Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

РОЗРОБКА СИСТЕМИ ЗБИРАННЯ ТА ОБРОБКИ ДАНИХ ДЛЯ ПІДВИЩЕННЯ БЕЗПЕКИ

DEVELOPMENT OF A DATA COLLECTION AND PROCESSING SYSTEM TO IMPROVE SECURITY

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Чепіль Микола Анатолійович

(підпис)

Керівник: к.т.н., доцент
Бортник Катерина Яківна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т.ТЕРЛЕЦЬКИЙ
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Чепілю Миколі Анатолійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Розробка системи збирання та обробки даних для підвищення безпеки*

Керівник роботи *к.т.н., доцент Бортник Катерина Яківна*

затверджені наказом закладу вищої освіти від «17» червня 2025 року № 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 09.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблеми за темою роботи та постановка завдань дослідження
Теоретичний аналіз підходів до створення систем контролю доступу
Аналіз існуючих рішень представлених на ринку
Огляд технологій, що використовуються в системах контролю доступу
Реалізація системи контролю доступу та дослідження ефективності її використання
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Теоретичні основи забезпечення безпеки фізичного доступу	*Бортник К.Я.., доцент*		
Обґрунтування вибору технологій та аналіз апаратно-програмних рішень	*Бортник К.Я.., доцент*		
Практична реалізація системи контролю доступу	*Бортник К.Я.., доцент*		
Нормоконтроль	*Багнюк Н.В., доцент*		
Гарант ОП	*Гринюк С.В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О.І., ст.викладач*		

7. Дата видачі завдання 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 01.08.2025 р.	
2.	*Аналіз проблеми за темою роботи та постановка завдань дослідження*	До 20.08.2025 р.	
3.	*Теоретичне дослідження та практична реалізація*	До 25.09.2025 р.	
4.	*Практична реалізація об'єкта проектування*	До 20.10.2025 р.	
5.	*Висновки та пропозиції*	До 25.10.2025 р	
6.	*Формування списку використаних джерел*	До 27.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 05.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 11.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ (підпис) Чепіль М.А. (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ (підпис) Бортник К.Я. (прізвище, ініціали)

АНОТАЦІЯ

Чепіль М. А. Розробка системи збирання та обробки даних для підвищення безпеки.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається з вступу, трьох розділів, висновків, списку використаних джерел, додатків.

Перший розділ присвячено огляду предметної області систем контролю доступу. Розглянуто принципи роботи електронних замків і датчиків, моделі збору подій, телеметрії та логування, а також проблему неповних та неузгоджених даних. Окремо подано короткий огляд комерційних рішень таких як Ajax, Yale, Nuki, SmartThings та їх можливостей.

Другий розділ описує вибір апаратних і програмних засобів. Порівнюються платформи Arduino, ESP32 та Raspberry Pi, типи датчиків і способи передавання даних через RS-232, RS-485 та Modbus RTU. Обґрунтовано використання Raspberry Pi, .NET, SQL Server, REST API та Blazor у розробці системи.

У третьому розділі наведено опис реалізованої системи: архітектуру, підключення замка й геркона, програму контролера, що зчитує стани та передає події через REST API, роботу серверної частини й структуру бази даних. Також проведено дослідження точності відтворення станів, оцінено коректні й некоректні сценарії роботи, затримки передачі подій та рівень узгодженості даних.

Ключові слова: система контролю доступу, електронний замок, геркон, Modbus RTU, Raspberry Pi, REST API, SQL Server, Blazor.

ANNOTATION

Chepil M. Development of a data collection and processing system to improve security.

Qualifying work of a Master's of EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

Qualification work consists of an introduction, three sections, conclusions, a references, three appendices.

The first chapter focuses on the overview of the domain of access control systems. It examines the principles of operation of electronic locks and door sensors, models of event collection, telemetry, and logging, as well as the problem of incomplete or inconsistent data. A brief overview of commercial solutions as Ajax, Yale, Nuki, SmartThings and their capabilities is also provided.

The second chapter describes the selection of hardware and software tools. It compares the platforms Arduino, ESP32, and Raspberry Pi, the types of sensors used, and methods of data transmission via RS-232, RS-485, and Modbus RTU. The choice of Raspberry Pi, .NET, SQL Server, REST API, and Blazor for system implementation is justified.

The third chapter presents the description of the implemented system: the architecture, the connection of the lock and reed switch, the controller software that reads states and transmits events via REST API, the operation of the server side, and the structure of the database. It also includes an evaluation of the system’s accuracy in reproducing real states, an analysis of correct and incorrect scenarios, transmission delays, and the overall consistency of collected data.

Keywords: access control system, electronic lock, reed switch, Modbus RTU, Raspberry Pi, REST API, SQL Server, Blazor.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасні системи контролю фізичного доступу активно переходять від механічних рішень до електронних і мережевих технологій, що забезпечують автоматизований моніторинг, реєстрацію подій та аналітику станів. Однак навіть у розвинених системах зберігається одна з ключових технологічних проблем – неповнота та неточність даних про реальний стан дверей і замкового механізму. У більшості рішень контроль доступу обмежується фіксацією логічного стану замка, тоді як фактичне положення дверей контролюється окремими сенсорами, а кореляція цих даних часто відсутня або недостатня. Це створює небезпечні сценарії, коли замок може перебувати в стані «заблоковано», але двері фізично відчинені, або навпаки – система повідомляє про успішне замикання, тоді як у реальності відбулося заїдання, поломка чи спроба злому.

У контексті підвищення вимог до кіберфізичної безпеки та віддаленого моніторингу виникає необхідність у розробці систем, здатних поєднувати дані з електронного замка та датчика дверей, аналізувати їх узгодженість і оцінювати достовірність отриманої інформації. Саме тому дослідження механізмів збору, обробки та інтерпретації даних з фізичних сенсорів стає критично важливим завданням. Актуальність підсилюється тим, що неузгодженість даних прямо впливає на здатність системи адекватно реагувати на інциденти, вести коректні журнали подій та забезпечувати реальну, а не формальну безпеку.

Метою роботи є розробка системи збирання та обробки даних для підвищення безпеки на основі електронного замка та датчика дверей, а також дослідження точності відображення фізичного стану дверей за допомогою поєднаних апаратних та програмних засобів.

Об’єкт дослідження – процеси реєстрації, обробки та зберігання подій у системах контролю доступу.

Предмет дослідження – методи та програмно-апаратні засоби збирання даних від електронного замка та магнітного датчика, а також підходи до оцінки достовірності та узгодженості отриманих даних.

Для досягнення поставленої мети необхідно виконати такі завдання:

– здійснити теоретичний аналіз систем контролю доступу, принципів роботи електронних замків та сенсорів, моделей збору подій, телеметрії й логування;

– проаналізувати сучасні комерційні рішення та визначити їхні можливості й обмеження;

– обґрунтувати вибір апаратної та програмної платформи для побудови системи, включно з Raspberry Pi, .NET, REST API, SQL Server, RS-232/RS-485 та Modbus;

– розробити архітектуру системи, апаратну частину та програмне забезпечення контролера, серверну частину й інтерфейс моніторингу;

– реалізувати повний цикл збирання й передавання даних про стан замка та дверей;

– провести дослідження точності й узгодженості даних, визначивши сценарії коректної та некоректної роботи системи;

– виконати оцінку достовірності, обчислити ключові показники кореляції та відхилень і сформувати висновки щодо можливостей та обмежень побудованої системи.

Апробація: результати, отримані в ході виконання кваліфікаційної роботи, були апробовані та представлені у науковому журналі «Комп’ютерно-інтегровані технології: освіта, наука, виробництво» [1].

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ФІЗИЧНОГО ДОСТУПУ

1.1 Загальна характеристика систем контролю доступу

Системи контролю доступу (СКД) – це комплекс технічних засобів, призначених для регулювання й фіксації фізичного доступу до приміщень шляхом ідентифікації користувачів, прийняття рішень щодо допуску та контролю фактичного стану дверей або зони доступ [2]. Їхня ефективність визначається поєднанням трьох ключових аспектів: принципів роботи, типу використовуваного замкового механізму та сенсорної системи, що забезпечує підтвердження реального стану об’єкта [3].

1.1.1 Принципи роботи систем контролю доступу

Робота будь-якої СКД базується на послідовності типових операцій:

– ідентифікація користувача. На першому етапі система визначає особу, яка намагається отримати доступ. Методи ідентифікації можуть включати використання ключів, електронних карт, PIN-кодів, RFID-міток, мобільних додатків або біометричних даних [4];

– автентифікація та перевірка прав доступу. Після ідентифікації контролер звіряє отримані дані з базою дозволів. На цьому етапі визначається, чи має користувач право увійти саме до цієї зони в цей момент часу [5];

– прийняття рішення та керування замком. У разі успішної перевірки контролер формує команду на розблокування замка. Якщо права доступу відсутні або виявлено спробу маніпуляції – замок залишається заблокованим;

– контроль фактичного стану дверей. Для підвищення безпеки важливо не лише подати команду на розблокування, а й отримати підтвердження, що двері дійсно були відчинені або закриті. Це дозволяє виявляти невідповідності, спроби злому та технічні несправності;

– реєстрація подій та реагування на аномалії. Система фіксує всі події доступу, відхилення від нормальної поведінки та несанкціоновані дії. У випадку небезпечної ситуації ініціюється локальна або віддалена тривога [6].

1.1.2 Типи замків у системах контролю доступу

Замковий механізм є центральним елементом СКД, оскільки саме він фізично регулює доступ до приміщення. Основні типи замків:

– механічні замки, що забезпечують базовий рівень безпеки. Не підтримують ідентифікацію та журналювання, тому не часто застосовуються в сучасних СКД;

– електромеханічні замки, що поєднують механічний вузол і електричний привід, що дозволяє розблоковувати замок за сигналом контролера. Забезпечують надійність і можливість інтеграції з простими системами доступу;

– електромагнітні замки, що створюють стримувальне магнітне поле, яке блокує двері до моменту подання команди. Відзначаються простотою, але потребують постійного живлення;

– електронні та «розумні» замки, оснащені мікроконтролерами, датчиками стану та модулями зв’язку [7]. Можуть передавати інформацію про стан замка, рівень заряду батареї, історію доступу та події [8].

1.1.3 Типи сенсорів у системах контролю доступу

Для забезпечення повного циклу безпеки замок сам по собі недостатній – потрібен контроль стану дверей та середовища. Найпоширеніші сенсори:

– герконові датчики, що фіксують факт відкриття або закриття дверей. Є базовим і надійним способом визначити реальну позицію дверей;

– оптичні та інфрачервоні датчики, що визначають наявність перешкоди або момент проходження через дверний проріз. Можуть використовуватися в системах високої точності;

– датчики вібрації та удару, які реагують на механічний вплив, що дозволяє виявляти спроби силового злому або маніпуляції з полотном дверей;

– тамперні сенсори, що встановлюються для фіксації спроби демонтажу, відкриття корпусу або втручання в апаратну частину системи.

Сенсорні модулі відіграють ключову роль у створенні розширеної картини подій, дозволяючи системі контролю доступу не лише регулювати вхід, але й визначати аномалії, які не можуть бути виявлені за допомогою одного лише замкового механізму.

1.2 Проблематика фіксації подій у системах доступу

Ефективність систем контролю доступу значною мірою залежить від того, наскільки повно й достовірно фіксуються події, пов’язані з відкриттям, закриттям і використанням дверей. Неповнота або неточність даних призводить до втрати критично важливої інформації, що унеможливлює своєчасне виявлення загроз, створює «сліпі зони» в системі безпеки та значно знижує рівень контролю над фізичним доступом. Проблематика фіксації подій охоплює кілька ключових аспектів: наявні типові загрози, обмеження традиційних рішень та випадки, коли дані про доступ залишаються неповними.

1.2.1 Типові загрози у системах контролю доступу

Найбільш поширені загрози пов’язані з тим, що факт подання команди на розблокування замка не гарантує реального контролю над ситуацією [9]. До таких загроз належать:

– силовий злом при заблокованому замку, у випадку фізичного впливу на двері наприклад: віджимання, вибивання, деформація полотна, замок може залишатися заблокованим, але двері фактично відкриваються. Якщо система не контролює реальний стан дверей, така подія залишиться непоміченою;

– спроби обходу замкового механізму, наприклад, маніпуляції з петлями, підважування дверей або відкриття без взаємодії із замком. Без окремого датчика дверей система не може зафіксувати подібні дії;

– несанкціоноване відкриття зсередини в ситуаціях де двері можуть бути відкриті зсередини при заблокованому ззовні замку, якщо система враховує лише стан замка, така дія не буде зареєстрована як інцидент;

– тривале утримання дверей відчиненими, що створює як фізичні ризики, так і можливості для обходу захисту, але часто залишається невиявленим у базових системах [10].

1.2.2 Обмеження традиційних рішень контролю доступу

Більшість наявних рішень, особливо у нижчих цінових сегментах, мають низку обмежень:

– фіксується лише стан замка, а не стан дверей, замок може повідомляти про свій стан але не про реальне положення дверей. Це створює логічну прогалину: система «вважає», що доступ заблоковано, хоча двері можуть бути фізично відчинені [11];

– відсутність кореляції подій замка та дверей, які не аналізуються разом, що унеможливлює виявлення аномалій, наприклад, двері відкриті без команди розблокування;

– низький рівень деталізації журналу подій, у багатьох системах фіксуються лише основні події: «замок відкрито», «замок закрито». Не ведеться облік часу затримки між операціями, тривалості відкриття дверей або нестандартних послідовностей;

– відсутність тампер-контролю, дешеві або спрощені системи не мають механізмів виявлення демонтажу, ударів, вібрацій чи інших маніпуляцій;

– орієнтація на реактивний, а не превентивний підхід. Традиційні системи часто лише блокують доступ, тоді як сучасні вимоги передбачають превентивне виявлення нестандартної поведінки й спроб злому [12].

1.2.3 Випадки неповноти даних у системах доступу

Неповні або неоднозначні дані – одна з найкритичніших проблем, оскільки вона створює умови для пропуску реальних інцидентів [13]. Типові випадки:

– команда на розблокування надійшла, але двері фактично не відкрилися. Це може вказувати на несправність механізму, заїдання або хибну взаємодію користувача;

– двері відкрилися, але система не отримала або не зафіксувала стан замка. Подібна ситуація може бути ознакою обходу системи або збою датчика;

– замок заблоковано, але двері залишилися прочиненими. Такий сценарій створює вразливість, але за відсутності сенсора дверей не може бути виявленим;

– аномальні послідовності подій. Наприклад: замок – locked, двері – open. Без повної картини подій система не здатна інтерпретувати це як інцидент;

– відсутність даних у журналах через втрату зв’язку або низький рівень телеметрії. Це унеможливлює подальший коректний аналіз або реконструкцію події.

Таким чином, проблематика фіксації подій у системах контролю доступу полягає у розриві між логічним станом замка й реальним положенням дверей, недостатній деталізації телеметрії та обмежених можливостях традиційних рішень. Це створює основу для потреби в удосконалених системах, що поєднують дані з кількох джерел та застосовують аналітику для виявлення аномалій, що значно підвищує рівень безпеки фізичного доступу [14].

1.3 Методи та моделі збору даних у системах безпеки

1.3.1 Подійні моделі

Подійні моделі є однією з базових концепцій збирання даних у сучасних системах безпеки, оскільки вони дозволяють фіксувати кожну зміну стану об’єкта як окрему сутність у часі. У контексті систем контролю доступу подія трактується як конкретний факт, що має чітко визначений момент виникнення, причину та наслідки. Це може бути подання команди на розблокування замка, фактичне його відкриття, зміна стану дверей, спроба використання невірного ідентифікатора або будь-яка інша дія, яка має значення для безпеки об’єкта. Застосування подійної моделі забезпечує можливість точної реконструкції поведінки системи у часовій послідовності, що є критично важливим у випадках інцидентів або спроб злому.

Подійний підхід передбачає формування дискретного запису щоразу, коли відбувається зміна стану. На відміну від періодичного опитування, де дані збираються з певним інтервалом і частина інформації може бути втрачена між

циклами, подійна модель фіксує всі переходи станів у реальному часі. Це дає змогу визначити не лише факт події, але й її точний час, тривалість, попередні й наступні стани, що дозволяє виявляти нетипові патерни. Наприклад, якщо замок отримав команду на розблокування, але двері не відкрилися в межах очікуваного інтервалу, це може свідчити про заїдання механізму або невдалу спробу доступу. Якщо ж двері відкриваються без відповідної події розблокування замка, система має визначити це як потенційний інцидент.

Однак ефективність подійних моделей залежить від якості фіксації кожного стану, коректної роботи сенсорів і стабільності каналу зв’язку. Втрата навіть одного запису може розірвати логічний ланцюг подій і створити ситуацію, коли послідовність виглядає правдоподібною, але містить критичні прогалини. Наприклад, якщо система зареєструвала відкриття дверей, але з технічних причин не отримала сигнал про розблокування замка, аналіз може помилково інтерпретувати це як спробу силового проникнення. Аналогічно, якщо система не фіксує момент закриття дверей, але отримує подію блокування замка, то її інтерпретація стану об’єкта також буде неповною. Саме тому подійні моделі потребують механізмів дублювання, підтвердження доставки, використання часових міток високої точності та засобів виявлення пропущених подій.

Подійний підхід також тісно пов’язаний з архітектурою обробки даних. У розподілених системах контролю доступу події можуть передаватися асинхронно, через мережеві протоколи з можливими затримками. Це створює додаткові виклики щодо синхронізації часових міток і коректної інтерпретації порядку надходження подій. У практичному застосуванні для вирішення таких проблем часто використовуються черги повідомлень, механізми підтвердження доставки, локальне кешування та повторна передача подій при відновленні зв’язку.

Таким чином, подійні моделі забезпечують високий рівень точності та інформативності при збиранні даних у системах безпеки, дозволяючи виявляти складні сценарії поведінки, будувати логіку визначення аномалій та формувати повну картину доступу. Проте для їх ефективної роботи необхідні надійні

сенсорні механізми, стійкі канали зв’язку та алгоритми компенсування неповноти або втрати подій. Саме поєднання подійної моделі з іншими підходами до збирання інформації створює основу для побудови інтелектуальних та високонадійних систем контролю доступу.

1.3.2 Телеметрія пристроїв

Телеметрія пристроїв є фундаментальною складовою систем збирання даних у сфері фізичної безпеки, оскільки забезпечує постійний контроль за технічним станом обладнання, на якому базується система доступу. На відміну від подійної моделі, що фіксує дискретні дії, телеметрія надає безперервний або майже безперервний потік інформації, що описує робочі параметри пристроїв, серед яких рівень заряду живлення, стан елементів керування, температура, напруга живлення, інтенсивність та стабільність бездротового з’єднання, а також індикатори працездатності мікроконтролера чи замкового механізму. Ці параметри не є подіями у класичному розумінні, проте вони формують важливий контекст для інтерпретації поведінки системи та визначення потенційних відхилень від норми.

Використання телеметрії дає змогу завчасно виявляти проблеми, які можуть призвести до погіршення роботи або відмови системи. Наприклад, поступове падіння рівня живлення може попереджати про необхідність заміни батареї, послаблення сигналу бездротового модуля може вказувати на перешкоди або зловмисне глушіння, а підвищена температура контролера – на перенавантаження або внутрішню несправність. Завдяки цьому телеметрія слугує превентивним інструментом, що дозволяє мінімізувати ризики простоїв і забезпечити стабільну роботу системи доступу, особливо у критично важливих об’єктах.

Ще одним важливим аспектом телеметрії є її здатність сигналізувати про можливі спроби втручання. Зміни напруги, раптові переривання зв’язку, некоректні цикли перезавантаження або аномальна поведінка сенсорів можуть бути ознаками спроби злому чи кібератаки, спрямованої на порушення роботи

системи доступу. У такому разі телеметричні дані дозволяють швидко виявити джерело проблеми та запобігти її розвитку.

Попри свою інформативність, телеметрія не може бути самостійним джерелом даних про події доступу. Вона відображає лише технічний стан системи, але не фіксує конкретні дії користувачів, такі як відкриття дверей чи спроби ідентифікації. Це створює ситуацію, коли технічно справний пристрій може приховувати реальні загрози, якщо не реєструються зміни станів замка або дверей. Наприклад, рівень сигналу або батареї може бути ідеальним, але двері можуть бути силоміць відчинені без жодної події у журналі, якщо система покладається лише на телеметрію й не використовує подійні моделі.

Таким чином, телеметрія відіграє надзвичайно важливу роль у системах безпеки, забезпечуючи глибинний технічний контекст і можливість прогнозування несправностей, але її використання повинно бути комплексно доповнене подійними даними. Лише поєднання цих підходів дозволяє сформувати повну та достовірну модель роботи системи контролю доступу, необхідну для коректної інтерпретації інцидентів і виявлення аномалій у поведінці об’єкта.

1.3.3 Логування безпеки

Логування безпеки є центральним елементом інформаційної складової систем контролю доступу, оскільки саме воно забезпечує накопичення й збереження повної історії подій, необхідної для аналізу, аудиту та відтворення інцидентів. На відміну від подійної моделі, яка відображає лише факт зміни стану, та телеметрії, що характеризує загальний технічний контекст, логування інтегрує обидва ці підходи в єдину хронологічно впорядковану структуру. Такий журнал подій дозволяє встановити причинно-наслідкові зв’язки, оцінити поведінку користувачів, визначити реальні дії з дверима й замком, а також виявити відхилення від нормального режиму роботи. У випадку інцидентів логування відіграє роль основного джерела доказової бази, без якого неможливо виконати точну реконструкцію подій.

Коректне логування передбачає фіксацію не лише самих подій, але й їх контексту: точного часу, послідовності, супровідних телеметричних параметрів та станів усіх ключових компонентів системи. Саме завдяки цьому журнал не лише відображає лінійну послідовність дій, а й дозволяє робити висновки щодо аномалій. Наприклад, якщо в журналі зафіксовано відкриття дверей без попереднього розблокування замка, це може свідчити про спробу зламу. Якщо система декілька разів поспіль фіксує невідповідність між станом замка та станом дверей, це може бути ознакою несправності датчика або маніпуляції з обладнанням. Таким чином, логування є не лише реєстрацією фактів, але й інструментом виявлення логічних порушень у роботі системи.

Разом із тим саме журнал подій є найбільш уразливим елементом у контексті неповноти даних. Втрата одного або декількох записів через нестабільний канал зв’язку, переповнення буфера чи збій у роботі контролера може призвести до формування хибної картини. Неправильно синхронізовані часові мітки створюють ситуацію, коли події відображаються в іншому порядку, ніж відбувалися фактично, що унеможливлює коректний аналіз. Недостатня деталізація записів – наприклад, відсутність інформації про тривалість відкриття дверей або технічний стан датчика в момент події – також суттєво знижує цінність журналу. Помилки у фіксації крайових станів, такі як однократне реєстрування відкриття без подальшого закриття, можуть приховати інциденти або створити помилкові припущення про їхню природу.

Для забезпечення надійності логування необхідна правильна архітектура збирання та зберігання даних. Це включає використання механізмів підтвердження доставки повідомлень, дублювання записів, перевірки цілісності журналів та часової синхронізації між усіма компонентами системи. У розподілених системах особливого значення набувають технології черг повідомлень, що гарантують доставку та дозволяють уникати втрати подій у разі тимчасових відмов мережі. Крім того, важливо забезпечити захист журналів від несанкціонованого доступу та модифікації, оскільки підміна або видалення записів може спотворити аналітику і приховати сліди зламу.

Отже, логування безпеки є основою достовірного аналізу та аудиту систем контролю доступу, проте воно вимагає особливої уваги до питань цілісності, повноти та надійності даних. Лише за умови правильно організованого логування можливе точне відтворення інцидентів, виявлення комплексних загроз і забезпечення високого рівня інформаційної та фізичної безпеки об’єкта.

1.4 Аналітика подій та виявлення аномалій

Аналітика подій у системах контролю доступу спрямована на те, щоб не лише реєструвати факти зміни стану, а й інтерпретувати їх у ширшому контексті безпеки. Сучасні системи переходять від простого журналювання до інтелектуального аналізу, який дозволяє автоматично виявляти підозрілу поведінку, неузгодженість між подіями або дії, що не відповідають очікуваному сценарію. Основою такого аналізу є логічні моделі, часові патерни та кореляція станів різних компонентів системи, зокрема замка і дверей.

Логічні моделі аналітики засновані на визначенні правильних і неправильних послідовностей подій. Вони формують набір правил, які описують нормальну поведінку системи, наприклад: команда на розблокування замка має передувати фактичному відкриттю дверей; закриття дверей має передувати блокуванню замка; повторні невдалі спроби ідентифікації протягом короткого часу можуть вказувати на спробу підбору коду або RFID-мітки. Коли реальні події не відповідають визначеним правилам, система інтерпретує це як аномалію. Логічні моделі є основою rule-based detection – підходу, що дозволяє швидко локалізувати відхилення на основі простих умов, які легко формалізуються й не потребують складних обчислень.

Часові патерни доповнюють логічні моделі, оскільки не менш важливою, ніж сама послідовність подій, є їх тривалість та інтервали між ними. У нормальному режимі існують очікувані часові межі: між розблокуванням замка та відкриттям дверей зазвичай проходить кілька секунд; двері залишаються відкритими протягом типового часу; між відкриттям і повторним закриттям є

стабільна тривалість. Відхилення від цих часових норм можуть свідчити про збої або спроби втручання. Наприклад, якщо двері довго залишаються відчиненими, це може бути ознакою недбалості або підготовки до обходу системи. Якщо замок отримав команду на розблокування, але двері не відкрилися у визначений інтервал, це може вказувати на механічну несправність або хибну спробу доступу. Часовий аналіз дозволяє виявляти ситуації, які не можна визначити лише за послідовністю подій.

Найважливішим елементом аналітики подій у системах фізичного доступу є кореляція станів замка і дверей. Замок сам по собі фіксує лише логічну команду блокування чи розблокування, тоді як двері відображають реальний фізичний стан об’єкта. Кореляція цих двох потоків даних дозволяє ідентифікувати найкритичніші інциденти: відкриття дверей при заблокованому замку, що свідчить про спробу злому; блокування замка, коли двері залишаються відчиненими, що створює хибне відчуття безпеки; відсутність відкриття дверей після розблокування замка, що може означати механічну несправність або помилку користувача. Саме ця кореляція створює більш повну картину подій і дозволяє системі переходити від реактивного до проактивного підходу у визначенні загроз.

Отже, аналітика подій і виявлення аномалій є критично важливими складовими сучасних систем контролю доступу. Поєднання логічних моделей, часових патернів і кореляції станів замка й дверей забезпечує можливість точного розпізнавання небезпечних ситуацій, попередження інцидентів і підвищення якості моніторингу, що значно підсилює загальний рівень фізичної безпеки на об’єкті.

1.5 Огляд сучасних комерційних рішень

1.5.1 Огляд Ajax

Ajax є одним із найбільш відомих сучасних комплексів охоронної сигналізації та автоматизації для житлових і комерційних приміщень. Це

бездротова модульна система, що поєднує у собі датчики безпеки, централі керування, виконавчі модулі та мобільний застосунок для дистанційного адміністрування. Основною ідеєю платформи є створення екосистеми, у якій усі компоненти взаємодіють між собою через фірмові радіопротоколи з високим рівнем захищеності та енергоефективності (рис. 1.1).

Рисунок 1.1 – Пристрої системи безпеки Ajax [15]

Центральним елементом системи є хаб, який забезпечує збір даних від датчиків, керування сценаріями безпеки та передачу інформації до хмарного сервера й мобільного додатку. Хаб підтримує мережеві інтерфейси такі як: Ethernet, Wi-Fi або мобільний зв’язок, що забезпечує стабільність роботи навіть у разі несправностей основного каналу. Саме він відповідає за аналіз отриманих сигналів і ініціювання тривог або сповіщень.

До системи підключаються різноманітні бездротові датчики: магнітні датчики відкриття, сенсори руху, датчики розбиття скла, пожежні та затоплення, а також модулі керування живленням і виконавчі реле. Окремі компоненти можуть забезпечувати додаткові функції, такі як передавання фото при спрацюванні тривоги, контроль температури чи визначення рівня освітленості. Завдяки власним протоколам Jeweller та Wings система забезпечує мале енергоспоживання, великий радіус дії та захист від перешкод і блокування сигналу.

Керування системою здійснюється через мобільний застосунок або веб інтерфейс, де користувач може переглядати стан датчиків, налаштовувати сценарії автоматизації, отримувати сповіщення про події та здійснювати контроль над виконавчими модулями. Ajax підтримує інтеграцію з охоронними компаніями та може передавати тривожні події до пультів централізованого спостереження. Окрім цього, система дозволяє створювати автоматизовані сценарії, наприклад, увімкнення сирени при відкритті дверей чи активування реле при зміні охоронного режиму.

Ajax вирізняється високою увагою до надійності та захисту інформації. Бездротове спілкування між компонентами шифрується, застосовуються механізми виявлення перешкод, спроб глушіння та втручання. Багато пристроїв мають незалежні джерела живлення та можуть функціонувати під час відсутності основного електроживлення.

У цілому Ajax є потужною, гнучкою та масштабованою системою, яка поєднує охоронні функції, автоматизацію та моніторинг технічного стану середовища. Її архітектура орієнтована на кінцевого користувача, який потребує комплексного рішення для безпеки домівки або бізнесу без складного монтажу й налаштування.

1.5.2 Yale

Yale є одним із найстаріших і найвідоміших світових виробників замкових систем і рішень для фізичної безпеки. Упродовж останніх десятиліть компанія здійснила поступовий перехід від традиційних механічних замків до розумних систем доступу, інтегрованих у цифрову екосистему «smart home». Її сучасна лінійка продуктів включає електронні замки, смарт-замки з підтримкою мобільних технологій, модулі для підключення до систем автоматизації та аксесуари для підвищення безпеки.

Смарт-замки Yale обладнані електронним керуванням, що дозволяє відкривати двері за допомогою PIN-кодів, карток, брелоків, мобільних додатків або Bluetooth-з’єднання. Частина моделей підтримує інтеграцію з голосовими асистентами та платформами розумного дому, такими як Google Home, Amazon

Alexa чи Apple HomeKit. Завдяки цьому користувач може дистанційно керувати доступом, переглядати журнали подій, створювати віртуальні ключі для тимчасових користувачів та отримувати сповіщення про відкриття або закриття дверей.

Багато моделей Yale мають модульну конструкцію, що дозволяє оновлювати функціональність замка шляхом встановлення різних комунікаційних модулів – Wi-Fi, Z-Wave, Zigbee або власних інтерфейсів. Це робить пристрої сумісними з широким спектром систем автоматизації та дає змогу адаптувати їх під особливості конкретної інфраструктури. Окрім основної функції дистанційного керування, замки забезпечують журналювання подій, фіксацію часу використання та ведення історії доступу, що є важливою складовою контролю безпеки.

Продукти Yale особливо орієнтовані на споживачів, яким потрібна зручність та можливість швидко інтегрувати замок у вже існуючу домашню екосистему. Більшість моделей оснащені датчиками положення, що дозволяють визначати, чи замкнені двері, або датчиками стану замка, які контролюють положення ригеля. Проте фокус компанії залишається насамперед на управлінні замком і надання доступу, тоді як повноцінна охоронна аналітика зазвичай делегується зовнішнім платформам розумного дому.

Yale також приділяє увагу безпеці комунікаційних протоколів: з’єднання шифрується, а механізми керування доступом захищені від простих способів несанкціонованого зчитування або підміни команд. Батарейне живлення з низьким енергоспоживанням та система попередження про низький заряд забезпечують стабільну роботу замків навіть без підключення до електромережі.

У підсумку Yale пропонує зрілі рішення у сфері розумних замків, що поєднують механічну надійність із сучасними цифровими можливостями. Продукти компанії орієнтовані на зручність користувача, інтеграцію з екосистемами розумного дому та забезпечення базового рівня контролю доступу з можливістю віддаленого керування та ведення історії подій.

1.5.3 Nuki

Nuki – це сучасна європейська платформа розумних замків, орієнтована на максимально просту інтеграцію в існуючу дверну інфраструктуру без потреби у заміні механічних елементів. Основна концепція бренду полягає у створенні універсального накладного смарт-замка, який встановлюється поверх стандартного європейського циліндра та здійснює керування ключем шляхом автоматизованого обертання. Такий підхід дозволяє перетворити будь-які традиційні двері на електронно керовані без втручання в їхню конструкцію.

Завдяки компактному форм-фактору та бездротовому способу монтажу Nuki здобула популярність у житловому секторі, зокрема серед користувачів квартир і приватних будинків. Замки Nuki підтримують керування через Bluetooth, Wi-Fi, а також можуть інтегруватися з платформами розумного дому, такими як Apple HomeKit, Google Home та Amazon Alexa. Через мобільний додаток користувач може дистанційно відчиняти або зачиняти замок, надавати гостьові доступи, створювати тимчасові цифрові ключі та переглядати журнал подій.

У своїй роботі Nuki реалізує функції автоматичного відчинення та автоматичного зачинення із затримкою, які базуються на геолокації та визначення наближення. Це робить використання замка зручним у повсякденному житті, дозволяючи користувачеві уникнути взаємодії з фізичними ключами. Журнали доступу фіксуються у хмарному середовищі й можуть містити інформацію про час, тип події та ідентифікатора користувача. За допомогою окремих модулів система може підтримувати навіть інтеграцію з Airbnb або іншими сервісами короткострокової оренди.

З точки зору технічної реалізації Nuki містить вбудовані сенсори положення, які дозволяють визначати поточний стан ригеля – замкнений він чи розблокований. Окремі моделі також оснащуються дверними сенсорами для визначення факту відчинення дверей. Проте основна увага приділяється зручності використання, мобільності та сумісності, а не складним охоронним сценаріям. Комунікація між замком, мостом та застосунком здійснюється із

застосуванням шифрування, що забезпечує базовий рівень захисту від перехоплення чи підміни команд.

Великим плюсом екосистеми Nuki є її модульність: окрім самого замка, користувач може додатково інтегрувати бездротову клавіатуру, датчик дверей, брелоки або хаби для розширення функціональності. Завдяки цьому система підходить для широкого спектра практичних сценаріїв – від індивідуального використання до керування орендними об’єктами або офісами.

Таким чином, Nuki пропонує інноваційний та зручний підхід до організації доступу, який поєднує мобільність, простоту встановлення та можливість гнучкого налаштування. Система орієнтована на користувачів, яким потрібна легка в розгортанні, адаптивна та сумісна з розумним домом платформа для керування фізичним доступом.

1.5.4 SmartThings

SmartThings – це універсальна платформа розумного дому, розроблена компанією Samsung, яка забезпечує інтеграцію, автоматизацію та централізоване керування широким спектром «smart» пристроїв. На відміну від суто охоронних або суто замкових рішень, SmartThings виступає в ролі екосистеми, здатної об’єднувати обладнання різних виробників, включаючи датчики, замки, камери, освітлення, побутову техніку, а також модулі автоматизації. Платформа підтримує бездротові протоколи Zigbee, Z-Wave та Wi-Fi, що робить її однією з найгнучкіших на ринку систем розумного дому.

Центральним компонентом є SmartThings Hub, який виконує роль координаційного вузла для всіх підключених пристроїв. Хаб приймає сигнали із сенсорів, віддає команди виконавчим механізмам та забезпечує взаємодію з хмарною інфраструктурою. Завдяки цьому він може керувати великим набором пристроїв, включаючи електронні замки різних виробників, датчики руху, геркони дверей, розумні розетки та інші модулі. Хмарна платформа Samsung дозволяє синхронізувати стан системи між різними пристроями користувача, створювати складні сценарії взаємодії та здійснювати віддалений контроль.

SmartThings підтримує широкий набір автоматизацій, що базуються на правилах «якщо-то», де події від одного пристрою можуть автоматично викликати реакції інших. Наприклад, відкриття дверей може ініціювати увімкнення світла, а зміна стану замка – переключення режиму сигналізації. Замки, інтегровані у SmartThings, можуть передавати дані про розблокування, використаний метод доступу та час події, що дозволяє вести базове журналювання та контролювати використання дверей. Деякі моделі замків також надають можливість створювати тимчасові цифрові ключі або керувати PIN-кодами через платформу.

Інтерфейс SmartThings доступний через мобільні додатки та інтегрується з голосовими асистентами Samsung Bixby, Google Assistant та Amazon Alexa. Завдяки цьому користувачі можуть керувати доступом і станом дверей як вручну, так і в межах комплексних сценаріїв автоматизації. Платформа також підтримує прив’язку до геолокації, що дозволяє автоматично виконувати певні дії на основі наближення чи віддалення користувача.

З погляду безпеки SmartThings забезпечує шифрування переданих даних та механізми аутентифікації, проте сама платформа не є охоронною системою у класичному сенсі: вона виступає швидше універсальним інтегратором, що пов’язує між собою різноманітні «розумні» пристрої й надає їм можливість взаємодії. Завдяки відкритій архітектурі та підтримці великої кількості виробників екосистема SmartThings охоплює широкий спектр рішень для побутового та частково комерційного використання, створюючи гнучку основу для автоматизації доступу, моніторингу та управління домашньою інфраструктурою.

У підсумку SmartThings є масштабованою платформою, здатною інтегрувати електронні замки, датчики та інші елементи розумного дому в єдину систему. Вона забезпечує зручний механізм автоматизації, базове журналювання доступу та широкі можливості для створення різнорівневих сценаріїв взаємодії між різними пристроями, що робить її важливим прикладом сучасного комерційного рішення у сфері домашньої автоматизації та контролю доступу.

Загалом результати огляду підтверджують, що сучасні системи контролю доступу знаходяться на етапі переходу від традиційних охоронних рішень до інтелектуальних платформ, здатних виконувати аналіз подій та виявляти складні інциденти. Проте існує потреба у подальшому розвитку підходів до збору та кореляції даних, які забезпечували б підвищений рівень точності, надійності та адаптивності систем безпеки.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА АНАЛІЗ АПАРАТНО-ПРОГРАМНИХ РІШЕНЬ

2.1 Аналіз можливих апаратних платформ

Для побудови системи збору й аналізу даних контролю доступу можливе використання кількох типів апаратних платформ: ESP32, Raspberry Pi та Arduino. Кожна з них має відмінні апаратні характеристики, обчислювальні можливості, підтримку мережевих інтерфейсів та програмних середовищ, що впливає на їхню придатність для задач реального часу та інтеграції з програмними сервісами.

2.1.1 Arduino

Arduino є найпростішою апаратною платформою серед розглянутих рішень і належить до класу мікроконтролерних систем, призначених насамперед для роботи з базовими сенсорами та виконавчими елементами. Плата функціонує на основі мікроконтролерів сімейства AVR або ARM Cortex-M, що забезпечують мінімально необхідний рівень обчислювальних ресурсів для виконання простих циклічних програм. Її архітектура не передбачає використання повноцінної операційної системи: програми виконуються у вигляді скетчів, що працюють у режимі однопоточного безперервного циклу. Це забезпечує передбачуваність поведінки, але суттєво обмежує можливість масштабування, паралельної обробки даних і реалізації складної логіки.

Arduino має широкий набір цифрових та аналогових входів/виходів, що робить її зручною платформою для початкових прототипів, лабораторних досліджень або побудови простих сенсорних вузлів. Проте можливості роботи з мережевими інтерфейсами істотно обмежені: базові плати не мають вбудованого Wi-Fi чи Ethernet, а додаткові модулі, наприклад, ESP8266 або Ethernet Shield забезпечують тільки мінімальний набір функцій, необхідних для передачі даних. Відсутність апаратної підтримки багатьох криптографічних алгоритмів ускладнює використання шифрування, що є критично важливим для систем безпеки. Також Arduino не підтримує сучасні мережеві рішення, такі як HTTPS,

SSL/TLS у повному обсязі, або складні мережеві бібліотеки, характерні для високорівневих платформ.

Через обмеженість обчислювальних можливостей Arduino не здатне обробляти великі обсяги даних, виконувати аналітичні операції чи працювати з високорівневими протоколами на кшталт REST API, WebSockets або MQTT у стабільному режимі. Підтримка промислових протоколів і стандартів, таких як Modbus RTU або Modbus TCP, також потребує додаткових бібліотек і часто є нестабільною через обмежену кількість пам’яті та недостатню швидкість мікроконтролера.

З огляду на це Arduino може ефективно виконувати роль периферійного модуля збору сигналів, наприклад, для зчитування стану простих датчиків або перетворення аналогових сигналів у цифрові. Однак використання цієї платформи як основного контролера системи, що потребує складної логіки прийняття рішень, інтеграції з серверною частиною, забезпечення криптографічного захисту чи обробки телеметрії у реальному часі, є недоцільним. Обмеження у надійності, швидкодії, мережевих можливостях і відсутність операційної системи роблять Arduino невідповідним вибором для побудови масштабованої системи контролю доступу.

2.1.2 ESP32

ESP32 є більш функціональною та технологічно розвиненою платформою порівняно з Arduino і належить до класу мікропроцесорних модулів зі значно ширшими можливостями у сфері мережевої взаємодії та периферійної інтеграції. В основі ESP32 – двоядерний процесор Tensilica LX6 або LX7, який працює на частоті до 240 МГц, забезпечуючи суттєво вищу продуктивність у порівнянні з класичними мікроконтролерами AVR. Пристрій має вбудовані модулі Wi-Fi та Bluetooth, що робить його популярним у проектах інтернету речей та бездротових сенсорних систем.

На відміну від Arduino, ESP32 підтримує багатопотоковість у межах FreeRTOS, що дозволяє виконувати одночасно кілька незалежних задач: опитування сенсорів, передавання даних, обробку подій і керування периферією.

Це розширює можливості платформи та робить її придатною для складніших сценаріїв. Завдяки апаратній підтримці Wi-Fi ESP32 здатна працювати з протоколами TCP/IP, а також міжпроцесними каналами комунікації, що дозволяє реалізовувати базові клієнтські або серверні мережеві сервіси.

Попри значні переваги, ESP32 має низку важливих обмежень, які варто враховувати при розробці систем контролю доступу. Хоча платформа підтримує протоколи MQTT, WebSockets та прості HTTP-запити, її ресурси не забезпечують стабільну роботу складних стеків, таких як повноцінні REST API з обробкою великих обсягів даних, SSL/TLS у промисловій якості або криптографічні функції з інтенсивними обчисленнями. Підтримка шифрування на ESP32 є можливою, але часто призводить до зростання затримок або нестабільності на високому навантаженні. Це суттєво обмежує використання модуля у системах, що потребують високого рівня інформаційної безпеки.

Ще однією проблемою є обмеження в обсязі оперативної та флеш-пам’яті. Навіть у розширених версіях ESP32 обсяг RAM набагато менший, ніж у повноцінних одноплатних комп’ютерів, тому можливість виконання складних алгоритмів обробки подій, буферизації телеметрії або аналізу аномалій на краю мережі є обмеженою. Платформа не призначена для роботи з великими або нестандартними структурами даних, що також ускладнює реалізацію розвинених механізмів журналювання.

Інтеграція з промисловими протоколами, наприклад Modbus RTU через RS-485, можлива, але потребує додаткових апаратних модулів та спеціальних бібліотек. Стабільність такої інтеграції часто залежить від коректності налаштувань, а обробка великих або високочастотних потоків даних може призвести до перевантаження пристрою. ESP32 не має повноцінної операційної системи загального призначення, що робить неможливим запуск зрілих сервісів, контейнерів або розподілених модулів обробки подій.

Незважаючи на ці обмеження, ESP32 залишається потужною платформою для створення компактних сенсорних вузлів, бездротових датчиків, точок збору подій або низькорівневих комунікаційних модулів. Проте для системи, що

потребує надійної логіки керування, інтеграції з REST API-серверами, складної обробки подій чи роботи з великими журналами, ESP32 не може виступати основним контролером. Її оптимальна роль – периферійний інтелектуальний сенсорний модуль, а не центральний обчислювальний вузол.

2.1.3 Raspberry Pi

Raspberry Pi (рис. 2.1) належить до класу одноплатних комп'ютерів і є значно потужнішою платформою порівняно з мікроконтролерними рішеннями типу Arduino чи ESP32. На відміну від них, Raspberry Pi оснащена повноцінним процесором ARM із підтримкою багатоядерності, значним обсягом оперативної пам'яті та можливістю запуску операційної системи загального призначення – зазвичай Linux. Це кардинально розширює можливості платформи й робить її придатною для реалізації високорівневих програмних стеків, серверних компонентів та складних алгоритмів обробки даних.

Рисунок 2.1 – Вигляд комп'ютера Raspberry Pi [16]

Однією з ключових переваг Raspberry Pi є те, що вона підтримує той самий програмний інструментарій, який застосовується на повноцінних серверах: .NET 8, Python, Node.js, Docker-контейнери, системні служби Ubuntu/Debian, роботу з SQL-серверами, SSL/TLS, сучасні мережеві протоколи та криптографічні модулі.

Завдяки цьому розробник може реалізувати на ній локальні REST API-сервіси, модулі обробки подій, буферизацію журналів, алгоритми аномалій і навіть edge-аналітику без потреби у зовнішніх високопродуктивних серверах.

Raspberry Pi також має розвинені можливості апаратної взаємодії: GPIO для роботи з сенсорами, SPI та I²C для периферії, UART-інтерфейси, які дозволяють реалізовувати RS-232 та RS-485 через відповідні адаптери. Це робить платформу сумісною з промисловими стандартами, такими як Modbus RTU або ASCII-протоколи електронних замків, що важливо для побудови систем контролю доступу. На відміну від ESP32, Raspberry Pi здатна стабільно обробляти високочастотні потоки даних, виконувати паралельні задачі й підтримувати великі обсяги телеметрії без ризику перевантаження.

Наявність повноцінної операційної системи відкриває можливість багатозадачності: Raspberry Pi може одночасно управляти пристроями, отримувати дані по RS-485, передавати події у серверну частину через REST API, логувати дані у локальну базу, працювати з криптографічними модулями та перевіряти цілісність пакетів. Платформа дозволяє розгортати внутрішні сервіси для обробки черг подій, а у разі тимчасової втрати зв’язку – забезпечувати буферизацію даних і гарантовану доставку їх на центральний сервер.

Ще однією перевагою є наявність значно ширшої екосистеми бібліотек та підтримка великих спільнот розробників. Практично будь-яке завдання – від реалізації MQTT або Modbus до побудови REST API або підключення SQL Server – має готові інструменти з відкритим кодом, які можуть бути адаптовані для потреб системи. Raspberry Pi також відзначається стабільністю, можливістю автоматизації, підтримкою оновлень та створенням резервних копій системи.

Попри свої переваги, Raspberry Pi має певні обмеження: це не промисловий контролер, тому її робота залежить від стабільності карти пам’яті, якості живлення та температурних умов. Вона менш стійка до екстремальних умов та електромагнітних завад, ніж спеціалізовані промислові PLC-контролери. Проте у контексті розробки інтелектуальної системи збору та обробки подій контролю

доступу Raspberry Pi є оптимальним компромісом між функціональністю, продуктивністю та вартістю.

Узагалі Raspberry Pi виступає найсильнішою платформою серед розглянутих у цьому розділі завдяки підтримці повноцінної ОС, високорівневого програмування, стабільної мережевої взаємодії, сумісності з промисловими протоколами та можливістю виконання складної аналітики на краю мережі. Саме ці характеристики роблять її доцільною основою для побудови системи збирання та обробки подій контролю доступу.

2.2 Аналіз сенсорів для визначення стану дверей

Для формування достовірних подій контролю доступу важливо мати незалежний сенсорний канал, який підтверджує фактичний стан дверей. Існує кілька типів датчиків, що застосовуються для цього завдання: герконові датчики, оптичні сенсори та датчики вібрації.

Герконові датчики є найбільш поширеним рішенням (рис. 2.2). Вони реагують на положення магніту та дозволяють визначати два стани: «двері зачинені» та «двері відкриті».

Рисунок 2.2 – Вигляд герконового датчика [17]

Геркони відзначаються високою надійністю, низькою вартістю, простотою інтеграції з RS-інтерфейсами або GPIO Raspberry Pi, стійкістю до перешкод та

мінімальною затримкою реакції. Їхній основний недолік – бінарність, тобто вони не фіксують ступінь відкриття чи силу впливу.

Оптичні датчики забезпечують точніші вимірювання, зокрема можуть визначати неповне відкриття, наявність перешкоди або швидкість руху дверей. Однак вони дорожчі, більш чутливі до забруднень та умов освітлення, вимагають складнішої обробки сигналу і часто потребують додаткового калібрування. Застосовуються переважно у складних або спеціалізованих системах.

Датчики вібрації здатні фіксувати удари, спроби злому або силового впливу. Вони не інформують про положення дверей, але додають додатковий канал безпеки. Вібраційні сенсори ефективні у сценаріях виявлення аномальних дій, проте вимагають алгоритмів фільтрації хибних спрацювань.

Для базової системи збору та аналізу подій добре підходить герконовий датчик, оскільки він простий, надійний, економічний і забезпечує достатню інформацію для фіксації стану дверей. У поєднанні з аналітикою на сервері геркон дає все необхідне для коректної кореляції подій замка та дверей.

2.3 Протоколи передавання даних та архітектура взаємодії

Система потребує механізму передачі подій і телеметрії від Raspberry Pi до серверної частини. Найчастіше для цього використовуються три протоколи: MQTT, REST API та WebSockets.

MQTT – легковаговий брокерський протокол публікації та підписки, добре підходить для IoT-пристроїв та систем з високою частотою подій. Проте він потребує окремого брокера, складнішої інфраструктури та додаткових механізмів авторизації.

REST API є універсальним, простим у реалізації, підтримується .NET «з коробки» і дає змогу ефективно передавати події, телеметрію, журнали та здійснювати запити на стан системи. REST підходить для систем, де події надходять не надто часто, але потребують надійності та журналювання на рівні транспорту й сервера.

WebSockets забезпечують двосторонній канал у реальному часі та можуть використовуватися для живих оновлень стану в адміністраторській панелі. Це хороше рішення для інтерфейсу Blazor або для push-сповіщень, але надмірне для первинного збору даних.

Для передачі подій і телеметрії обраним рішенням є REST API на платформі .NET, оскільки воно забезпечує просту інтеграцію з Blazor, SQL Server, надійність і контрольованість потоку даних. WebSockets можуть бути використані додатково для онлайн-відображення подій на панелі моніторингу.

2.4 Засоби обробки та зберігання подій

Система має забезпечувати довготривале зберігання журналів подій, їхню кореляцію та можливість подальшого аналізу. Для цього розглянуто PostgreSQL та SQLite.

SQLite є легкою embedded-базою даних, підходить для локальних систем та невеликих обсягів інформації, не потребує серверної інфраструктури, але має обмеження у масштабованості, одночасному доступі та продуктивності при великих потоках подій.

PostgreSQL є промисловою СУБД із широкими можливостями оптимізації, підтримкою складних типів даних, транзакцій, індексації та високих навантажень. Підходить для систем, де важливі стабільність, аналітика та можливість роботи з великими обсягами журналів.

Моделі даних повинні забезпечувати зберігання подій, телеметрії, кореляцій замка та дверей, тимчасових інтервалів, а також забезпечувати швидкий пошук подій за часом та типом.

Вимоги до системи:

– атомарність записів;

– незмінність журналів;

– часові мітки високої роздільності;

– можливість горизонтального масштабування.

У рамках цієї роботи доцільним є використання PostgreSQL або SQL Server.

Обраний технологічний стек – Raspberry Pi як апаратна платформа, геркон як сенсор стану дверей, REST API як канал передавання даних та SQL Server як система зберігання – забезпечує надійну основу для реалізації системи контролю доступу, орієнтованої на точність, стабільність і можливість подальшого розвитку.

РОЗДІЛ 3
ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Архітектура системи

Архітектура розробленої системи збирання та обробки даних для підвищення безпеки базується на принципах модульності, розподіленості та забезпечення надійної взаємодії між апаратними й програмними компонентами. Система складається з кількох ключових підсистем: апаратного контролера на базі Raspberry Pi, сенсорного модуля визначення стану дверей, електронного замка, серверної частини з REST API, реляційної бази даних SQL Server та веб-інтерфейсу адміністрування, реалізованого на Blazor. Така побудова забезпечує чітке логічне розмежування функцій і створює умови для масштабування та подальших розширень системи.

3.1.1 Загальна структурна схема системи

Усі компоненти системи взаємодіють між собою у межах розподіленої клієнт-серверної архітектури. На нижньому рівні знаходиться Raspberry Pi, що виконує роль контролера, який безпосередньо працює із сенсорами та виконавчими пристроями. Контролер відповідає за збір подій, формування телеметрії та передачу цих даних на сервер. На наступному рівні розташований серверний застосунок на платформі .NET, який приймає події від контролера, перевіряє їх валідність, зберігає у базі даних та виконує додаткову логіку обробки. Фронтендна частина системи забезпечує можливість моніторингу й аналізу подій у реальному часі через веб-інтерфейс.

У загальному вигляді архітектура складається з таких основних модулів:

– Raspberry Pi – збір подій, взаємодія з сенсорами, передавання даних на сервер;

– сенсорний модуль – герконовий датчик, що визначає фактичне положення дверей;

– електронний замок – виконавчий пристрій, що підтримує індикацію свого стану або взаємодію через послідовний інтерфейс;

– REST API-сервер – централізований вузол прийому даних, реалізований на платформі .NET;

– SQL Server – база подій, телеметрії та станів пристроїв;

– адміністративний інтерфейс побудований на технології Blazor WebAssembly/Server – модуль перегляду, аналізу та візуалізації даних.

Ця структурна схема забезпечує логічне розділення обов’язків між компонентами, що є важливою вимогою для надійності систем безпеки.

3.1.2 Raspberry Pi як контролер

Raspberry Pi виступає основним обчислювальним вузлом системи, забезпечуючи взаємодію з апаратними компонентами та реалізацію початкової логіки обробки подій. Завдяки повноцінній операційній системі Raspberry Pi OS контролер може виконувати:

– роботу з GPIO для зчитування стану сенсорів;

– обмін даними з електронним замком через UART/RS-232/RS-485;

– локальне журналювання у випадку відсутності зв’язку;

– перетворення низькорівневих подій у структуровані повідомлення;

– передачу подій і телеметрії на сервер через REST API;

– запуск сервісів або демонів на платформі .NET.

Контролер працює як edge-вузол, що виконує попередню обробку даних, зменшуючи навантаження на сервер та забезпечуючи роботу системи навіть за умов нестабільного мережевого підключення.

3.1.3 Герконовий датчик та електронний замок

Сенсорний модуль представлений герконовим датчиком, який монтується на дверну коробку та фіксує два основних стани: «двері зачинені» та «двері відкриті». Геркон надає надійний бінарний сигнал без затримок і практично не схильний до хибних спрацювань. Його використання є критично важливим для коректної інтерпретації подій, оскільки стан дверей не завжди може бути достовірно визначений самим замком.

Електронний замок забезпечує стан locked/unlocked, може інформувати контролер про поточний режим або приймати керуючі команди. У випадку

роботи через RS-232 або RS-485 замок може періодично надсилати власні телеметричні повідомлення, які Raspberry Pi фіксує та передає на сервер.

3.1.4 Зв’язок із сервером через REST API

Передавання подій і телеметрії від Raspberry Pi до серверної частини здійснюється через REST API поверх HTTP(S). Такий спосіб комунікації має низку переваг:

– простота та стандартизованість протоколу;

– можливість використання безпечного каналу HTTPS;

– чітка структура запитів і відповідей;

– хороша інтеграція з .NET-серверами;

– можливість контролювати обсяг, частоту та формат переданих даних.

REST API дозволяє реалізувати такі маршрути:

– POST /events – передавання подій;

– POST /telemetry – поточний технічний стан пристроїв;

– GET /device/{id}/state – запит стану контролера;

– POST /heartbeat – життєвий сигнал для контролю доступності Raspberry Pi.

Таким чином забезпечується надійна двостороння взаємодія та контроль цілісності даних.

3.1.5 Серверний застосунок на .NET

Серверна частина побудована на платформі .NET 8, що забезпечує високу продуктивність, можливість масштабування та зручний інструментарій для роботи з базами даних і веб-сервісами. Основними функціями серверного застосунку є:

– приймання подій від контролера;

– валідація, нормалізація та обробка вхідних даних;

– зберігання подій у SQL Server;

– обробка телеметрії та контроль доступності пристроїв;

– формування зведених записів у журналах;

– надання API для веб-інтерфейсу моніторингу.

На цьому рівні реалізовано бізнес-логіку, яка забезпечує зв’язок між станом замка та реальним положенням дверей, формуючи повноцінну картину подій.

3.1.6 База даних SQL Server

Для зберігання подій, технічних даних і користувацької інформації використовується одна із найсучасніших СУБД розроблена компанією Microsoft – SQL Server, який забезпечує:

– повноцінну транзакційність;

– індексацію для прискореного пошуку;

– зберігання великої кількості подій з мінімальними затримками;

– можливість побудови аналітичних запитів;

– збереження журналу транзакцій;

– високий рівень надійності та відмовостійкості.

Модель даних включає кілька основних таблиць: Events, DoorState, LockState, Telemetry, Devices, що дає змогу повністю реконструювати історію доступу.

3.1.7 Веб-інтерфейс на Blazor

Blazor виступає інструментом для створення зручного веб-інтерфейсу моніторингу, який дозволяє:

– переглядати повний журнал подій;

– аналізувати часові послідовності;

– переглядати стан дверей і замка в реальному часі;

– контролювати роботу контролера;

– візуалізувати телеметрію.

Використання Blazor дозволило реалізувати єдиний технологічний стек на .NET як на сервері, так і у фронтенді, що спрощує підтримку та розвиток системи.

3.2 Апаратна частина

Апаратна частина системи є фундаментом її надійності, оскільки саме на цьому рівні здійснюється зчитування фактичних фізичних станів дверей і замка та перетворення їх у цифрові сигнали для подальшої програмної обробки. Вибір компонентів, способи їхнього підключення та особливості монтажу безпосередньо впливають на точність, швидкість реакції та стабільність роботи всієї системи контролю доступу. У цьому підрозділі розглянуто схему підключення герконового датчика, організацію взаємодії контролера з електронним замком через RS-232/RS-485, вимоги до живлення та рекомендації щодо монтажу апаратної частини.

3.2.1 Схема підключення герконового датчика

Герконовий датчик використовується для визначення фактичного стану дверей – відкриті вони чи зачинені. Стандартна схема підключення передбачає:

– контактний елемент з герметично запаяними магніточутливими контактами;

– магніт, який встановлюється на дверях;

Підключення до Raspberry Pi здійснюється через USB.

3.2.2 Електронні замки, що використовуються в системі, можуть підтримувати послідовні інтерфейси RS-232 або RS-485, а також працювати за протоколом Modbus RTU. Конкретний інтерфейс залежить від моделі замка, однак принципи інтеграції залишаються подібними. У найпростішому випадку замок підключається через RS-232 за допомогою USB-RS232 перехідника, оскільки Raspberry Pi не має апаратного порту RS-232. У такій конфігурації замок обмінюється даними через порт /dev/ttyUSB0 і може надсилати свій поточний стан, повідомляти про помилки механізму або приймати команди зміни положення ригеля. Комунікація налаштовується відповідно до параметрів, вказаних у специфікації замка: швидкість, кількість стоп-бітів, тип парності тощо.

Для більш складних об’єктів або там, де необхідні великі відстані між контролером та виконавчим пристроєм, доцільним є використання RS-485. Підключення виконують через USB-RS485 адаптер із диференційною парою проводів A/B. RS-485 забезпечує високу перешкодостійкість, підтримує топологію «шинної» лінії та дозволяє розташовувати замки на відстані до 1200 метрів від контролера. У цьому випадку замок може використовувати власний протокол або працювати за стандартом Modbus RTU (рис. 3.1).

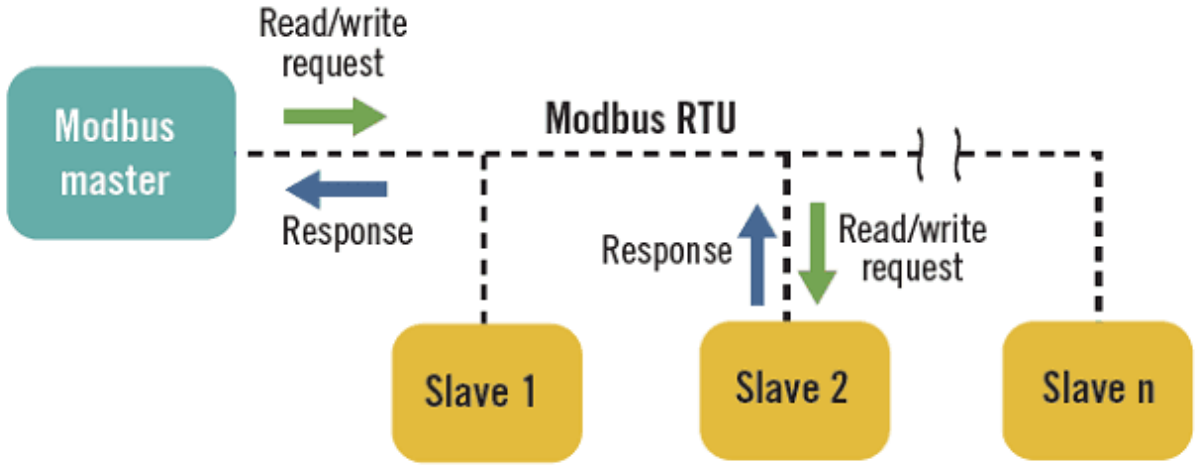

Рисунок 3.1 – Схема взаємодії по протоколу modbus [18]

У поточній реалізації замок розглядається як Modbus slave-пристрій, а Raspberry Pi – як master, що дозволяє опитувати замок у стандартизований спосіб. Стан замка передається у вигляді цілого числа, де кожен біт відповідає одному входу пристрою. Оскільки контролер підтримує до 12 замків, їхні стани кодуються у вигляді 12-бітового значення від 0 до 4095, яке Raspberry Pi зчитує одним запитом до holding-регістра, після чого розкладає байти на біти де кожен біт відповідає за стан конкретного замка. Це дозволяє за один цикл опитування отримувати повний набір станів, визначати будь-які зміни за допомогою бітових операцій та формувати події з мінімальною кількістю переданих даних.

Використання Modbus забезпечує універсальність, простоту інтеграції та доступність готових .NET-бібліотек, що робить його оптимальним вибором для

системи контролю доступу, де один контролер обслуговує декілька замків одночасно.

3.3 Програмне забезпечення Raspberry Pi

Програмне забезпечення контролера реалізоване у вигляді фонової служби .NET 8, яка працює під керуванням Raspberry Pi OS. Сервіс автоматично запускається при старті системи та забезпечує зчитування стану замка, аналіз стану дверей, формування подій і їх передавання на серверну частину через REST API. Використання технології Worker Service дозволяє розробити автономний, надійний і відмовостійкий компонент, що відповідає вимогам систем реального часу.

3.3.1 Програмне середовище та структура служби

Сервіс запускається як systemd-unit, що являє собою службу в операційних системах типу Linux [19] і складається з основного робочого циклу Worker продемонстрованого в лістингу 3.1, який виконує періодичне опитування пристроїв і обробляє події. Це створює службу, яка буде виконуватися безперервно.

Лістинг 3.1 – Приклад фрагмента коду Program.cs

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

Host.CreateDefaultBuilder(args)
    .UseSystemd()
    .ConfigureServices(services =>
    {
        services.AddHostedService<ControllerWorker>();
        services.AddHttpClient("api", client =>
        {
            client.BaseAddress = new Uri("https://server-address/api/");
        });
    })
    .Build()
    .Run();
```

кінець лістингу 3.1

3.3.2 Зчитування стану електронного замка через SerialPort

Стан замка зчитується з USB-UART адаптера, який Raspberry Pi визначає як /dev/ttyUSB0. Замок може надсилати текстові або у нашому випадку бінарні повідомлення. Приклад відкриття порту реалізовано у лістингу 3.2.

Лістинг 3.2 – Приклад зчитування статусу замка

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;

Host.CreateDefaultBuilder(args)
    .UseSystemd()
    .ConfigureServices(services =>
    {
        services.AddHostedService<ControllerWorker>();
        services.AddHttpClient("api", client =>
        {
            client.BaseAddress = new Uri("https://server-address/api/");
        });
    })
    .Build()
    .Run();
```

кінець лістингу 3.2

3.3.3 Зчитування стану дверей через USB-сенсор

Геркон підключається через USB-інтерфейс комп’ютера, комунікація відбувається через віртуальний послідовний порт за рахунок перетворювача сигналу з USB на сигнали типу RS-232 і навпаки, який передає простий сигнал open, або closed, що отримується з при з’єднанні, або роз’єднанні магнітних частин герконового датчика. Приклад зчитування стану дверей отриманого з геркона, або ж магнітного датчика продемонстровано на лістингу 3.3.

Лістинг 3.3 – Приклад фрагмента коду зчитування датчика геркона

```
public class DoorSensor
{
    private readonly SerialPort _port;

    public DoorSensor(string portName = "/dev/ttyUSB1")
    {
        _port = new SerialPort(portName, 9600);
        _port.Open();
    }

    public string ReadDoor()
    {
        try
        {
            return _port.ReadLine().Trim();
        }
        catch
        {
            return "UNKNOWN";
        }
    }
}
```

кінець лістингу 3.3

Якщо геркон підключено через Arduino як USB-перетворювач – це саме те, що Arduino буде надсилати.

3.3.4 Формування подій

Сервіс реалізований у лістингу 3.4 формує об’єкт події відкриття, або ж закриття замка, а також з’єднання, або роз’єднання магнітного датчика, який згодом надсилається на сервер за допомогою технології REST API, що базується на протоколі передачі даних HTTP(S), який в свою чергу побудований на протоколі передачі даних TCP, що гаранту цілісність доставленої інформації та інформування про це клієнта, що її надіслав [20].

Лістинг 3.4 – Приклад об’єкта події

```
public class EventRecord
{
    public string DeviceId { get; set; }
    public string EventType { get; set; }
    public DateTime Timestamp { get; set; }
    public string Value { get; set; }
}
```

кінець лістингу 3.4

Метод формування події, що буде передаватися на сервер для подальшої обробки реалізовано всередині класу Worker зображено на лістингу 3.5.

Лістинг 3.5 – Приклад генерації події

```
private EventRecord CreateEvent(string type, string value)
{
    return new EventRecord
    {
        DeviceId = "Pi-01",
        EventType = type,
        Value = value,
        Timestamp = DateTime.UtcNow
    };
}
```

кінець лістингу 3.5

Передавання подій на сервер через REST API при якому контролер надсилає сформований запис події методом POST за маршрутом events, використовуючи серіалізацію в JSON. Клас HttpClient виступає клієнтом для асинхронної комунікації із сервером, а метод SendEventAsync повертає логічне значення, що вказує на успішність операції. Якщо сервер відповідає статусом 2xx, подія вважається успішно переданою; у протилежному випадку вона може бути додана до локального буфера для повторної спроби відправлення. Цей

підхід забезпечує надійність обміну даними та стійкість системи до короткочасних перебоїв зв’язку, код у лістингу 3.6.

Лістинг 3.6 – Приклад передачі подій на сервер

```
private readonly HttpClient _client;

public async Task<bool> SendEventAsync(EventRecord record)
{
    var response = await _client.PostAsJsonAsync("events", record);

    return response.IsSuccessStatusCode;
}
```

кінець лістингу 3.6

Оскільки контролер підтримує підключення до 12 окремих замків, їхні стани доцільно зчитувати не як окремі значення, а у вигляді одного цілого числа. У цьому числовому представленні кожен біт відповідає одному замку: якщо біт дорівнює 1 – замок перебуває в замкненому, якщо 0 – у відчиненому. Таким чином, усі 12 замків можуть бути закодовані у 12-бітове число від 0 до 4095. Це дозволяє значно зменшити обсяг переданих даних та значно спростити логіку визначення змін: достатньо порівняти поточне числове значення з попереднім і за допомогою бітових операцій визначити, який саме замок змінив стан. Зчитування виконується через інтерфейс Modbus RTU по RS-485, де контролер передає запит на читання регістра, у якому зберігається бітове представлення станів усіх замків. У відповідь пристрій повертає два байти, які разом формують ціле число, що містить повну інформацію про 12 входів.

На лістингу 3.7 наведено приклад того, як може виглядати читання стану замків через Modbus та перетворення отриманих даних на число, де кожен біт відповідає одному замку:

Лістинг 3.7 – Приклад читання даних через Modbus

```
public class MultiLockReader
{
    private readonly ModbusClient _client;
    public MultiLockReader(string port)
    {
        _client = new ModbusClient(port)
        {
            Baudrate = 9600,
            Parity = Parity.None,
            StopBits = StopBits.One
        };
        _client.Connect();
    }
    // Зчитуємо 12 станів одним числом (0–4095)
    public int ReadCombinedState()
    {
        // Читаємо один holding-регістр, де зберігаються біти станів
        // Наприклад, регістр 40001
        ushort[] data = _client.ReadHoldingRegisters(0, 1);
        // Modbus повертає ushort, який вже є 16-бітовим числом.
        // Потрібні нам лише перші 12 біт.
        int state = data[0] & 0x0FFF;
        return state;
    }
}
```

кінець лістингу 3.7

Якщо, наприклад, число дорівнює 37, то його двійкове представлення 0000 0010 0101 означає, що замки з індексами 0, 2 і 5 активні, а решта – ні. Зміна будь-якого замка приводить до зміни відповідного біта. Такий метод є надійним, швидким і легко масштабованим, що робить його оптимальним для систем, у яких один контролер обслуговує багато входів одночасно [1].

3.4 Серверна частина

Серверна частина системи реалізована на платформі .NET 8 і виконує функції централізованого приймання та обробки даних, що надходять від контролера Raspberry Pi. Взаємодія між пристроєм і сервером здійснюється через REST API, що забезпечує просту та стандартизовану передачу подій. Основні маршрути включають /events для отримання подій зміни стану дверей і замка, /telemetry для передавання технічного стану контролера, /device-state для отримання актуального стану пристрою та /heartbeat для підтвердження його активності. Перед доступом до будь-якого маршруту сервер перевіряє API-ключ, який контролер надсилає у заголовку запиту; це забезпечує базову авторизацію і захист від сторонніх джерел.

Після отримання події сервер виконує її валідацію, перевіряє коректність часової мітки та відповідність логічним правилам роботи системи контролю доступу. Далі подія зберігається у SQL Server, де формується повний журнал роботи замка, дверей і контролера. Сервер також оновлює останній відомий стан пристрою, щоб забезпечити швидке відображення його активності у веб інтерфейсі. У разі неможливості опрацювати подію сервер повідомляє про помилку, що дозволяє Raspberry Pi повторно надіслати її або записати у локальний буфер. Завдяки такій організації серверна частина забезпечує надійну, передбачувану та безперервну обробку інформації, яка є основою для подальшого аналізу та відображення даних у системі.

3.5 Модель даних та база даних

База даних є центральним елементом серверної частини системи, оскільки забезпечує зберігання повної історії подій та технічних станів контролера, необхідних для подальшого аналізу та відтворення послідовності дій. Для цих потреб використовується SQL Server, який надає широкі можливості для індексації, оптимізації запитів, транзакційної цілісності та ведення журналів.

Модель даних побудована таким чином, щоб відображати як дискретні події, що надходять від контролера, так і довготривалі стани пристроїв.

Основною таблицею є Events, у якій зберігаються всі події, пов’язані зі зміною стану дверей, замка або стану контролера. Кожен запис містить унікальний ідентифікатор, тип події, час її виникнення, ідентифікатор пристрою та допоміжні дані. Поруч із нею функціонують таблиці DoorState та LockState, що зберігають останні зафіксовані стани відповідно дверей і замка; вони використовуються для швидкого отримання актуальної інформації без необхідності перегляду всього журналу подій. Таблиця Telemetry призначена для технічних параметрів, таких як напруга живлення, помилки зв’язку, температура контролера чи стан послідовного порту. Таблиця Device містить інформацію про контролери в системі, включно з часом їхньої останньої активності, статусом онлайн/офлайн та ключами доступу. Для адміністрування передбачено таблицю Users, яка зберігає облікові записи, ролі й параметри доступу до веб інтерфейсу.

Для забезпечення ефективного доступу й швидкої обробки даних у моделі використовуються індекси за часовими мітками подій, а також індекси за ідентифікаторами пристроїв, що дозволяє швидко робити вибірки за конкретним контролером або часовим проміжком. Часові мітки зберігаються у форматі UTC для уникнення проблем із часовими зонами й забезпечення коректної хронології. Цілісність даних підтримується через зовнішні ключі між Events, Telemetry та Device, а також завдяки використанню транзакцій під час збереження складних або взаємопов’язаних подій. SQL Server забезпечує незмінність журналу, що є критично важливим у системах безпеки, оскільки дає змогу точно відтворити всі дії та стани в хронологічному порядку.

Таким чином, модель даних побудована так, щоб забезпечити одночасно високу продуктивність, масштабованість і надійність системи, а також можливість подальшого аналізу, виявлення проблем чи реконструкції інцидентів на основі повного журналу подій.

3.6 WEB-інтерфейс

WEB-інтерфейс моніторингу є важливою складовою системи, оскільки забезпечує зручний доступ до журналу подій, відображення актуального стану контролера та візуалізацію роботи системи в реальному часі. Для реалізації інтерфейсу використано технологію Blazor, яка дозволяє створити інтерактивний веб застосунок на основі .NET без використання зовнішніх JavaScript-фреймворків. Завдяки цьому усі частини системи – сервер, контролер і веб інтерфейс – використовують єдиний технологічний стек, що спрощує підтримку та подальший розвиток.

Головним елементом інтерфейсу є сторінка перегляду подій, на якій користувач може бачити повний журнал змін станів дверей, замка та технічних повідомлень контролера. Дані відображаються у вигляді таблиці з сортуванням, пагінацією та можливістю фільтрації за типом події, пристроєм або часовим проміжком. Це дозволяє швидко знаходити потрібну інформацію та аналізувати роботу системи в окремих ситуаціях. Для покращення зручності реалізовано можливість динамічного пошуку та автооновлення списку подій у режимі реального часу.

Окремий компонент інтерфейсу присвячений відображенню актуального стану дверей і замка. Він використовує періодичні запити або WebSockets для отримання останнього стану з сервера та показує користувачу, чи відкриті двері, чи заблокований замок і коли відбулася остання зміна. Це забезпечує можливість швидкого виявлення аномальних ситуацій або некоректних послідовностей дій без необхідності переглядати повний журнал.

Для наочності роботи системи веб інтерфейс доповнений елементами візуалізації. Зокрема, графіки та діаграми дозволяють аналізувати динаміку подій, час відкриття дверей, частоту використання замка чи періоди збоїв зв’язку. Завдяки таким інструментам адміністратор може оцінювати тенденції та робити висновки щодо стабільності та ефективності системи у реальних умовах експлуатації.

Загалом веб інтерфейс створює зручне та інтуїтивно зрозуміле середовище для моніторингу роботи всієї системи, забезпечуючи повний доступ до журналів, актуальних даних і аналітичної інформації без необхідності взаємодії з низькорівневими інструментами або серверною частиною.

3.7 Проблематика дослідження та значення точного відображення стану системи

Точне визначення реального стану дверей і механізму замикання є критично важливим елементом будь-якої системи контролю доступу, оскільки саме на основі цих даних формується уявлення про фактичні дії користувача та фізичну безпеку об’єкта. Навіть незначні неточності у фіксації моменту відкриття або закриття дверей можуть призвести до неправильних висновків щодо ситуації, що відбувалась у реальному світі: створити помилкове враження про несанкціонований доступ, приховати спробу злому або викликати хибне спрацьовування систем безпеки. У багатьох випадках різниця у декілька секунд чи одна неврахована подія здатні змінити інтерпретацію всієї послідовності дій, тому повнота та достовірність даних мають принципове значення.

Використання лише одного джерела даних, наприклад електронного замка або тільки герконового датчика, не дає повної гарантії достовірності інформації. Замок може сигналізувати про перехід у стан «Locked», навіть якщо двері фізично залишаються прочиненими, а геркон – через магнітні перешкоди чи механічні неточності – інколи фіксує хибне спрацювання. Тому дослідження взаємної узгодженості цих двох каналів даних є необхідним для формування цілісної картини того, що відбувається з дверима в реальності. Важливо розуміти, наскільки сенсорні дані дійсно відображають фактичний фізичний стан, і чи можуть вони забезпечити надійну інформацію для подальшого аналізу.

Саме тому ключовою проблемою цього дослідження є оцінка точності, з якою система, побудована на основі електронного замка та герконового датчика, здатна відтворювати реальні події. Коректне відображення станів є фундаментом

безпеки, адже від нього залежить правильність журналювання, ефективність виявлення аномалій та здатність системи однозначно інтерпретувати ситуації, що відбувалися з об’єктом доступу. Без достовірних і узгоджених даних система контролю доступу втрачає інформаційну цінність і не може виконувати своїх функцій у повному обсязі. Тому дослідження точності та узгодженості даних є не лише технічним завданням, а й необхідною умовою для підвищення надійності та довіри до системи безпеки в цілому.

3.8 Аналіз типових реальних ситуацій, що потребують перевірки

Для оцінки точності відображення фактичного стану дверей і замка необхідно проаналізувати низку характерних реальних ситуацій, що регулярно виникають під час експлуатації систем контролю доступу. Ці ситуації не є штучними або лабораторними – навпаки, саме вони найчастіше трапляються у повсякденних умовах та визначають, наскільки отримані сенсорні дані дозволяють сформувати достовірну хронологію подій. Правильне відтворення фізичної поведінки об’єкта доступу залежить від того, наскільки електронний замок та геркон узгоджують між собою інформацію про стан дверей, і чи дозволяє система правильно інтерпретувати нестандартні або складні сценарії.

Однією з найпоширеніших ситуацій є випадок, коли двері фізично зачинені, але механізм замка з певних причин не перейшов у стан блокування. Це може трапитися через механічні перешкоди, недостатню амплітуду руху ригеля, перекоси конструкції або навіть через неточне встановлення замка. У такій ситуації замок може залишати стан «Unlocked», хоча для користувача двері виглядають цілком зачиненими. Для системи це створює потенційно небезпечний сценарій, адже подальша оцінка доступу буде базуватися на помилковому припущенні про незаблокований стан.

Зворотна ситуація – замок сигналізує «Locked», хоча двері залишаються прочиненими або не притягнутими до кінця. Геркон у цьому випадку залишається у стані «Open», що суперечить повідомленню замка. Поява таких

розбіжностей є типовою для багатьох комерційних систем і часто становить основу для подальшого аналізу аномалій. Саме ці випадки дозволяють оцінити, наскільки система здатна правильно інтерпретувати реальний стан, коли дані не узгоджуються між собою.

Окремої уваги потребує ситуація із затримкою спрацювання геркона. Магнітні датчики можуть давати короткочасні або запізнілі зміни стану через вібрації дверей, неточність положення магніта, або інерційні коливання у момент удару дверей об коробку. У таких випадках замок може вже перейти у стан «Locked», тоді як геркон ще не зафіксував «Closed». Ці невеликі часові розбіжності формують логічні розриви, що можуть вплинути на коректність журналів подій. Важливо оцінити, чи здатна система інтерпретувати такі розриви як допустиму похибку, чи класифікує їх як потенційну аномалію.

Ще одним сценарієм є ситуація, коли замок повідомляє стан «Closed» або «Locked», але геркон сигналізує про «Open». Такі «хрестоподібні» конфлікти між сигналами можуть виникати через зовнішні магнітні поля, механічний шум або часткове роз’єднання конструкції. Аналіз цього типу випадків дозволяє визначити, яке джерело даних є більш надійним та чи можна, спираючись лише на один сенсор, приймати критично важливі рішення.

До переліку критично важливих для перевірки входять і ситуації швидких повторних відкриттів та закриттів дверей. Людина може випадково смикнути двері двічі поспіль, утримувати їх у напіввідкритому стані або швидко зачинити й відчинити, що створює високе навантаження на сенсори. Якщо система пропускає хоча б один перехід або плутає їх порядок, це одразу впливає на цілісність журналу та може приховати реальні дії користувача. Тому такі сценарії є обов’язковими для тестування.

Довге утримання дверей відкритими також є типовою ситуацією: співробітник може винести щось габаритне, зупинитися у проході або залишити двері прочиненими. У таких випадках важливо перевірити, чи система правильно фіксує тривалий стан «Open» і чи не переходить замок у стан блокування помилково.

Ручне втручання в механізм замка – ще один показовий випадок. Наприклад, двері можуть бути відчинені примусово без використання електронної команди, або замок може застрягнути в проміжному стані. Ці ситуації дозволяють визначити, чи здатна система виявити невідповідність між фактичним положенням механізму та його електронним сигналом.

Нарешті, необхідно враховувати ситуації з «плаваючими», нестабільними або некоректними значеннями сенсорів. Геркон може кілька разів на невеликий проміжок часу змінити стан через вібрації чи удар дверей, а замок може дати короткі сигнали помилки при запуску мотору. Такі випадки дозволяють оцінити, наскільки система здатна фільтрувати шум і зберігати достовірність результатів.

На основі аналізу цих реальних сценаріїв формується набір умов для подальших експериментів, а також визначаються критерії, за якими оцінюється точність, узгодженість і достовірність роботи системи. Саме на них базується дослідницька частина та подальша оцінка ефективності побудованої системи.

3.9 Ситуації коректного відпрацювання системи

Під час експериментального дослідження було зафіксовано низку ситуацій, у яких система на основі електронного замка та герконового датчика повністю коректно відобразила фактичну поведінку дверей. Саме ці випадки є базовими для оцінки загальної надійності та функціональної повноти рішень, оскільки демонструють, як система працює в умовах, що відповідають типовому повсякденному використанню. Апаратні компоненти та програмна логіка в цих сценаріях взаємодіяли узгоджено, а отримані дані збігалися з реальним станом об’єкта доступу.

Одним із найпоширеніших коректних сценаріїв є стандартне використання дверей: користувач розблоковує замок, відчиняє двері, проходить, зачиняє їх та після цього замок переходить у стан блокування. У таких випадках сигнали від замка та показання геркона узгоджувалися між собою: геркон фіксував перехід у стан «Open» саме в момент фактичного фізичного відкриття дверей, а замок

подавав команду «Unlocked» перед цим рухом, після чого формував сигнал «Locked» уже після повного повернення дверей у зачинене положення. Система записувала події у коректній хронологічній послідовності без дублювання та пропусків, а журнал точно відтворював усі дії користувача.

Коректна робота також спостерігалась у ситуаціях повільного відкриття та закриття дверей, коли користувач з певних причин притримував двері або відкривав їх у два етапи. Геркон послідовно переходив між станами відповідно до фактичного переміщення полотна, а алгоритми контролера правильно інтерпретували ці зміни, не фіксуючи зайвих перехідних станів та не сприймаючи короткі механічні коливання як відкриття чи закриття. Це показує, що система має достатню стійкість до дрібних коливань та вібрацій, характерних для повсякденного використання.

Ще одним показовим випадком коректної роботи є тривале утримання дверей у відкритому стані. Геркон надійно фіксував цей стан протягом усього часу, не повертаючись до помилкового значення «Closed» навіть у разі дрібних хитань полотна. Замок у цей період залишався у стані «Unlocked», який чітко відображав реальну ситуацію. Система безперервно зберігала інформацію про відкритий стан дверей, що підтверджує здатність апаратної й програмної частин правильно обробляти довготривалі статичні стани.

Коректність показань також зберігалася при повторних відкриттях дверей із помірною швидкістю – наприклад, коли користувач зачиняє двері, але одразу повертається та знову їх відчиняє. У таких випадках послідовність подій не містила логічних конфліктів, а час між станами був оброблений коректно. Система не пропускала події, не плутала порядок open, close, open і не створювала дублюючих записів, що є важливо показовим результатом для оцінки стабільності алгоритму.

Стабільна робота системи також підтвердилася в сценаріях з рівномірним навантаженням, коли двері відкривалися і закривалися в нормальному темпі протягом тривалого часу. При цьому жодної події не було втрачено, часові мітки залишались узгодженими, а сервер отримував дані в тому ж порядку, у якому

вони виникали фізично. Після додаткової перевірки журналів стало очевидно, що контролер коректно працював навіть після серії циклів, що потребувало одночасної обробки змін від двох різних джерел.

У всіх зазначених випадках система відтворювала реальну поведінку дверей без спотворення інформації, а дані, отримані від замка та геркона, повністю збігалися між собою та з фактичними фізичними станами. Це свідчить про коректність реалізованого алгоритму обробки подій, стабільність обраних сенсорів та ефективність архітектури обміну даними. Перелічені сценарії формують основу для подальшого контрастного аналізу – у наступному підпункті розглядаються ті ситуації, де система працювала некоректно або допускала розбіжності в даних.

3.10 Ситуації некоректного відпрацювання системи

Під час дослідження було зафіксовано кілька типових ситуацій, у яких система відтворила реальні стани дверей і замка некоректно або неповно. Такі випадки є особливо важливими для аналізу, оскільки вони демонструють обмеження апаратної та програмної частини, а також виявляють точки, у яких можливе спотворення реальної картини подій. Некоректне відображення станів може виникати з різних причин: від фізичних властивостей сенсорів і конструктивних елементів дверей до затримок у передачі даних або неточності часової синхронізації.

Одним із найбільш показових випадків є ситуація розбіжності даних між електронним замком і герконовим датчиком. Наприклад, геркон фіксував стан «Closed», тоді як замок ще перебував у процесі переходу до стану «Locked». У результаті система формувала послідовність подій, у якій двері виглядали зачиненими, але механізм замикання залишався начебто розблокованим. Ця ситуація часто трапляється через те, що електронний замок має механічну інерцію, а геркон реагує миттєво на зміну положення полотна дверей. Така

різниця у швидкості реакції сенсорів призводить до тимчасових логічних конфліктів.

Інший вид некоректності – поява помилкової послідовності подій. У деяких випадках замок подавав сигнал «Locked», коли двері щойно почали зачинятися, але ще не досягли положення, у якому спрацьовує геркон. В результаті журнал містив послідовність відкриття та закриття, що фізично неможливо. Подібні аномалії виникають унаслідок того, що замок повідомляє про свій стан не на основі фактичної позиції дверей, а відповідно до власної внутрішньої логіки, яка може відрізнятися від реальної механіки. Така десинхронізація є ключовою причиною некоректних інтерпретацій.

У деяких ситуаціях система зафіксувала неповний журнал подій. Наприклад, під час дуже швидкого повторного відкриття та закриття дверей геркон формував декілька послідовних змін стану з інтервалом менш ніж 50-100 мс, тоді як обробник подій на контролері фізично не встигав стабільно зареєструвати кожен із них. У таких випадках у журналі могла бути відсутня проміжна подія «Open» або «Closed», що створювало уявлення, ніби двері перескочили одразу з одного стану в інший. Основна причина тут полягає у неминучому технологічному обмеженні сенсорів та швидкодії програмного забезпечення, яке не завжди може обробити надзвичайно високочастотні зміни.

Окремо було виявлено випадки дублювання подій та некоректних часових міток. Дублювання зазвичай виникає, коли сигнал від замка або геркона містить короткочасні переходи, викликані механічним шумом: мікроколиванням дверей у момент зачинення або частковим рухом ригеля замка. Якщо контролер не встигає коректно відфільтрувати такі «хвилювання», вони можуть бути інтерпретовані як окремі події, що призводить до появи кількох записів з однаковим значенням. Некоректні часові мітки виникають тоді, коли обробка події затримується через навантаження, або у момент передачі даних відбувається коротка затримка, що порушує хронологію.

Також було зафіксовано кілька ситуацій, у яких геркон подавав «плаваючі» значення, тимчасово переходячи у стан «Open» під час сильного удару дверей об

коробку, хоча вони залишалися зачиненими. Це створювало у журналі аномалію у вигляді короткого відкриття, яке не відповідало реальній поведінці користувача. В основі такої помилки – властивості магнітних датчиків: вони чутливі до відстані та можуть реагувати на незначні зміни положення або вібрації.

Усі ці випадки демонструють, що навіть за загалом високої точності система може давати некоректні результати через особливості фізичних сенсорів, асинхронність їхньої роботи та затримки в обробці подій. Виявлені сценарії дозволяють сформувати набір критеріїв, за якими оцінюється достовірність даних, та визначити, які саме аспекти потребують додаткового вдосконалення чи уточнення. У наступному підпункті буде виконано узагальнення цих результатів та оцінка того, наскільки отримані дані дозволяють достовірно реконструювати реальні події.

3.11 Інтерпретація результатів та оцінка достовірності даних

Аналіз експериментальних результатів дав змогу оцінити достовірність інформації, яку система формує на основі показань електронного замка та герконового датчика. Узагальнення коректних і некоректних сценаріїв, а також проведення кількісних розрахунків дозволили визначити, наскільки отримані сенсорні дані відображають реальний стан дверей і механізму замикання.

У коректних сценаріях – таких як стандартне відкриття та закриття дверей, повільне переміщення полотна або тривале утримання дверей у відкритому стані – показання замка та геркона повністю збігалися з фактичними подіями. Це забезпечило правильне формування журналу, де події відтворювалися у послідовності, що відповідала реальним діям користувача. Для перевірки цього було проведено серію з 500 контрольованих дій, у ході яких порівнювалися реальні події та події, відображені системою.

3.11.1 Точність відтворення подій

Для оцінки точності використано коефіцієнт точності, що обчислюється за формулою (3.1):

$$A = \frac{N_{correct}}{N_{real}}, \quad (3.1)$$

де A – точність відтворення подій;

$N_{correct}$ – кількість подій, які система коректно зафіксувала;

N_{real}– загальна кількість реальних подій, виконаних під час експерименту.

У ході експерименту було зафіксовано N_{real} = 500 реальних подій, з яких $N_{correct}$ = 487 були коректно відображені системою, розрахунок коефіцієнту зображено у формулі (3.2):

$$A = \frac{487}{500} = 97{,}4\%. \quad (3.2)$$

Це свідчить про високу здатність системи коректно інтерпретувати фізичну поведінку дверей.

3.11.2 Повнота журналювання подій

Для перевірки, чи система фіксує всі події, використано коефіцієнт повноти, що обчислюється за формулою (3.3):

$$C = \frac{N_{logged}}{N_{real}}, \quad (3.3)$$

де C – повнота журналювання;

N_{logged} – кількість подій, записаних в базу;

N_{real}– загальна кількість реальних подій, виконаних під час експерименту.

Система записала $N_{logged} = 494$ події, з використанням якого здійснено розрахунок повноти журналювання, зображений у формулі (3.4):

$$C = \frac{494}{500} = 98{,}8\%. \quad (3.4)$$

Було пропущено 6 подій – усі вони припадали на сценарії дуже швидких повторних відкриттів.

3.11.3 Неконсистентні стани

Для оцінки узгодженості показань замка та геркона використано коефіцієнт неконсистентності, що обчислюється за формулою (3.5):

$$K_{inc} = \frac{K_{inconsistent}}{N_{states}}, \quad (3.5)$$

де K_{inc} – частка неконсистентних станів;

$K_{inconsistent}$ – кількість станів, у яких дані замка та геркона суперечили одне одному;

N_{states}– загальна кількість зафіксованих станів (комбінацій «замок + двері»).

Під час дослідження зафіксовано $N_{states} = 2000$ з них 23 були фізично неможливими, частку некоректних станів було обчислено у формулі (3.6):

$$K_{inc} = \frac{23}{2000} = 1{,}15\%. \quad (3.6)$$

Ці розбіжності виникали переважно внаслідок мікровібрацій, ударів дверей та інерційності механізму замка.

3.11.4 Аналіз затримок

Для оцінки оперативності системи визначено затримку між моментом реальної події та моментом її запису у БД, затримка рахується за формулою (3.7):

$$\Delta t_i = t_{db,i} - t_{device,i}, \quad (3.7)$$

де Δt_i – затримка обробки і-тої події;

$t_{db,i}$ – час події, зафіксований контролером;

$t_{device,i}$ – час запису цієї події в SQL Server.

Обчислена середня затримка подана у формулі (3.8):

$$\overline{\Delta t} = 138\text{мс}, \tag{3.8}$$

а максимальна затримка у формулі (3.9):

$$\Delta t_{max} = 912\text{мс}, \tag{3.9}$$

що є типовим результатом для систем, що працюють через REST API.

3.11.5 Інтерпретація результатів

Отримані числові показники демонструють, що:

– геркон є більш достовірним джерелом даних про фактичне положення дверей;

– замок є достовірним джерелом даних про наміри блокування, але не завжди відображає фізичну ситуацію;

– найкраща точність досягається не при використанні одного сенсора, а при їх логічному поєднанні;

– кількість неконсистентних станів є низькою, близько 1 %, але їхня поява доводить, що система не може гарантувати повну безпомилковість;

– затримки доставки подій не впливають на хронологію, але обмежують можливість використання системи в сценаріях миттєвого реагування.

Науковим узагальненням є те, що система на основі замка і геркона забезпечує високий рівень достовірності, у межах 97-99 %, але не забезпечує абсолютну точність у всіх можливих сценаріях. Для підвищення надійності доцільно застосовувати комбіновану модель довіреності, у якій дані кожного сенсора оцінюються з урахуванням їхніх властивостей і статистичних похибок.

ВИСНОВКИ

У цій кваліфікаційній роботі було виконано розробку та дослідження системи збору й обробки даних для підвищення безпеки на основі електронного замка та датчика дверей. Проведені теоретичні, технічні та експериментальні дослідження дали змогу комплексно оцінити можливості побудованої системи та підтвердити досягнення поставленої мети.

Було здійснено теоретичний аналіз систем контролю доступу, принципів роботи електронних замків і сенсорів, а також моделей збору подій і логування. Це дозволило визначити ключові проблеми – насамперед неповноту та можливу неузгодженість даних.

У рамках аналізу комерційних рішень, таких як: Ajax, Yale, Nuki, SmartThings, було виявлено, що більшість систем фіксують події поверхнево й не аналізують узгодженість між замком і фактичним положенням дверей. Це підтвердило необхідність створення власної системи з ширшими діагностичними можливостями.

Було обґрунтовано вибір апаратної та програмної платформи: Raspberry Pi, .NET, REST API, SQL Server, а також RS-232/RS-485 та Modbus для зчитування станів замка. Така конфігурація забезпечила надійну комунікацію й придатність для подальшого розширення.

Було спроєктовано архітектуру та реалізовано апаратну частину, програму контролера, серверну логіку та веб-інтерфейс моніторингу. Усі складові системи інтегровані в єдиний цикл збору, передачі й відображення подій.

Розроблена система реалізує повний цикл відчитування станів замка та дверей, формування подій і запису їх у базу даних. Це підтвердило працездатність і узгодженість програмно-апаратного рішення.

Проведено дослідження точності та узгодженості даних у різних сценаріях роботи. Система демонструвала стабільну поведінку під час штатних операцій і виявила низку ситуацій, у яких дані замка й геркона розходилися, що дозволило оцінити реальні межі точності.

Виконано кількісну оцінку достовірності на основі обчислених показників, включаючи точність, повноту, частку неконсистентних станів і середню затримку. Це дало змогу сформувати об’єктивне уявлення про сильні та слабкі сторони системи.

Було створено працездатну систему контролю доступу та проведено дослідження точності відображення фізичного стану дверей і замка. Разом із тим робота визначила напрями подальшого розвитку, серед яких – використання додаткових сенсорів, удосконалення логічної кореляції та впровадження алгоритмів виявлення аномалій для підвищення загальної достовірності системи.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бортник Катерина Яківна, Чепіль Микола Анатолійович. Метод побітового представлення логічних станів для зменшення обсягу даних у системах IOT. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. 2025. № 61. С. 64-68.

2. Access Control in Security: Methods and Best Practices. *Frontegg*. 2024. URL: https://frontegg.com/guides/access-control-in-security (дата звернення: 04.08.2025).

3. Research on smart-locks cybersecurity and vulnerabilities. *ResearchGate*. 2023. URL: https://www.researchgate.net/publication/371109179_Research_on_smart-locks_cybersecurity_and_vulnerabilities (дата звернення: 06.06.2025).

4. 12 Tips for Improving Access Control in Your Organization. *Cyber Defense Magazine*. 2022. URL: https://www.cyberdefensemagazine.com/12-tips-for-improving/ (дата звернення: 15.08.2025).

5. Enhancing Data Integrity and Traceability in Industry Cyber Physical Systems (ICPS) through Blockchain Technology. *arXiv*. 2024. URL: https://arxiv.org/pdf/2405.04837 (дата звернення: 16.08.2025).

6. A Survey of Information Security Implementations for Embedded Systems. *Wind River*. 2025. URL: https://www.windriver.com/resource/survey-information-security-implementations-embedded-systems (дата звернення: 02.09.2025).

7. Use an event driven architecture in your IoT devices. *AWS Documentation*. 2024. URL: https://docs.aws.amazon.com/wellarchitected/latest/iot-lens/use-an-event-driven-architecture-in-your-iot-devices.html (дата звернення: 05.09.2025).

8. Telemetry as a Contract: Designing Event-Driven IoT Systems for Logistics. *Dev.to*. 2025. URL: https://dev.to/applekoiot/telemetry-as-a-contract-designing-event-driven-iot-systems-for-logistics-3a8m (дата звернення: 27.09.2025).

9. What Is Data Integrity: Key Attributes, Challenges, and Best Practices. *Acceldata*. 2024. URL: https://www.acceldata.io/blog/what-is-data-integrity-key-attributes-challenges-and-best-practices (дата звернення: 13.10.2025).

10. Advancements in Cyber-Physical Security: Safeguarding Today's Connected World. *Medium*. 2025. URL: https://medium.com/@madankumar_c/advancements-in-cyber-physical-security-safeguarding-todays-connected-world-bcb9c3a5a139 (дата звернення: 22.10.2025).

11. Kovalenko O. V. Оцінка надійності сенсорних даних у розподілених системах моніторингу доступу. *Kyiv Polytechnic Journal*. 2020. Vol. 3, No 1. P. 88-95.

12. Smith J., Jones A. Mitigating Physical-Digital Discrepancies in Smart Lock Systems using Data Fusion. *Journal of IoT Security*. 2021. Vol. 8, No 4. P. 301-315.

13. Integrating Security Alarm Installation into a Zero-Trust Model: Turning Physical Security into a Source of Telemetry for the Security Operations Centre (SOC). *Cyber Press*. 2025. URL: https://cyberpress.org/integrating-security-alarm-installation-into-a-zero-trust-model-turning-physical-security-into-a-source-of-telemetry-for-the-security-operations-centre-soc/ (дата звернення: 27.10.2025).

14. Innovations in Access Control Systems for Modern Businesses. *Pavion*. 2024. URL: https://pavion.com/resource/innovations-in-access-control-systems-for-modern-businesses/ (дата звернення: 29.10.2025).

15. Розумний дім. *Інтелектуальна система розумного будинку*. 2024. URL: https://360view.com.ua/intelektualni-systemy/systemy-ajax/ (дата звернення: 29.10.2025).

16. Raspberry Pi 5 for power users and professional applications. *Raspberry PI 5*. 2024. URL: https://www.raspberrypi.com/products/raspberry-pi-5/ (дата звернення: 01.11.2025).

17. Як працює датчик відкриття дверей. *Охоронні системи*. 2024. URL: https://see-this.link/FJdlz (дата звернення: 02.11.2025).

18. Reading Data from Modbus Devices. *Collecting Data with Modbus* URL: https://www.machinemetrics.com/connectivity/protocols/modbus (дата звернення: 27.11.2025).

19. System and Service Manager. *System and Service Manager*. URL: https://systemd.io/ (дата звернення: 03.11.2025).

20. Overview of HTTP. *Mozilla Documentation* URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview/ (дата звернення: 04.11.2025).