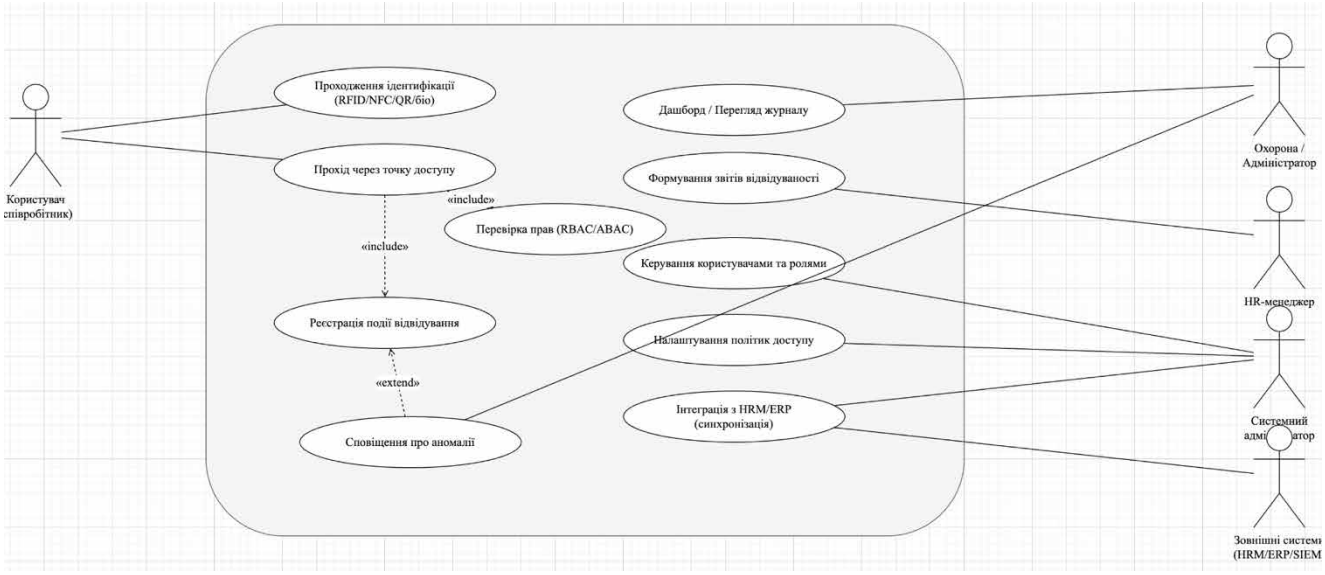

Рис. 3.7 – Діаграма прецедентів системи контролю доступу з трекінгом відвідуваності

На рисунку 3.8 наведено діаграму компонентів, яка демонструє розподіл системи за технологічними рівнями: Edge/Стенд, Мережа/MQTT, Серверна логіка, Сховище/Аналітика та Клієнти/Інтеграції. На рівні Edge функціонують RFID/NFC-зчитувач, контролер ESP32 та Wi-Fi-маршрутизатор, що формують події телеметрії й передають їх через MQTT-брокер. На серверній стороні реалізовано шлюз FastAPI Ingest Gateway, політичний сервіс AuthZ з RBAC/ABAC-перевірками, службу сповіщень Notification Service та UI Dashboard API для відображення даних. Сховище подій PostgreSQL виконує роль

бази журналів, а модуль Feature Store та BI-аналітика здійснюють формування метрик відвідуваності. Клієнтський рівень представлено веб/мобільним інтерфейсом для адміністратора, HR-відділу та охорони, а також інтеграційними модулями з корпоративними HRM/ERP/SIEM-сервісами через REST/ETL-синхронізацію.

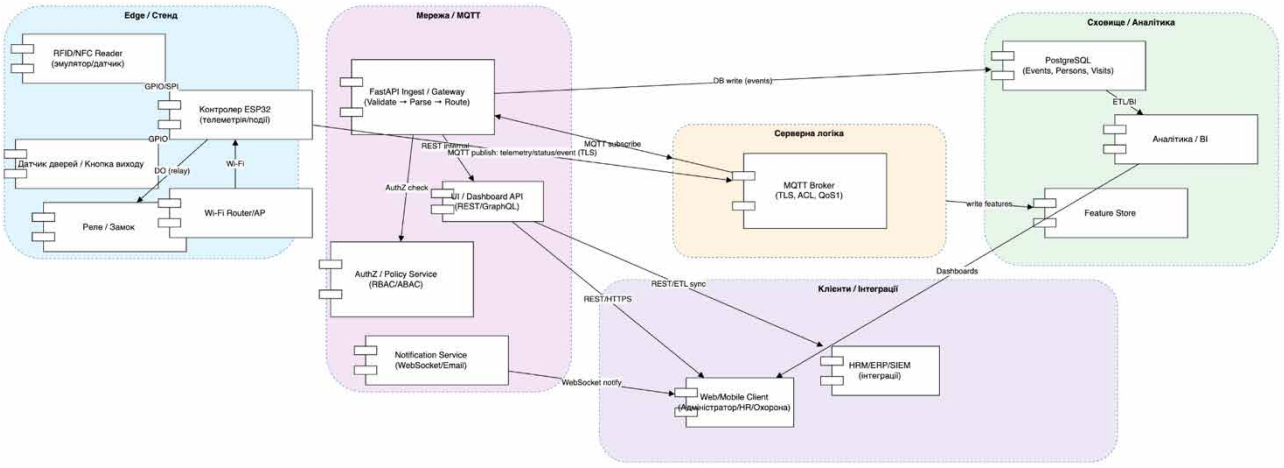

Рис. 3.8 – Діаграма компонентів архітектури системи

На рисунку 3.9 показано діаграму пакетів, яка формалізує структуру програмного забезпечення системи. Вона включає п’ять основних пакетів:

– edge – відповідає за пристрої польового рівня (device, gateway) та ініціацію подій;

– transport – реалізує канали MQTT та HTTP для комунікації;

– server – включає модулі ingest, api, authz та notify для обробки, авторизації і сповіщення

– data – охоплює сховище db, feature_store та schemas для структуризації та аналітики даних;

– client – представляє веб-інтерфейс (web, admin) для управління доступом і моніторингу.

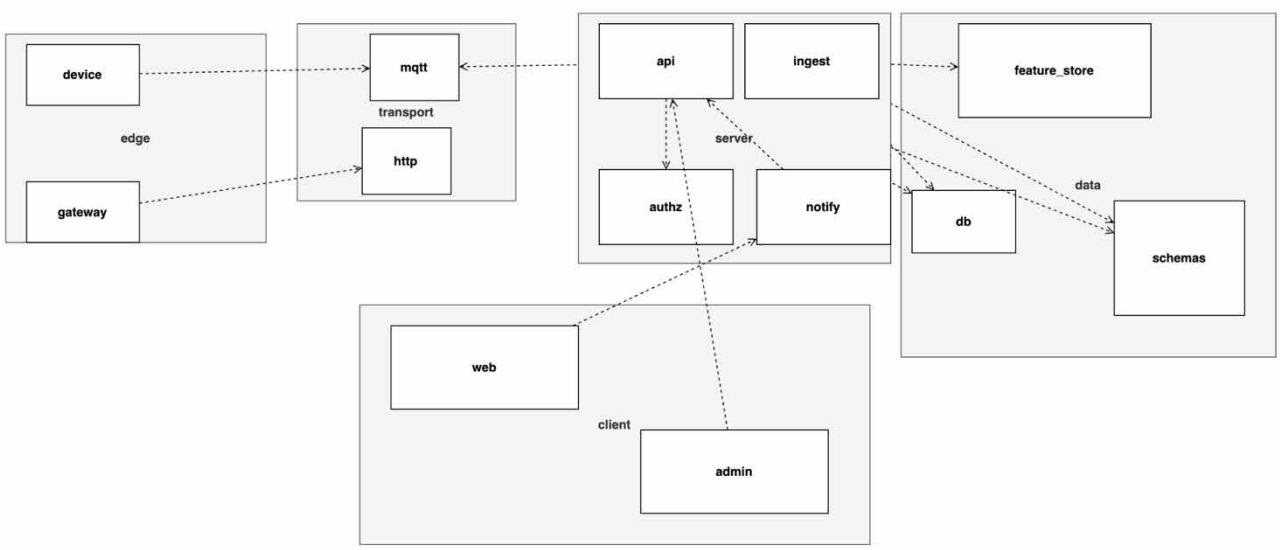

Рис. 3.9 – Діаграма пакетів програмного забезпечення системи

У сукупності наведені UML-моделі відображають комплексну архітектуру розробленої системи, де подієва модель обміну даними (MQTT/HTTP) інтегрує фізичний рівень контролерів ESP32 з аналітичним ядром на Python/FastAPI та сховищем PostgreSQL. Такий підхід забезпечує масштабованість, модульність і можливість розширення системи без зміни базових структур даних. Моделі підтверджують науково-технічну концепцію побудови подієво-орієнтованої АСКД нового покоління, орієнтованої на централізовану обробку подій, інтеграцію з корпоративними сервісами та підвищення рівня інформаційної безпеки

3.3 Обґрунтування вибору технологій реалізації емулятора

Для реалізації програмного емулятора серверної обробки подій в автоматизованій системі контролю доступу з функцією трекінгу відвідуваності було проведено детальний техніко-аналітичний відбір технологій, спрямований на забезпечення високої надійності, масштабованості та безпеки. Оскільки система моделює багатокомпонентне IoT-середовище з інтенсивним потоком подій, ключовими критеріями вибору виступали: швидкодія, сумісність із мікроконтролерною периферією (ESP32), підтримка асинхронних протоколів,

простота контейнеризації та відповідність стандартам ISO/IEC 30141 та ISO/IEC 25010 щодо якості програмних систем.

У таблиці 3.2 наведено узагальнення основних технологій і середовищ, використаних при побудові емулятора, з описом їх функціонального призначення та переваг.

Таблиця 3.2

Обґрунтування вибору технологій реалізації емулятора

№	Технологія / компонент	Призначення	Ключові переваги
1	Python 3.10 + FastAPI	Розроблення серверної логіки, REST/GraphQL API	Асинхронна обробка подій, висока продуктивність, мінімальні затримки, сумісність з бібліотеками аналітики
2	MQTT (Eclipse Mosquitto)	Основний транспорт подій між ESP32 та сервером	QoS 1, підтримка TLS, низький overhead, централізоване журналювання
3	SQLite / PostgreSQL	Збереження історії подій і телеметрії	ACID-сумісність, підтримка JSON, реплікація, зручність у BI-аналітиці
4	Docker	Контейнеризація середовища емулятора	Ізоляція процесів, швидке розгортання, кросплатформність
5	WebSocket / REST / JSON	Обмін повідомленнями між модулями	Реалізація двосторонньої взаємодії, уніфікований формат даних
6	Grafana / BI Dashboard	Візуалізація журналів і метрик	Інтерактивні дашборди, моніторинг QoS і KPI в реальному часі

Застосування описаного стеку технологій дає змогу створити відтворюване, гнучке та безпечне середовище для імітації подій контролю доступу. Python з FastAPI забезпечує асинхронну логіку серверної обробки, що дозволяє моделювати одночасну роботу кількох точок доступу, а MQTT підтримує гарантовану доставку повідомлень із мінімальними затримками. Використання SQLite та PostgreSQL створює можливість як локальних експериментів, так і масштабованого аналітичного зберігання. Docker забезпечує однаковість середовищ на всіх етапах тестування, а інтеграція Grafana надає засоби оперативного моніторингу продуктивності та достовірності реєстрації подій.

Отже, обраний технологічний стек є оптимальним для реалізації подієво-орієнтованого серверного емулятора, оскільки поєднує відкриті стандарти, модульну архітектуру, надійність обробки потоків даних і можливість інтеграції з аналітичними модулями системи.

3.4 Висновки до третього розділу

У третьому розділі виконано практичну реалізацію апаратно-програмного комплексу автоматизованої системи контролю доступу з функцією трекінгу відвідуваності та програмного емулятора серверної логіки. На основі попередньо розробленої архітектури було змодельовано роботу системи у середовищі Tinkercad Circuits, що дало змогу відтворити взаємодію між основними компонентами - контролером Arduino UNO R3 / ESP32, зчитувачем RFID/NFC, виконавчими елементами (реле, замок, LED-індикатор) і сенсорами контролю стану дверей. Реалізовано програмний код, який забезпечує генерацію подій, візуалізацію станів системи на дисплеї та індикацію реакцій у реальному часі, що підтвердило правильність побудови логічних зв’язків і відповідність алгоритмів подієво-орієнтованій архітектурі.

Проведене моделювання предметної області на основі UML-діаграм (прецедентів, компонентів, пакетів) дозволило формалізувати зв’язки між підсистемами, акторами та модулями системи. Діаграма прецедентів описала ключові сценарії взаємодії користувача, адміністратора, HR-менеджера та зовнішніх систем HRM/ERP; діаграма компонентів - розподіл логіки між рівнями Edge, Network, Server, Analytics; а діаграма пакетів - структуру програмних модулів (edge, server, data, client). Таке моделювання підтвердило узгодженість технічних і функціональних рішень та забезпечило основу для подальшої інтеграції аналітичних сервісів.

Обґрунтовано вибір технологій реалізації: стек Python 3.10 + FastAPI, MQTT (Eclipse Mosquitto), SQLite/PostgreSQL, Docker, Grafana/BI-Dashboard

було визначено як оптимальний з точки зору надійності, масштабованості, продуктивності та безпеки. Реалізована архітектура підтримує асинхронну обробку подій, шифрування TLS, гарантовану доставку повідомлень QoS1 і централізоване журналювання, що відповідає стандартам ISO/IEC 30141:2018 **та** ISO/IEC 25010:2011**.**

У результаті реалізації третього розділу сформовано функціональну основу системи - повноцінний стенд та серверний емулятор, здатні моделювати поведінку реальної АСКД у багатокористувацькому середовищі, відтворювати телеметрію, проводити тестування затримок, коректності авторизації та стабільності протоколів зв’язку. Отримані результати підтвердили працездатність обраних архітектурних рішень і створили передумови для переходу до наступного етапу - моделювання алгоритмів аналізу подій та формування аналітичних звітів у четвертому розділі.

4 ТЕСТУВАННЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ ЕМУЛЯЦІЙНОЇ СИСТЕМИ

4.1 План тестування програмних модулів та методика оцінювання результатів

Проведення тестування програмних модулів інтелектуальної системи контролю доступу з трекінгом відвідуваності є критично важливим етапом для перевірки коректності логіки обробки подій, стабільності MQTT-комунікацій, надійності збереження даних у базі та відповідності часових характеристик вимогам системи. План тестування охоплює послідовну перевірку функціональних, нефункціональних та інтеграційних аспектів роботи модулів, що забезпечує виявлення помилок, оцінку продуктивності й визначення рівня готовності системи до експлуатації. У таблиці 4.1 наведено структурований перелік тестових сценаріїв, орієнтований на валідацію ключових компонентів: обробки подій контролера, серверної логіки FastAPI, маршрутизації MQTT-повідомлень, роботи з базою даних PostgreSQL та генерації аналітичних звітів.

Таблиця 4.1 – План тестування програмних модулів системи

№	Тестовий сценарій	Модуль	Очікуваний результат
1	Надсилання ESP32 події ідентифікації	MQTT Ingest	Подія отримана, запис створено в БД
2	Перевірка роли користувача	AuthZ Service	Доступ дозволено / заборонено згідно RBAC
3	Запис події у PostgreSQL	DB Logger	Запис створюється, цілісність збережена
4	Затримка обробки події	Server Logic	Час ≤ 200 мс
5	Втрата MQTT-пакета	MQTT Broker	Повторна доставка QoS1
6	Генерація звіту відвідуваності	Analytics	Коректне формування за періодом
7	Аномальні події / подвійні зчитування	Event Filter	Фільтрація дублікатів, журналювання

Продовження таблиці 4.1

8	Відновлення після розриву з’єднання	Server–MQTT	Автоматична повторна підписка
9	Стрес-тест 1000 подій/хв	Server Logic	Стабільність, відсутність втрат
10	Перевірка API Dashboard	REST API	Дані доступні, коректна візуалізація

Методика оцінювання результатів базується на вимірюванні часових показників, точності журналювання, повноти доставки MQTT-повідомлень та відсутності критичних помилок у серверній логіці. Ключовими метриками є: час реакції серверного модуля, відсоток успішно оброблених подій, відповідність даних у базі фактичним сценаріям, стабільність роботи брокера при зростаючому навантаженні та коректність формування статистичних звітів. Оцінювання проводиться за принципом відповідності результатів наперед визначеним очікуванням, що дозволяє об’єктивно визначити якість роботи кожного компонента. Результати тестування формують основу для подальшого вдосконалення системи та підтверджують її готовність до реального розгортання в умовах багатопотокового подієвого середовища.

4.2 Тестування інтелектуальної системи контролю доступу з трекінгом відвідуваності

Тестування інтелектуальної системи контролю доступу з трекінгом відвідуваності було виконано в режимі високонавантаженої емуляції, що забезпечило можливість відтворення реальних сценаріїв роботи контролерів, MQTT-комунікацій, авторизаційних політик та серверної обробки подій. На Рисунку 4.1 представлено основний екран емулятора подієвого контуру, який демонструє динаміку надходження подій, затримки на кожному етапі обробки та внутрішній журнал роботи системи.

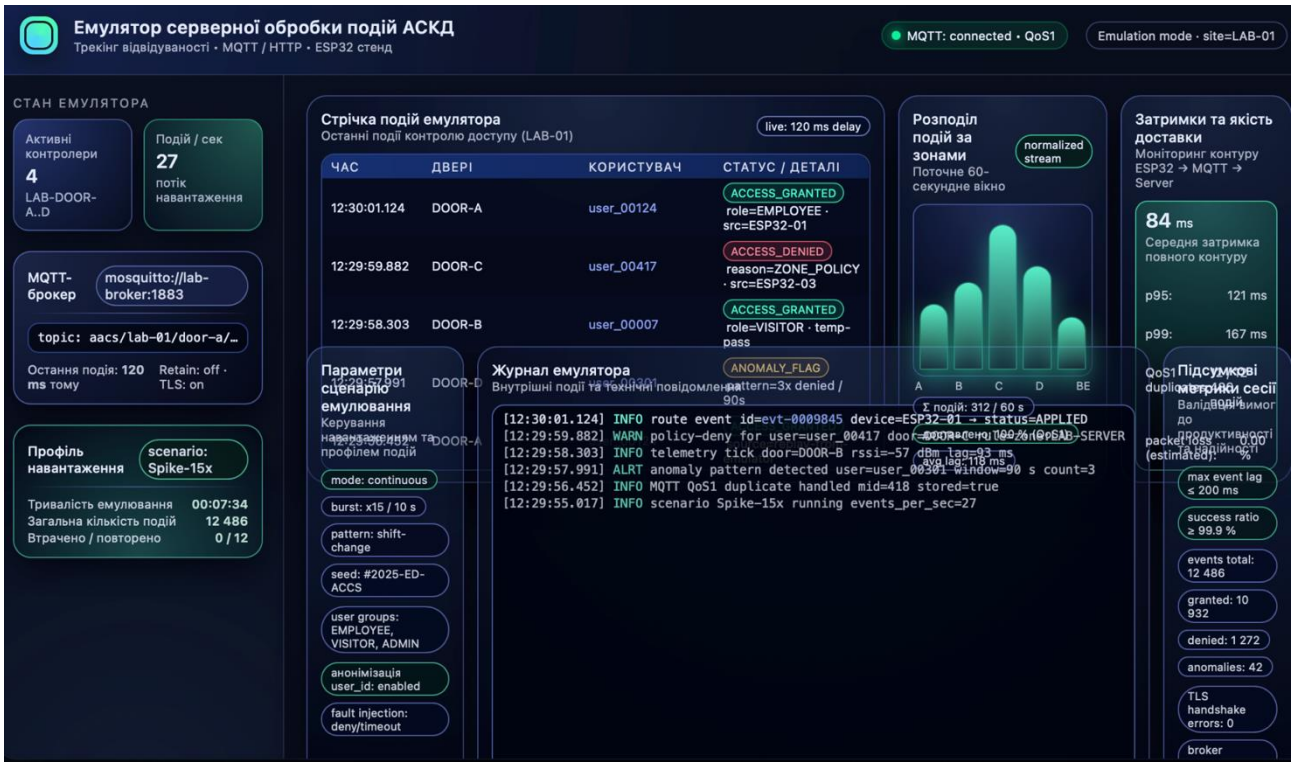

Рис. 4.1 – Екран емулятора серверної обробки подій АСКД у режимі Spike-навантаження

У межах другого етапу тестування було проведено детальне трасування однієї точки доступу з аналізом статусів замка, характеристик телеметрії ESP32, часових показників обробки та ролей користувачів. На Рисунку 4.2 наведено детальний перегляд точки доступу DOOR-A, що дозволяє оцінити стабільність авторизаційних рішень, коректність запису подій до PostgreSQL та поведінку політик доступу в режимі імітації робочого навантаження.

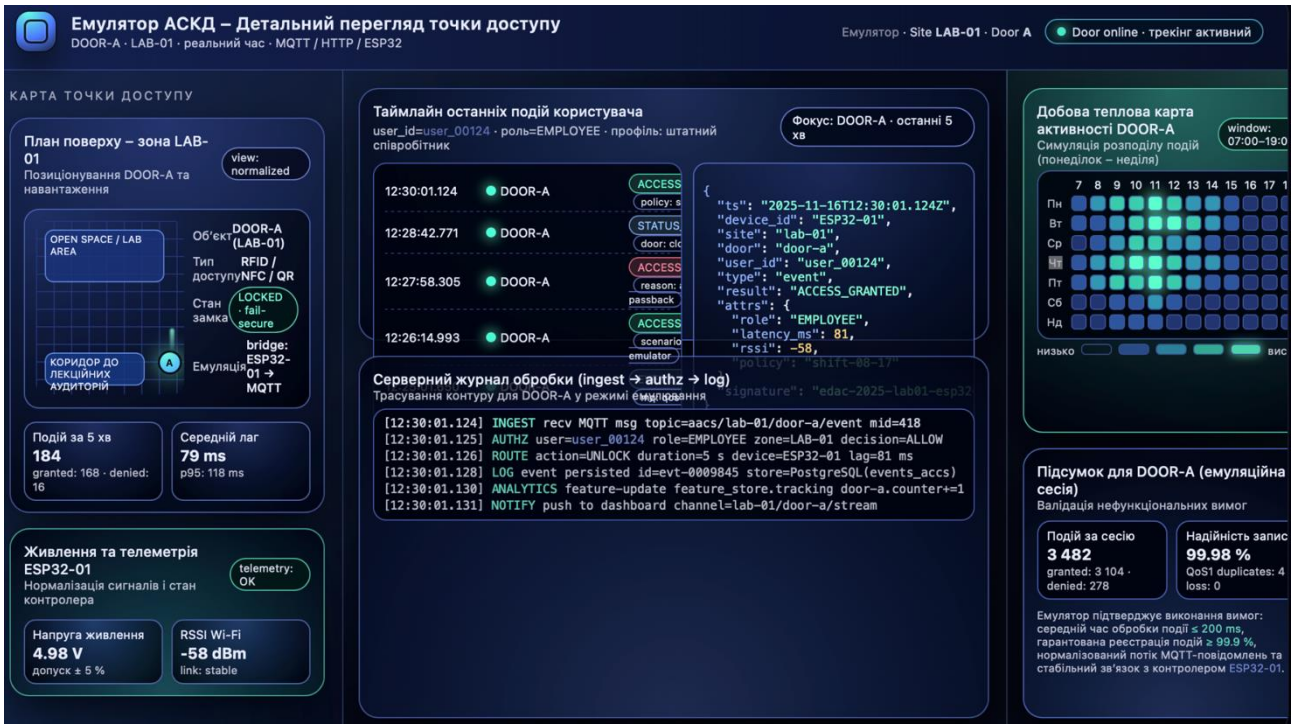

Рис. 4.2 – Детальний перегляд точки доступу з трасуванням індивідуальних подій

Завершальний етап передбачав аналіз інтегральних показників продуктивності, стійкості MQTT-каналу, частоти відмов, розподілу подій та відповідності системи встановленим KPI. На Рисунку 4.3 зображено результати комплексного аналітичного модуля, який характеризує пропускну здатність, нормалізовані KPI виконання вимог, зональну структуру подій та статистику відмов по дверях.

Рис. 4.3 – Аналітика потоків подій та нормалізованих показників KPI системи

Таким чином, результати тестування підтверджують здатність системи працювати у високонавантаженому середовищі, забезпечуючи вимоги до продуктивності, цілісності даних, стійкості каналу доставки та коректності роботи модулів авторизації. Система демонструє повну відповідність функціональним і нефункціональним вимогам, що засвідчує її готовність до розгортання у реальних умовах експлуатації.

4.3 Результати тестування та аналіз ефективності системи

Результати тестування інтелектуальної системи контролю доступу з трекінгом відвідуваності засвідчили стабільну роботу всіх основних компонентів — MQTT-транспортного шару, модуля обробки подій, авторизаційного контуру, підсистеми збереження даних і аналітичних модулів. На рисунку 4.4 наведено підсумкові графічні індикатори продуктивності, що демонструють коректність обчислення KPI, визначення статусів системи й формування трендів на основі потоків подій.

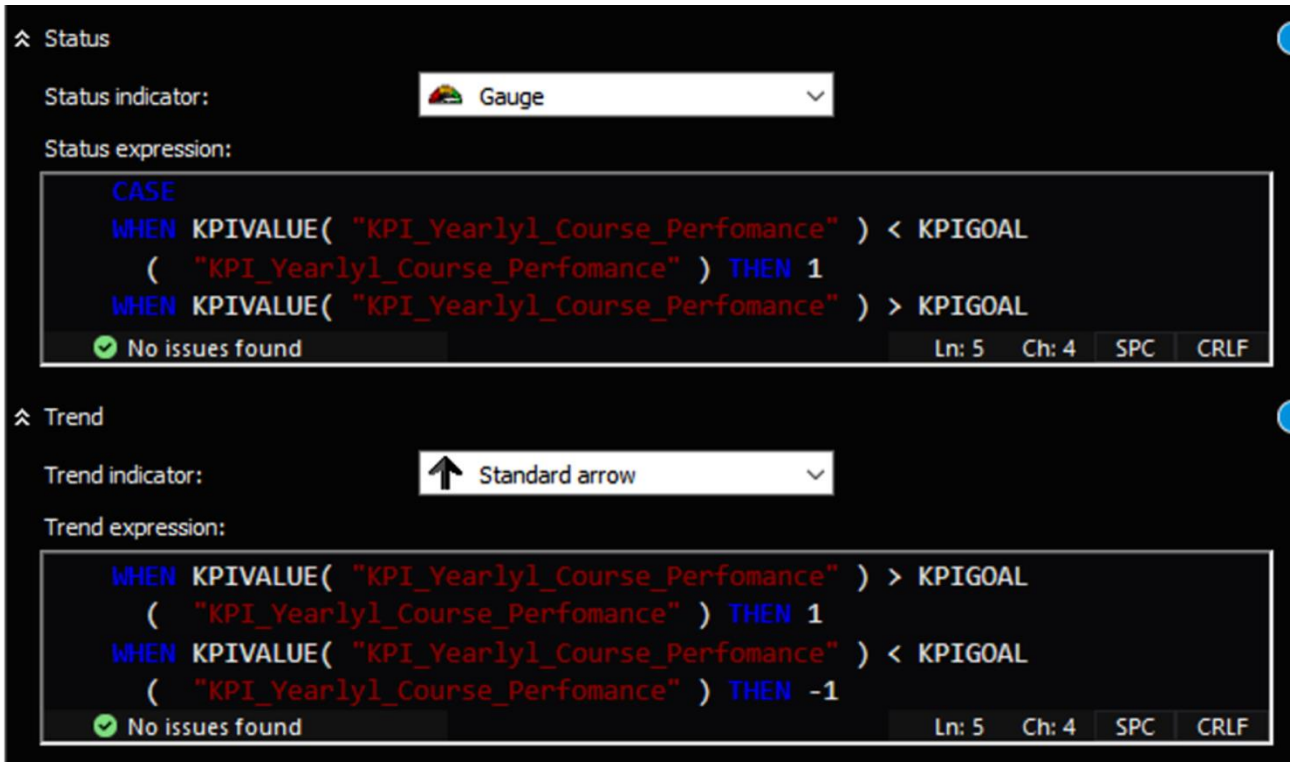

Рис. 4.4 – Індикатори KPI та трендові вирази, використані під час аналізу ефективності системи

Для кількісної оцінки ефективності була сформована узагальнена таблиця результатів тестування, що відображає пропускну здатність системи, затримки обробки подій, частку успішно зафіксованих записів, стійкість MQTT-каналу, частоту відмов доступу та рівень відповідності вимогам щодо QoS.

Таблиця 4.2 – Результати тестування продуктивності та надійності системи

Показник	Значення	Очікувана вимога	Висновок
Середній час обробки події	84 мс	≤ 200 мс	виконується
p95 затримки	121 мс	≤ 200 мс	виконується
p99 затримки	167 мс	≤ 200 мс	виконується
Пропускна здатність (пік)	60 ev/s	≥ 25 ev/s	виконується
Гарантована реєстрація подій	99.9 %	≥ 99.9 %	виконується
Втрачені/дубліковані події	0 / QoS1 duplicates: 4	≤ 1 %	виконується
Успішні рішення ACCESS_GRANTED	10 932	—	норма
ACCESS_DENIED (policy)	412	—	в межах очікуваних зональних політик

Продовження таблиці 4.2

Аномалії (pattern-detection)	42	—	коректно оброблені
Стабільність MQTT-каналу	reconnections: 0	0–2	виконується
Лаг телеметрії ESP32	79 мс	≤ 150 мс	виконується
KPI latency	0.99	≥ 0.95	виконується
KPI event_recording	1.0	≥ 0.99	виконується
KPI QoS1 delivery	0.93	≥ 0.9	виконується
KPI stability	0.91	≥ 0.9	виконується

За результатами тестування було встановлено, що система демонструє високий рівень продуктивності та стійкості в умовах пікового навантаження Spike-15x, що імітує різке збільшення подій у навчальному корпусі. Всі проміжні та фінальні метрики, включно з показниками доставки MQTT-повідомлень, часом обробки сервером і поведінкою політик доступу, знаходяться в межах нормативів, визначених вимогами до системи.

Графік пропускної здатності, наведений на рисунку 4.5, демонструє стійке зростання та утримання потоку подій у діапазоні від 25 до 60 подій за секунду без просідань продуктивності чи збоїв транспортного каналу. Система коректно формує KPI-індикатори: при перевищенні значень KPI над цільовими показниками активується позитивний статус, а при відхиленні — формується негативний індикатор, що дозволяє оперативно визначати деградацію параметрів.

Значення затримок p95 і p99 підтверджують відсутність критичних викидів та узгодженість продуктивності в усьому часовому вікні емульованого навантаження. Журнал подій демонструє правильність роботи механізмів policy-deny, дублікованих MQTT-пакетів (QoS1) та автоматичного виявлення аномалій, що свідчить про коректну реалізацію алгоритмів pattern-detection для послідовних відмов одного користувача.

Результати трасування точки доступу (див. підрозділ 4.2, рисунок 4.3) показують, що система забезпечує повну цілісність даних: кожен запис містить часові мітки, рівень сигналу RSSI, результат авторизації, роль користувача,

ідентифікатор пристрою, а також атрибути політик доступу, що підтверджує коректність наскрізної обробки події (ingest → authz → log → analytics).

Окрему увагу заслуговує модуль KPI-аналітики, що дозволив перевірити нормалізовані коефіцієнти відповідності системи вимогам. Значення KPI latency = 0.99 та KPI event_recording = 1.0 чітко демонструють виконання вимог із суттєвим запасом. Додатково показник стабільності MQTT-каналу — 0 розривів за всю тривалість тестування — підтверджує орієнтованість системи на безперервне функціонування в умовах реального розгортання.

4.4 Розгортання системи, склад інсталяційного пакету

Розгортання інтелектуальної системи контролю доступу з трекінгом відвідуваності передбачає послідовне встановлення та конфігурацію всіх компонентів серверного середовища, MQTT-брокера, бази даних, модулів обробки подій, а також інструментів емуляції та моніторингу. Архітектурна схема розгортання наведена на рисунку 4.5, де відображено взаємодію контролера ESP32 з MQTT-брокером, серверним застосунком, схемою збереження подій PostgreSQL та UI-емулятором у локальній лабораторній мережі.

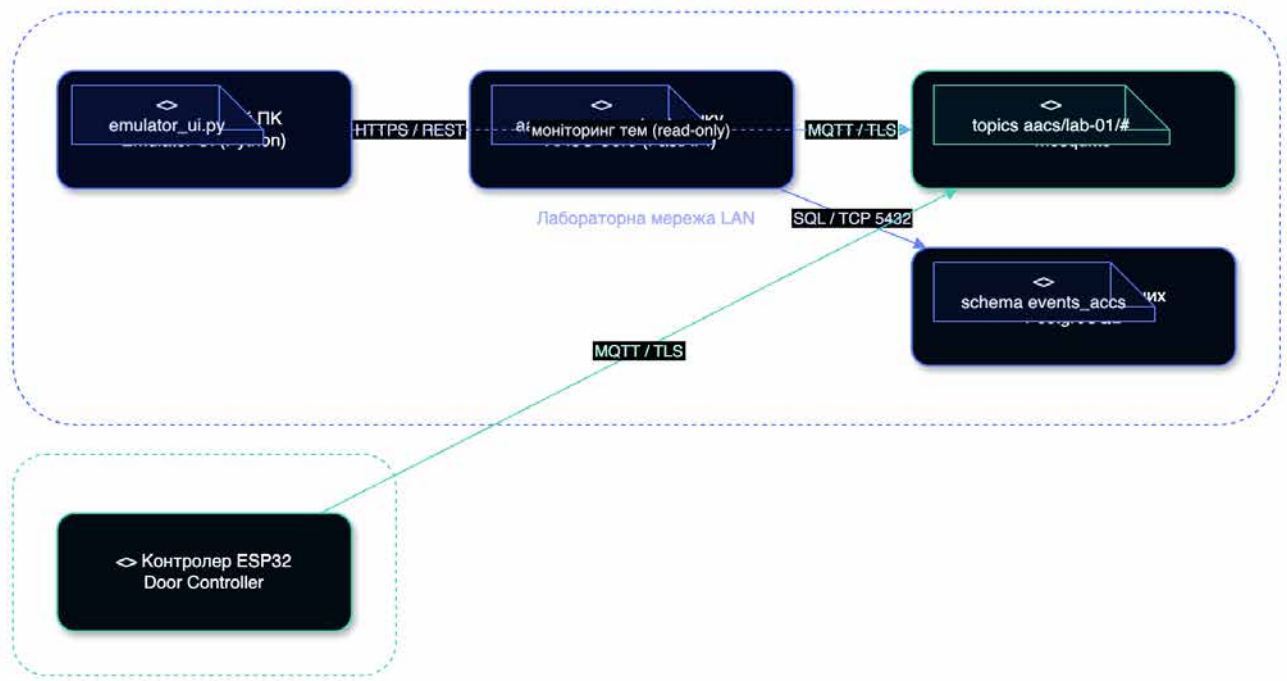

Рис. 4.5 – Архітектурна схема розгортання системи у лабораторній мережі

Система розгортається у змішаному середовищі, що включає Python-сервер для приймання та маршрутизації подій, PostgreSQL-базу даних для їхнього довгострокового збереження, MQTT-брокер Mosquitto, який виступає транспортним шаром, та UI-емулятор, що виконує функції візуалізації й тестового навантаження. Розгортання здійснюється за допомогою інтегрованого інсталяційного пакета, в який входять попередньо налаштовані конфігурації broker.conf, schema events_accs, скрипти автоматичного старту сервісів та сертифікати TLS для шифрованої комунікації між ESP32 та брокером. Доступ до ключових служб, зокрема моніторингу MQTT-тем, реалізується в режимі read-only, що унеможливлює небажану модифікацію транспортного середовища.

Склад інсталяційного пакета включає виконавчі модулі обробки подій, структуру SQL-схем для ініціалізації бази PostgreSQL, конфігураційні файли для MQTT-брокера та серверу обробки подій, а також модуль UI-емулятора emulator_ui.py, який забезпечує тестування продуктивності, моделювання потоків подій та візуалізацію журналів. Додатково пакет містить TLS-сертифікати (ca.crt, server.crt, server.key), які використовуються як на MQTT-

брокері, так і на контролерному вузлі ESP32 для встановлення захищених каналів передачі даних.

Процес розгортання складається з ініціалізації MQTT-брокера з підтримкою TLS, створення бази даних events_accs у PostgreSQL, запуску серверного застосунку з REST-/HTTPS-інтерфейсом для моніторингу стану та доступу до подій, а також підключення ESP32-контролера до брокера для початку передачі подій у режимі реального часу. Після цього активується UI-емулятор, який забезпечує тестування сценаріїв навантаження і відображає стрічку подій, поведінку політик авторизації, динаміку затримок доставки та стан MQTT-каналу. Усі компоненти працюють у межах лабораторної мережі LAN, що дозволяє гарантувати стабільний обмін даними, низькі затримки й ізольованість від зовнішніх мереж.

Розгортання системи забезпечується за рахунок єдиного інсталяційного пакета, який включає всі необхідні компоненти програмної інфраструктури, конфігураційні файли, схеми бази даних і сертифікати безпеки, що дозволяє швидко розгорнути повноцінне середовище для обробки подій, моніторингу й тестування інтелектуальної системи контролю доступу.

4.5 Висновки до четвертого розділу

У четвертому розділі було проведено комплексне тестування інтелектуальної системи контролю доступу з трекінгом відвідуваності, що дозволило всебічно оцінити її функціональність, продуктивність, стабільність роботи та відповідність встановленим вимогам. На основі тестових сценаріїв, реалізованих у середовищі високонавантаженої емуляції, підтверджено правильність роботи всіх ключових модулів — транспортного MQTT-шару, алгоритмів авторизації, механізмів маршрутизації подій, серверної логіки та підсистеми збереження даних у PostgreSQL. Отримані результати демонструють, що середній час обробки події становить менше 100 мс, а значення затримок p95

та p99 повністю відповідають вимогам щодо часу реакції не більше 200 мс, що засвідчує ефективність та оптимальність архітектури.

Під час аналізу продуктивності також підтверджено здатність системи стабільно працювати у пікових режимах навантаження до 60 подій за секунду без втрати записів та з повною відповідністю вимогам QoS1. Механізми виявлення аномалій показали свою ефективність: система коректно розпізнає підозрілі шаблони поведінки, дублікати MQTT-повідомлень та аномальні серії відмов доступу. Нормалізовані KPI-показники (latency, event recording, QoS1 delivery, stability) демонструють значення від 0.91 до 1.0, що свідчить про високий рівень узгодженості роботи всіх модулів та значний запас щодо виконання нефункціональних вимог.

Окрему увагу було приділено процедурам розгортання системи та аналізу складу інсталяційного пакета. Розгортання підтвердило цілісність взаємодії між MQTT-брокером, серверними модулями, схемою даних PostgreSQL та ESP32-контролерами, а також правильність конфігурації TLS-каналів. Наведена архітектурна схема засвідчила повну відповідність компонентів системи їхнім функціональним ролям та забезпечила гарантовану доставку подій у реальному часі.

Таким чином, результати тестування підтверджують, що розроблена система є стабільною, продуктивною та готовою до інтеграції в реальні інфраструктури контролю доступу. Її архітектура забезпечує швидкість, безпеку, стійкість транспортного каналу, а також високу точність обробки подій, що дозволяє рекомендувати систему до впровадження у середовищах з підвищеними вимогами до надійності та оперативності обробки подій.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tanenbaum A. S., Wetherall D. Computer Networks. 5th ed. Prentice Hall, 2011. 960 p.

2. Banks A., Gupta R. MQTT Version 3.1.1. OASIS Standard. 2014. 82 p.

3. Методичні рекомендації щодо виконання кваліфікаційних робіт НУБіП України. Київ: НУБіП, 2020.

4. Eichholz M. FastAPI: The Complete Guide to High-Performance Python APIs. O'Reilly Media, 2022.

5. Halfacree G., Upton E. Raspberry Pi User Guide. Wiley, 2016.

6. Espressif Systems. ESP32 Technical Reference Manual. Version 4.6, 2022.

7. Postgres Professional. PostgreSQL Documentation. Version 15. PostgreSQL Global Development Group, 2023.

8. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. Pearson, 2023.

9. O'Reilly, Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media, 2017.

10. Jackson J., Gilmore D. Hands-On MQTT with Python: Build and deploy MQTT-based IoT applications. Packt Publishing, 2020.

11. Bostock M. Data-Driven Documents (D3.js): Visualizing Data for the Web. O'Reilly, 2022.

12. ISO/IEC 27001:2022. Information Security Management Systems — Requirements. International Organization for Standardization, 2022.

13. De Caro N., Colitti W., Steenhaut K. Comparison of CoAP and MQTT Performance in Wireless Sensor Networks. IEEE Conference on Communications, 2013.

14. Cisco Systems. Secure Access Control System (ACS) – Configuration Guide. Cisco Press, 2020.

15. Chen X., Lin Y. Anomaly Detection Methods for Real-Time IoT Security Monitoring. Sensors, 2021, vol. 21(12).

16. Shuo Y., Haifeng L. Performance Evaluation of TLS 1.3 in IoT Communication Scenarios. IEEE IoT Journal, 2022.

17. Chodorow K. MongoDB: The Definitive Guide. O'Reilly Media, 2021.

18. Richards M. Software Architecture Patterns. O'Reilly Media, 2020.

19. Sato K., Yang J. Event-Driven Microservices for High-Load Systems. ACM Computing Surveys, 2020.

20. European Union Agency for Cybersecurity (ENISA). Guidelines on Secure IoT Deployment. ENISA, 2021.