

Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____Олексій СМІРНОВ
“_____” _____2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему:
“ Програмне забезпечення системи кібербезпеки в Zero-Trust
архітектурі безпечного розумного будинку”

Виконав здобувач вищої освіти
IV курсу, групи КБ-22-1
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____Паращенко Д.С.
«_____» _____2026р.

Керівник проекту
кандидат технічних наук, доцент
_____Дресєв О.М.
«_____» _____2026р.
Рецензент _____

Центральноукраїнський національний технічний університет
Факультет Механіко-технологічний
Кафедра Кібербезпеки та програмного забезпечення
Рівень вищої освіти бакалавр
Галузь знань 12 "Інформаційні технології"
Спеціальність 125 "Кібербезпека"
Освітньо-професійна (освітньо-наукова) програма "Кібербезпека"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
27 лютого 2026 року

**ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА
ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ**

Паращенко Денис Сергійович

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки в Zero-Trust архітектурі безпечного розумного будинку*

2. Керівник роботи *Дресєв Олександр Миколайович, кан. техн. наук, доцент.*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 134 -02 від 27.02.2026 року

3. Строк подання студентом роботи до захисту 10.06.2026

4. *Метою розробки є проектування та програмно-апаратна реалізація системи кібербезпеки для моніторингу, автентифікації та захисту даних у мережі «розумного будинку» на основі концепції Zero Trust.*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2. Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи 1 аркуш

Функціональна схема системи 1 аркуш

Діаграма процесів 1 аркуш

Блок-схеми алгоритму роботи додатку 2 аркуша

7. Дата видачі завдання 27.02.26

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем керування	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв'язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	14.05.2026 р.	
8.	Попередній захист роботи	16.05.2026 р.	

Дата видачі завдання 27.02.26

Підпис керівника

(прізвище та ініціали)

Завдання прийнято до виконання

27.02.26

Підпис здобувача

(прізвище та ініціали)

АНОТАЦІЯ

Паращенко Д.С. Програмне забезпечення системи кібербезпеки в Zero-Trust архітектурі безпечного розумного будинку. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній кваліфікаційній бакалаврській роботі розроблено комплексну систему безпеки «розумного будинку», яка базується на принципах архітектури нульової довіри (Zero Trust) та забезпечує захищену взаємодію між компонентами інфраструктури.

Метою розробки є проектування та програмно-апаратна реалізація системи кібербезпеки для моніторингу, автентифікації та захисту даних у мережі «розумного будинку» на основі концепції Zero Trust.

Результат роботи – програмна та апаратна реалізація системи захисту, що включає серверне програмне забезпечення для автентифікації пристроїв, контроль доступу на основі токенів, а також інтегровані мікроконтролери для безпечного збору та передачі телеметрії.

В процесі роботи над проектом виконано аналіз існуючих архітектурних рішень, загрозливих моделей та засобів захисту IoT-пристроїв.

Програмна частина сервера може використовуватися на ЕОМ під керуванням сучасних операційних систем (Windows / Linux), а апаратна частина реалізована на базі мікроконтролерів архітектури ESP32. Програмний комплекс розроблено з використанням сучасних середовищ розробки та мов програмування (Python, C++).

Програму розроблено в середовищі Visual Studio.

Ключові слова: кібербезпека, розумний будинок, Zero Trust, IoT.

ABSTRACT

Parashchenko D. S. Software of the Cybersecurity System in a Zero-Trust Architecture of a Secure Smart Home. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

This bachelor's thesis presents the development of a comprehensive security system for a "smart home," which is based on the principles of Zero Trust architecture and ensures secure interaction between infrastructure components.

The objective of the development is the design and hardware-software implementation of a cybersecurity system for monitoring, authentication, and data protection within a "smart home" network based on the Zero Trust concept.

The result of the work is a software and hardware implementation of a security system that includes server software for device authentication, token-based access control, as well as integrated microcontrollers for secure telemetry collection and transmission.

In the course of the project, an analysis of existing architectural solutions, threat models, and security tools for IoT devices was conducted.

The server software component can be utilized on computers running modern operating systems (Windows / Linux), while the hardware component is implemented based on ESP32 architecture microcontrollers. The software suite was developed using modern integrated development environments (IDEs) and programming languages (Python, C++).

The software was developed in the Visual Studio environment.

Keywords: cybersecurity, smart home, Zero Trust, IoT.

ЗМІСТ

ВСТУП.....	Ошибка! Закладка не определена.
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ.....	Ошибка! Закладка не определена.
1.1 Призначення системи.....	Ошибка! Закладка не определена.
1.2 Область застосування.....	Ошибка! Закладка не определена.
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ	7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим	7
2.2 Обґрунтування засобів для побудови системи та мови програмування.....	Ошибка! Закладка не определена.12
2.3 Розгорнута постановка завдання	14
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ	16
3.1 Опис функціонування системи	16
3.2 Розробка структурної схеми.....	19
3.3 Розробка функціональної схеми	21
3.4 Розробка діаграми процесів.....	23
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ.	26
4.1 Розробка блок–схем та опис алгоритмів функціонування системи.....	26
4.2 Захист розробленого програмного забезпечення.....	34
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ	39
6 ОСНОВНІ ВИСНОВКИ.....	42
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	44

					ВКРБ–125.26.00.14.00.00.ПЗ				
Зм.	Арк.	№ докум.			Програмне забезпечення системи кібербезпеки в Zero-Trust архітектурі безпечного розумного будинку	Літ.	Аркуш	Аркушів	
Розроб.		Паращенко Д.С					2	48	
Перевірів		Дресв О.М..							
Н. контр.		Коваленко А.С.				КБ–22–1			
Затв.		Смірнов О.А.							

ВСТУП

Актуальність теми. Розвиток технологій Інтернет речей(IoT) призвів до впровадження систем «розумний будинок», які об'єднують пристрої у єдину мережу. Зростання кількості підключених пристроїв та їх інтеграція підвищують вимоги до їхньої кібербезпеки.

Більшість IoT-пристроїв мають обмежені обчислювальні ресурси та слабкість вбудованих механізмів захисту, що створює фактори вразливості цифрової інфраструктури, ці загрози доцільно класифікувати за критеріями деструктивного впливу на систему:

- **Порушення конфіденційності:** несанкціонований доступ до вузлів мережі та перехоплення даних (eavesdropping), що призводить до витоку чутливої інформації користувача.

- **Порушення цілісності та автентичності:** підміна пристроїв (spoofing) та маніпуляція потоками даних, що дозволяє зловмиснику інтегрувати шкідливі вузли в екосистему.

- **Компрометація доступності та ресурсів:** експлуатація вразливостей для включення пристроїв до складу ботнетів, що виснажує обчислювальні ресурси і може бути використано для проведення DDoS-атак.

Традиційні підходи до забезпечення безпеки, засновані на довірі до внутрішньої мережі, виявляються недостатньо ефективними в умовах динамічної структури «розумного будинку» і постійного підключення нових пристроїв. Це обумовлює необхідність застосування нових підходів до забезпечення кібербезпеки.

У цьому контексті пріоритетним є впровадження інноваційних методів кіберзахисту, зокрема парадигми Zero Trust, що базується на принципі відсутності презумпції довіри до будь-яких вузлів мережі і обов'язковій верифікації кожного запиту. Реалізація цієї архітектури забезпечує підвищення

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		2

безпеки систем «розумного будинку» шляхом безперервної автентифікації, суворого контролю доступу та безперервного моніторингу мережевого трафіку.

Мета й завдання дослідження. Метою даної роботи є підвищення рівня кібербезпеки системи «розумного будинку» шляхом дослідження та застосування архітектури Zero Trust, а також розробки підходів до захисту IoT-пристроїв від загрози несанкціонованого доступу, витоку даних і мережевих атак.

Завдання дослідження:

- проаналізувати особливості функціонування систем «розумного будинку»;
- дослідити структуру і основні компоненти IoT-мереж;
- визначити основні загрози і вразливості IoT-пристроїв;
- проаналізувати існуючі підходи до забезпечення безпеки в IoT-системах;
- виявити недоліки традиційних моделей захисту;
- дослідити принципи та основні положення архітектури Zero Trust;
- проаналізувати можливості застосування Zero Trust у системах «розумного будинку».

Об'єктом дослідження є система «розумного будинку» як сукупність взаємопов'язаних IoT-пристроїв та мережевої інфраструктури. Предметом дослідження є методи та засоби забезпечення кібербезпеки IoT-мереж на основі архітектури Zero Trust.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для забезпечення безпечного керування та моніторингу пристроїв розумного будинку в умовах мережі Інтернет речей. Забезпечує збір телеметрії з датчиків, віддалено керує виконавчими пристроями, захищено обмінюється інформацією між компонентами системи і контролює доступ користувачів.

Концепція «розумного будинку» спрямована на створення житлового середовища, у якому більшість повсякденних процесів автоматизовані, а рівень безпеки мешканців залишається притримується високих стандартів.

«Розумний будинок» – технологія Інтернет речей (IoT). У межах цієї концепції побутові пристрої, сенсори та системи керування об'єднуються в єдину мережу. Освітлення, термостати, камери та інші пристрої постійно обмінюються даними між собою та з користувачем через локальну мережу або хмарні сервіси.

Призначення системи – спростити повсякденне життя, підвищити рівень комфорту і безпеки, передавши більшість рутинних завдань автоматизованим алгоритмам. Керування можливе як із середини будинку, так і через інтернет із будь-якої точки світу.

Архітектурно систему можна умовно поділити на три взаємопов'язані рівні:

– Рівень пристроїв (device layer). Сенсори та виконавчі механізми, які фіксують зміни в середовищі та виконують фізичні дії.

– Мережевий рівень (network layer). Канали зв'язку, що забезпечують обмін інформацією між пристроями і серверними компонентами за допомогою Wi-Fi, Zigbee, Bluetooth тощо.

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		4

– Прикладний рівень (application layer). Рівень обробки даних, де працюють алгоритми автоматизації та інтерфейси мобільних застосунків і вебсервісів.

У результаті формується складна динамічна система, яку можна постійно розширювати новими пристроями. Саме тому питання захисту інформації та керування доступом стає не менш важливим, ніж забезпечення комфорту.

Система вирішує одразу кілька задач:

– Автоматизація повсякденних процесів. Освітлення, мікроклімат і робота побутової техніки регулюються автоматично, що зменшує потребу в постійному втручанні людини.

– Віддалений контроль. Користувач може перевіряти стан житла та керувати ним через мобільний застосунок або вебінтерфейс незалежно від свого місцезнаходження.

– Підвищення безпеки. Датчики миттєво реагують на потенційні загрози: проникнення сторонніх осіб, появу диму, витік газу чи води.

– Енергоефективність. Система автоматично вимикає непотрібні пристрої за відсутності людей або працює за заданим графіком, зменшуючи витрати ресурсів.

1.2 Область застосування

Системи «розумного будинку» мають широкий спектр застосування і можуть впроваджуватися як у приватному, так і в комерційному секторі. Їх універсальність дозволяє адаптувати функціональність під потреби користувачів, масштаб об'єкта і вимоги до безпеки й енергоефективності.

Завдяки можливості гнучкого налаштування та масштабування подібні рішення застосовуються не лише в дорогому житлі:

Приватні будинки та квартири. Створення персоналізованих сценаріїв комфорту – від автоматичних штор до комплексних систем відеоспостереження.

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		5

Готелі та орендне житло. Використання смарт-замків, контроль витрат ресурсів і автоматизація підготовки приміщень до заселення.

Офісні та комерційні об'єкти. Інтелектуальні системи доступу та керування простором допомагають зменшити витрати на утримання будівель.

В даних випадках система забезпечує автоматизацію побутових процесів: керування освітленням, опаленням, кондиціонуванням повітря, побутовою технікою і мультимедійними пристроями. Значна увага приділяється безпеці, зокрема контролю доступу, відеоспостереженню і моніторингу аварійних ситуацій.

К6ПЗ-2026

					ВКРБ-125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		6

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим

Щоб визначити доцільність розробки безпечної системи «розумного будинку», необхідно провести аналіз існуючих програмних рішень, технологій і підходів до організації IoT-інфраструктури.

Системи «розумного будинку» відрізняються архітектурою, рівнем централізації, способами керування і механізмами забезпечення безпеки.

Потрібно приділити увагу підходам до захисту IoT-пристроїв, збільшення кількості підключених вузлів підвищує ризики несанкціонованого доступу і компрометації мережі.

Системи «розумного будинку» – комплекс апаратних і програмних засобів, забезпечуючи автоматизацію керування побутовими пристроями, моніторинг середовища і віддалений доступ, інтегруючи IoT-пристрої в єдину мережу.

Home Assistant – система автоматизації «розумного будинку» з відкритим вихідним кодом, що реалізує локальне керування IoT-пристроями без залежності від хмарних сервісів. Інтеграція великої кількості пристроїв підтримується через Wi-Fi, Zigbee, Bluetooth та інші протоколи.

Переваги системи:

- локальне розгортання;
- висока гнучкість конфігурації;
- підтримка великої кількості інтеграцій.

Недоліки:

- складність налаштування для користувача;
- необхідність самостійного адміністрування;

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		7

– безпека значною мірою залежить від конфігурації.

Google Home – централізована платформа для керування пристроями «розумного будинку», що працює переважно через хмарну інфраструктуру.

Переваги:

- простота використання;
- підтримка великої кількості пристроїв;
- інтеграція з голосовими сервісами.

Недоліки:

- залежність від інтернет-з'єднання;
- передача даних через хмарні сервіси;
- обмежений контроль над механізмами безпеки.

Amazon Alexa – централізоване керування IoT-пристроями та автоматизація сценаріїв, підтримує інтеграцію з великою кількістю пристроїв.

Переваги:

- високий рівень автоматизації;
- підтримка голосового керування;
- широка екосистема пристроїв.

Недоліки:

- використання хмарної інфраструктури;
- ризики конфіденційності;
- централізована модель довіри.

OpenHAB – платформа з відкритим вихідним кодом, призначена для локального керування пристроями, підтримує значну кількість протоколів і дозволяє реалізувати власні сценарії автоматизації.

Переваги:

- відкритий код;
- можливість локального розгортання;
- гнучкість налаштувань.

Недоліки:

- складність конфігурації;
- потреба у технічних знаннях;
- залежність рівня безпеки від правильності налаштування.

В системах «розумного будинку» щоб забезпечити безпеку використовують різні підходи, найпоширенішими є парольний захист, мережеві екрани (firewall) і віртуальні приватні мережі (VPN). Ефективність цих методів у IoT-системах обмежений через архітектуру та динамічність середовища.

Базовий механізм захисту – парольна автентифікація, вона передбачає використання облікових даних для доступу до пристроїв, сервісів. Не дивлячись на простота реалізації, даний підхід має недоліки:

- стандартні, слабкі паролі;
- відсутність багатфакторної автентифікації;
- складність централізованого управління доступом при збільшенні кількості пристроїв.

Мережевий екран (firewall) використовують для контролю трафіку на основі заданих шаблонів. Дозволяє обмежити доступ до мережі ззовні та фільтрувати небажані з'єднання. У IoT-систем ефективність знижується через кількість пристроїв, різноманітність протоколів і відсутність глибокого аналізу поведінки пристроїв.

Поширеним підходом є використання віртуальних приватних мереж (VPN), це забезпечує захищений канал зв'язку між користувачем і системою, шифрує передані дані і зменшує ризик їх перехоплення. На жаль, цей підхід не вирішує проблему внутрішніх загроз, після встановлення з'єднання користувач або пристрій отримує надмірний рівень доступу, який часто йому не потрібен до ресурсів мережі.

Застосовуються базові заходи безпеки, такі як сегментація мережі, використання антивірусного програмного забезпечення та оновлення

прошивок пристроїв. Ці методи не можуть повністю забезпечити комплексного захисту, вони не враховують поведінкові аспекти та не передбачають постійної перевірки довіри до пристроїв.

Стандартні моделі забезпечення безпеки в інформаційних системах базуються на припущенні довіри до внутрішньої мережі. З цим підходом, після проходження автентифікації користувач або пристрій вважається довіреним і отримує доступ до ресурсів системи без постійної перевірки. Дана модель є ефективною для статичних мереж. В умовах IoT-систем вона має обмеження.

Основним недоліком є надмірна довіра до внутрішньої мережі. При компрометації одного з пристроїв зломисник отримує доступ до інших елементів системи без додаткових перевірок, це підвищує ризик поширення атаки.

Проблемою є відсутність постійного контролю та моніторингу дій користувачів і пристроїв. У традиційних системах перевірка виконується лише на етапі входу, після чого активність у мережі не аналізується належним чином. Це ускладнює виявлення підозрілої поведінки та своєчасне реагування на інциденти.

Традиційні підходи не передбачають перевірки кожного запиту до ресурсів системи. Доступ надається один раз і зберігається протягом усього сеансу, це створює ризик використання скомпрометованих облікових даних або пристроїв.

Аналіз існуючих систем показав, що сучасні IoT-платформи забезпечують високий рівень автоматизації та підтримують велику кількість інтеграцій, проте мають декілька недоліків, пов'язаних з безпекою. Основна проблема централізованих систем, таких як Google Home та Amazon Alexa, є залежність від хмарної інфраструктури. Значна кількість операцій виконується через віддалені сервера, дані передаються через зовнішні сервіси. Це створює ризики порушення конфіденційності, витоку даних, втрати доступу до функціоналу системи при відсутності інтернет з'єднання.

В створеній системі дана проблема вирішується через розгортання локальної серверної частини. Процеси обробки телеметрії, автентифікації і керування пристроями виконується локально без залежності від хмарних сервісів. Це дозволяє знизити ризики компрометації даних і забезпечити працездатність системи при відсутності доступу до мережі.

Іншою проблема існуючих платформ є надмірна довіра до пристроїв після проходження автентифікації. В більшості традиційних систем користувач або пристрій після успішного входу автоматично отримує широкий рівень доступу до мережі. При компрометації це створює умови для поширення атаки на інші компоненти.

Щоб усунути цю проблема в проєктованій системі застосовано концепцію Zero Trust. Кожен запит до серверної частини проходить повторну перевірку автентичності та прав доступу. Таким чином, навіть після успішного входу довіра не дається автоматично, а доступ контролюється постійно.

Важлива проблема багатьох систем – використання лише паролів автентифікації. Механізм логіна та пароля часто є недостатнім. При збільшенні кількості пристроїв централізоване керування значно ускладнюється.

У розроблюваній системі механізм автентифікації реалізований через JWT-токени. Після автентифікації користувач отримує цифровий токен, що використовується для підтвердження прав доступу до API. Це дозволяє контролювати доступ до функціоналу системи, забезпечити захищену взаємодію між клієнтом і сервером.

Передача даних у відкритому вигляді може створити ризики перехоплення телеметричних даних або команд керування.

Для зниження ризиків використовується HTTPS, що забезпечує шифрування трафіку між клієнтською частиною, сервером та пристроями. Це мінімізує ризик перехоплення і зміни інформації під час передачі.

Складність розгортання і адміністрування серверної інфраструктури ускладнює підтримку системи.

Щоб вирішити цю проблему у проекті використовується контейнеризація. Серверні компоненти запускаються у контейнерах, що забезпечує ізоляцію сервісів, спрощує розгортання і дозволяє швидко переносити систему між різними середовищами без необхідності повторної конфігурації.

Недостатній рівень моніторингу активності системи і відсутності засобів аналізу поведінки пристроїв у реальному часі. У випадку підозрілої активності адміністратор може не зреагувати вчасно.

В спроектованій системі реалізований механізм моніторингу телеметричних даних і стану компонентів. Це дозволяє відслідковувати активність пристроїв, навантаження системи, параметри функціонування і оперативно виявляти аномальну поведінку.

2.2 Обґрунтування засобів для побудови системи та мови програмування

Щоб реалізувати безпечну систему «розумного будинку» було обрано набір програмних і апаратних засобів, що забезпечують надійність, зручність та відповідність вимогам безпеки. При виборі технологій враховувались особливості роботи з IoT-пристроїв.

Основна платформа реалізації IoT-вузлів було обрано мікроконтролер ESP32. Цей контролер має вбудовані модулі Wi-Fi та Bluetooth, достатню продуктивність для роботи з сенсорами й виконавчими механізмами, низьке енергоспоживання. ESP32 широко використовується в IoT-системах завдяки доступності, підтримці великої кількості бібліотек і можливості інтеграції з сучасними протоколами передачі даних. Мікроконтролер підтримує безпечну передачу інформації через HTTPS, що є важливим для побудови захищеної інфраструктури «розумного будинку».

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		12

Щоб реалізувати серверну частину було обрано мову програмування Python і фреймворк FastAPI. Python – одна із найбільш популярних мов програмування для серверних застосунків завдяки простоті синтаксису, кількості бібліотек і швидкості розробки. Використання Python дозволяє спростити інтеграції з базами даних, реалізувати механізм автентифікації і швидко масштабувати функціональність системи.

Вибір FastAPI обумовлений високою продуктивністю, підтримкою асинхронної обробки запитів і автоматичним створенням документації API. Цей фреймворк дозволяє реалізувати REST API для взаємодії між пристроями, сервером і клієнтськими застосунками, підтримує механізми автентифікації і інтеграції з бібліотеками для реалізації JWT-токенів.

Для зберігання інформації про пристроїв, телеметрію та параметри системи використовується система керування базами даних PostgreSQL. СКБД забезпечує надійне зберігання інформації, підтримує складні запити, має високу стабільність і масштабованість. Використання реляційної бази даних дозволяє структуровано зберігати інформацію.

Для спрощеного розгортання і ізоляції компонентів в системі використовується Docker. Контейнеризація дає змогу запускати серверні компоненти в ізольованому середовищі, уникати конфліктів залежностей і забезпечувати однакову роботу системи на різних пристроях і операційних системах. Docker спрощує масштабування системи і процес супроводу.

Для безпечного обміну даними використовується протокол HTTPS. Передача інформації між сервером, клієнтом та пристроями здійснюється у зашифрованому вигляді.

Для автентифікації і авторизації було обрано JWT. Використання токенів дозволяє підтверджувати автентичність користувача і пристрою при кожному запиті до серверної частини.

Щоб забезпечити моніторинг роботи системи та аналіз телеметричних даних використовується Grafana з OpenTelemetry. Це рішення забезпечує

візуалізацію параметрів роботи системи, контроль пристроїв і можливість швидкого виявлення аномалій.

2.3 Розгорнута постановка завдання

Метою роботи є проєктування і реалізація безпечної системи «розумного будинку» на основі технологій Інтернету речей із використанням принципів архітектури Zero Trust, щоб підвищити рівень захисту пристроїв, користувачів та переданих даних. Система повинна забезпечувати автоматизацію процесів керування побутовими пристроями, моніторинг стану середовища та безпечний віддалений доступ до компонентів інфраструктури.

В межах завдання необхідно реалізувати серверну платформу, що забезпечує централізовану взаємодію між IoT-пристроями, користувачем та системою моніторингу. Основною функцією серверної частини є приймання телеметричних даних від пристроїв, обробка запитів на керування, автентифікація користувачів та забезпечення контролю доступу відповідно до політик безпеки.

Ключовим завданням є реалізація захищеної взаємодії між компонентами системи. Необхідно впровадити механізми автентифікації і авторизації на основі JWT-токенів, що дозволить підтверджувати легітимність кожного запиту до API. Передача інформації між пристроями і сервером повинно здійснюватися із використанням протоколу HTTPS, що забезпечує шифрування трафіку.

В межах проєкту необхідно реалізувати IoT-вузол на базі мікроконтролера ESP32, що виконує функцію збору телеметричних даних із сенсорів та передачі на сервер. Пристрій повинен підтримувати двосторонню взаємодію із серверною частиною, що дозволить не лише передавати інформацію про стан середовища, а й отримувати команди керування для виконавчих механізмів.

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		14

Щоб забезпечити централізоване зберігання інформації необхідно реалізувати базу даних, що зберігатиме відомості про користувачів, пристроїв, параметри роботи системи, журнал активності і телеметричні дані.

Реалізація моніторингу стану системи в реальному часі. Для цього інтегруємо засоби збору і візуалізації метрик, що відстежуватиме параметри роботи серверної частини, активність пристроїв і можливі аномалії.

Для підвищення гнучкості і спрощенню процесу розгортання необхідно забезпечити контейнеризацію серверної інфраструктури із використанням Docker. Це ізолює окремі компоненти системи, спростить адміністрування і забезпечить стабільність роботи.

При виконанні роботи необхідно вирішити основні завдання:

- провести аналіз існуючих систем «розумного будинку» та підходів до забезпечення безпеки IoT–середовища;
- дослідити основні загрози та вразливості, характерні для IoT–пристроїв;
- обґрунтувати вибір технологій і програмних засобів для реалізації системи;
- розробити архітектуру безпечної системи «розумного будинку» на основі концепції Zero Trust;
- реалізувати серверну частину системи із використанням FastAPI та механізмів JWT–автентифікації;
- реалізувати IoT–вузол на базі ESP32 для збору та передавання телеметричних даних;
- забезпечити захищений обмін інформацією із використанням HTTPS;
- реалізувати механізм моніторингу системи із застосуванням Grafana та OpenTelemetry;
- виконати тестування працездатності та оцінити рівень безпеки розробленої системи.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Система «розумного будинку» на базі IoT-пристрою ESP32, серверної частини FastAPI та підсистем моніторингу Prometheus і Grafana функціонує як розподілена клієнт-серверна архітектура, що забезпечує повний цикл роботи з телеметричними даними: їх збір, передачу, обробку, зберігання та візуалізацію в режимі реального часу. Така архітектура дозволяє забезпечити масштабованість системи, гнучкість у розширенні функціоналу та високу відмовостійкість при роботі з великою кількістю IoT-вузлів.

Загальна логіка функціонування системи побудована на принципі постійної взаємодії між клієнтською частиною (ESP32), сервером (FastAPI) та інструментами моніторингу. При цьому кожен компонент виконує чітко визначену роль: IoT-пристрій відповідає за збір фізичних даних, сервер – за їх обробку та керування, а системи моніторингу – за аналіз та візуалізацію стану всієї інфраструктури.

На першому етапі роботи IoT-пристрій ESP32 ініціалізує підключення до бездротової мережі Wi-Fi. Цей процес є критично важливим, оскільки саме через мережеве з'єднання забезпечується вся подальша комунікація із серверною частиною системи. Після запуску пристрій виконує процедуру ініціалізації мережевих параметрів, підключається до заданої точки доступу та перевіряє стабільність з'єднання. У випадку втрати зв'язку передбачено повторні спроби підключення, що підвищує надійність роботи системи в умовах нестабільної мережі.

Після встановлення стабільного з'єднання пристрій переходить до етапу автентифікації на сервері. Для цього виконується HTTP-запит до endpoint /auth/device-login, у якому передається унікальний ідентифікатор пристрою та

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		16

секретний ключ, що зберігається на стороні пристрою. Серверна частина, реалізована на базі FastAPI, перевіряє отримані дані шляхом звернення до бази даних, де зберігається інформація про зареєстровані пристрої.

У разі успішної перевірки сервер формує JWT–токен (JSON Web Token), який є основним механізмом авторизації в системі. Даний токен містить закодовану інформацію про пристрій та термін його дії, що дозволяє уникнути необхідності повторної автентифікації при кожному запиті. Отриманий токен передається назад на ESP32 та зберігається в пам'яті пристрою для подальшого використання при взаємодії із сервером.

Після завершення процедури автентифікації IoT–пристрій переходить у робочий режим. У цьому режимі він виконує періодичний збір телеметричних даних з підключених сенсорів. Основними параметрами, що вимірюються, є температура навколишнього середовища та вологість повітря. Вибір саме цих параметрів обумовлений їх широким використанням у системах моніторингу «розумного будинку», оскільки вони дозволяють оцінювати базові умови навколишнього середовища.

Зібрані дані формуються у структурований JSON–об'єкт, який містить значення вимірних параметрів, ідентифікатор пристрою та часову мітку вимірювання. Далі цей пакет даних передається на сервер через endpoint /telemetry за допомогою HTTP POST–запиту. Важливою складовою запиту є передача JWT–токена в заголовок Authorization, що забезпечує захищений доступ до серверних ресурсів.

На стороні сервера FastAPI здійснюється комплексна обробка отриманих даних. Спочатку виконується перевірка валідності JWT–токена, після чого ідентифікується пристрій, що надіслав запит. Далі проводиться валідація структури JSON–даних, перевіряється коректність значень та їх відповідність очікуваним діапазнам. Це дозволяє уникнути запису некоректної або пошкодженої інформації в систему.

Після успішної валідації телеметричні дані зберігаються в локальній базі даних SQLite у таблиці `telemetry`. Такий підхід забезпечує простоту розгортання системи та достатню продуктивність для невеликих і середніх IoT-проектів. Паралельно із записом даних оновлюється інформація про стан пристрою в системі, зокрема поле `last_seen`, яке фіксує час останньої активності пристрою, а також статус `online/offline`. Це дозволяє в режимі реального часу визначати доступність кожного IoT-вузла.

Окрім основного процесу передачі даних, система підтримує механізм зворотного керування пристроями. Цей механізм реалізований через періодичні запити ESP32 до endpoint `/commands/{device_id}`. Під час такого запиту пристрій передає свій ідентифікатор та JWT-токен, а сервер, у разі наявності активних команд, повертає відповідну інструкцію у форматі JSON.

Отримана команда аналізується на стороні пристрою, після чого виконується відповідна дія. Наприклад, система може передати команду на вмикання або вимикання світлодіода (LED), що використовується як базовий елемент демонстрації керування. Такий підхід дозволяє реалізувати двосторонню взаємодію між сервером і IoT-пристроєм, що є ключовою особливістю сучасних систем «розумного будинку».

Важливою складовою архітектури є використання проміжного шару NGINX. Усі HTTP-запити спочатку проходять через цей сервер, який виконує функції HTTPS-термінації, маршрутизації запитів та базового захисту від зовнішніх атак. Використання HTTPS забезпечує шифрування даних між клієнтом і сервером, що є критично важливим у контексті передачі телеметричної інформації та команд керування.

Окрім основного функціоналу обробки запитів, сервер FastAPI також експортує метрики системи через endpoint `/metrics` у форматі, сумісному з Prometheus. До таких метрик належать кількість HTTP-запитів, середній час їх обробки, кількість помилок авторизації, а також поточні значення телеметричних даних.

Система Prometheus періодично здійснює опитування цього endpoint та зберігає отримані метрики у своїй часовій базі даних. Це дозволяє накопичувати історичні дані про роботу системи та проводити подальший аналіз її продуктивності та стабільності.

На основі даних, зібраних Prometheus, система Grafana будує графічні панелі (dashboard), які відображають ключові показники роботи системи у зручному для користувача вигляді. До таких показників належать навантаження на сервер, кількість активних пристроїв, зміни температури та вологості в часі, а також статистика запитів до API. Завдяки цьому користувач має можливість у режимі реального часу відстежувати стан всієї системи та оперативно реагувати на можливі відхилення.

У цілому, запропонована архітектура забезпечує ефективну інтеграцію IoT-пристроїв із серверною інфраструктурою та системами моніторингу, що дозволяє реалізувати повноцінну екосистему «розумного будинку» з підтримкою безпечної комунікації, масштабованості та аналітики в реальному часі.

3.2 Розробка структурної схеми

Структурна схема відображає загальну організацію взаємодії між основними апаратними та програмними компонентами, а також напрямки передачі даних у межах системи. Вона визначає логічну архітектуру рішення, розподіл функцій між модулями та принципи обміну інформацією.

ESP32 пристрій – периферійний IoT-вузол системи. Він підключається до мережі Wi-Fi, проходить автентифікацію на сервері за допомогою JWT-токена, передає телеметричні дані (температуру та вологість) та отримує команди керування.

Передача даних між ESP32 і серверною частиною здійснюється через NGINX, який виступає зворотним проксі-сервером (Reverse Proxy). Його основною функцією є забезпечення захищеного з'єднання HTTPS за допомогою

SSL/TLS сертифікатів, а також перенаправлення запитів до серверного застосунку.

Центральним елементом системи є FastAPI Backend, який реалізує логіку роботи системи. Сервер виконує автентифікацію пристроїв, перевірку JWT-токенів відповідно до концепції Zero Trust, обробляє телеметричні дані, надає REST API для взаємодії пристроїв та формує метрики моніторингу.

Для зберігання інформації використовується база даних SQLite, яка містить таблиці пристроїв (devices) та телеметричних даних (telemetry). Доступ до бази реалізовано через ORM-бібліотеку SQLAlchemy.

Контроль працездатності системи виконує Prometheus, який періодично збирає метрики FastAPI через endpoint /metrics, зокрема кількість HTTP-запитів, час відповіді сервера, кількість помилок автентифікації та показники датчиків.

Для візуального аналізу телеметрії та стану системи використовується Grafana, яка отримує дані від Prometheus та формує інформаційні панелі з графіками температури, вологості, навантаження системи та інших показників.

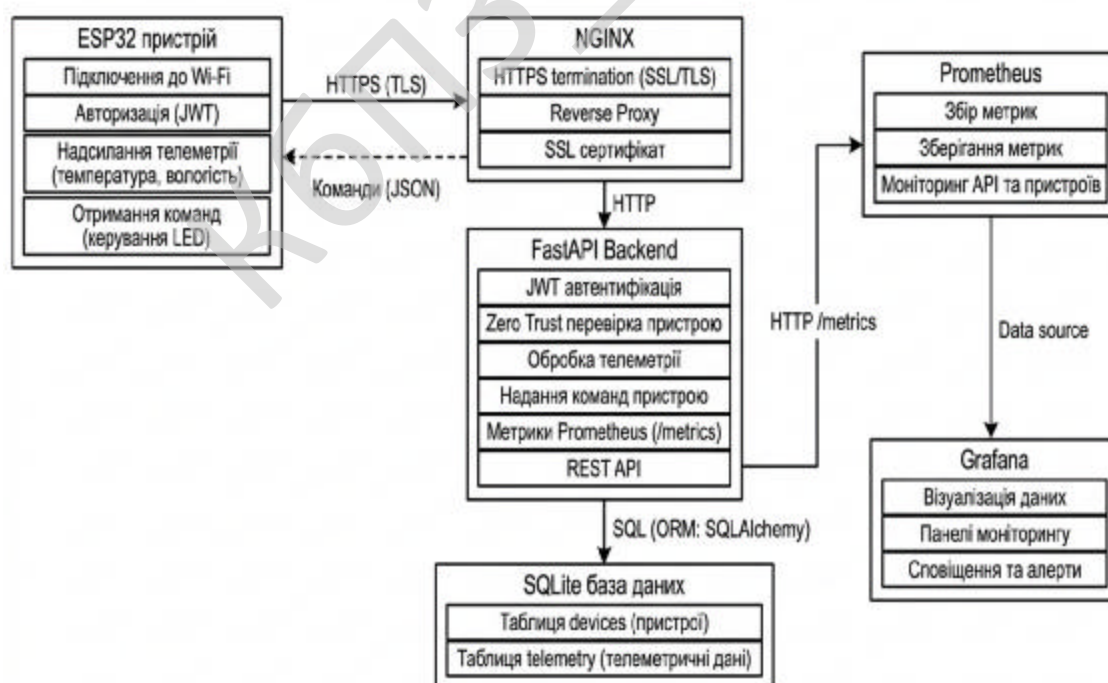


Рисунок 3.1 – Структурна схема системи захисту інформації

3.3 Розробка функціональної схеми

Функціональна схема системи відображає логіку взаємодії між основними програмними та апаратними компонентами, а також порядок обробки даних у процесі роботи системи. На схемі представлено послідовність передачі інформації між IoT-пристроєм, серверною частиною, базою даних та підсистемою моніторингу.

Основним елементом системи є IoT-пристрій на базі мікроконтролера ESP32, ЩО виконує функції збору даних із датчиків, підключення до мережі Wi-Fi і взаємодії із сервером через HTTPS-з'єднання. Пристрій формує телеметричні дані, зокрема значення температури і вологості, а також може отримувати команди керування виконавчими механізмами.

ESP32 проходить процедуру автентифікації надсилаючи запиту на endpoint /auth/device-login, у якому передається унікальний ідентифікатор пристрою (device_id) та секретний ключ (device_secret). Серверна частина перевіряє наявність пристрою в базі даних, його активний статус та коректність облікових даних. У разі успішної автентифікації генерується JWT-токен, який використовується для підтвердження автентичності всіх наступних запитів. Цей підхід реалізує концепцію Zero Trust, згідно з якою кожна взаємодія повинна проходити перевірку незалежно від джерела запиту.

Передача даних між клієнтською і серверною частинами проходить через сервер NGINX, що виконує роль Reverse Proxy. Воно обробляє HTTPS-з'єднання, SSL/TLS-термінацію, а також маршрутизацію запитів до backend-застосунку. NGINX ізолює внутрішню серверну логіку від зовнішніх клієнтів.

Після проходження автентифікації ESP32 періодично надсилає телеметричні дані на endpoint /telemetry. Сервер FastAPI виконує перевірку JWT-токена, ідентифікацію пристрою та валідацію переданих даних. Отримані значення температури та вологості зберігаються в базі даних SQLite у таблиці

telemetry, а також оновлюється інформація про стан пристрою, зокрема час останнього підключення (last_seen) та поточний статус (online).

Крім передачі телеметрії, IoT-пристрій має можливість отримання команд керування через endpoint /commands/{device_id}. Сервер обробляє запит, виконує перевірку автентичності пристрою та повертає відповідну команду у форматі JSON. У межах реалізованого прототипу підтримується керування LED-індикатором шляхом передачі команд LED_ON та LED_OFF.

Важливою частиною є підсистема моніторингу. Backend-застосунок експортує метрику через endpoint /metrics у форматі Prometheus. До них належать кількість HTTP-запитів, час обробки запитів, кількість невдалих спроб авторизації, а також останні отримані значення температури і вологості для кожного пристрою. Система Prometheus здійснює регулярний збір цих даних і забезпечує їх централізоване зберігання.

Щоб візуалізувати і аналізувати показники використовується Grafana, що підключається до Prometheus. Grafana формує панелі (dashboards), що дозволяють у реальному часі контролювати стан IoT-пристроїв, аналізувати телеметрію, відстежувати помилки авторизації і навантаження на сервер.

Короткий опис функціональної схеми системи

- ESP32 збирає дані (температура, вологість), підключається до Wi-Fi та передає телеметрію
- Виконує автентифікацію через /auth/device-login і отримує JWT
- Дані надсилаються на /telemetry, сервер їх перевіряє і зберігає в SQLite
- Через /commands пристрій отримує команди керування (LED ON/OFF)
- Весь трафік проходить через NGINX (HTTPS, reverse proxy)
- FastAPI обробляє запити та керує логікою системи
- Метрики експортуються через /metrics у Prometheus
- Prometheus збирає дані, Grafana візуалізує (дашборди, стан системи)

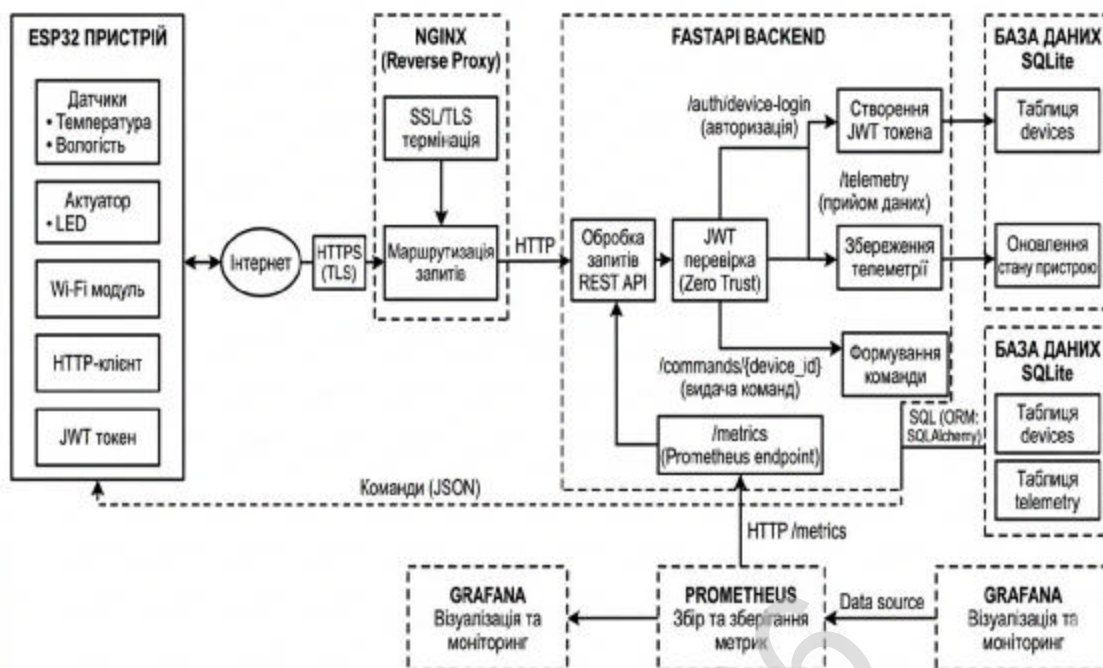


Рисунок 3.2 – Функціональна схема розробленого ПЗ

3.4 Розробка діаграми процесів

Діаграма процесів відображає логіку функціонування розробленої системи і представляє поведінку систему у вигляді графа процесів. Вершина графа – основні дії, що виконуються на стороні пристрою і серверної частини, а ребрами – переходи між цими діями, що здійснюються після завершення відповідного процесу.

Процеси подані у вигляді овальних блоків, які відповідають принципу представлення станів і дій у графах скінчених автоматів. Перехід відбувається за умови завершення попереднього етапу або отримання результату.

Початковий процес – ініціалізація роботи пристрою ESP32, запуск основного циклу виконання. Потім система виконує підключення до Wi-Fi або повторне підключення при втраті з'єднання. Це критичний етап, оскільки він забезпечує взаємодію пристрою з сервером.

Після цього йде процес автентифікації пристрою через механізми Zero Trust із використанням JWT–токена. Пристрій надсилає облікові дані на сервер, після чого при успішній перевірці отримує токен, що використовується для подальших запитів до API.

Після вдалої автентифікації виконується перевірка валідності токена, що забезпечує захищений доступ до ресурсів системи. При успішній перевірці система переходить до збору та передачі телеметричних даних.

Телеметрія включає формування і відправку даних сенсорів, зокрема температури та вологості, на сервер через HTTP–запит. Дані зберігаються у базі даних SQLite і паралельно передаються до системи моніторингу Prometheus, де формується метрики.

На основі цих метрик система Grafana здійснює візуалізацію стану пристроїв і показників середовища у вигляді дашбордів.

Окремим процесом йде перевірка команд керування, що надходять від сервера до пристрою. ESP32 періодично виконує HTTP–запит до API і отримує команди керування.

Завершальним етапом діаграми є циклічне повторення процесів із заданим інтервалом часу, що забезпечує безперервну роботу системи в режимі реального часу.

Короткий опис діаграми процесів системи:

- Ініціалізація ESP32 та запуск основного циклу роботи
- Підключення до Wi-Fi (з можливістю повторного підключення)
- Автентифікація пристрою через /auth/device-login і отримання JWT
- Перевірка валідності токена перед виконанням запитів
- Збір телеметричних даних (температура, вологість)
- Передача даних на сервер через /telemetry
- Збереження даних у SQLite та оновлення статусу пристрою
- Експорт метрик у Prometheus через /metrics
- Візуалізація даних у Grafana (дашборди)

- Перевірка та отримання команд через /commands/{device_id}
- Виконання команд на ESP32 (наприклад LED ON/OFF)
- Циклічне повторення всіх процесів у реальному часі

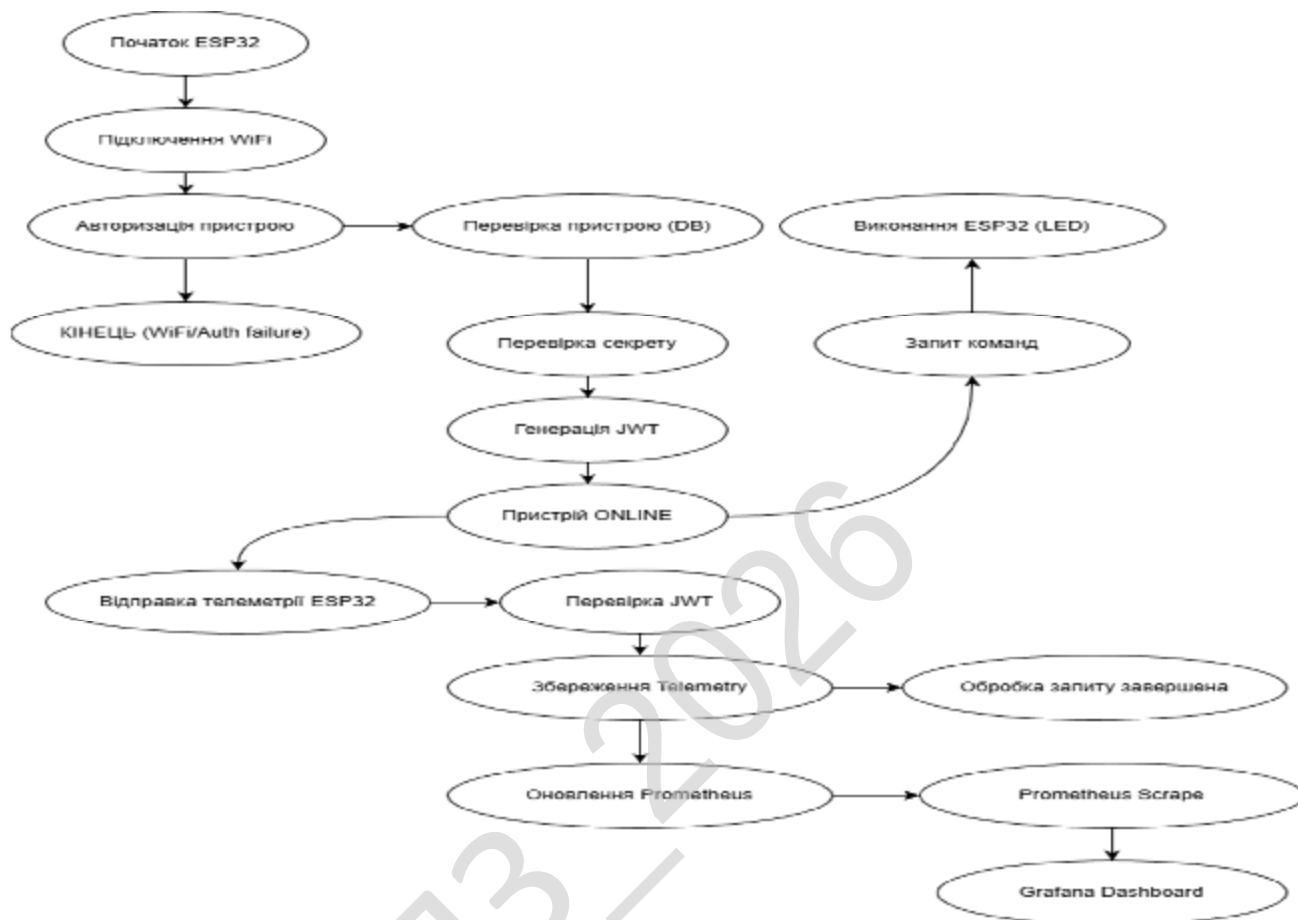


Рисунок 3.3 – Діаграма процесів розробленого ПЗ

Дана діаграма дозволяє відобразити взаємодію між усіма компонентами системи і чітко структурувати послідовність виконання процесів. Циклічний характер роботи забезпечує безперервний моніторинг стану середовища та постійний обмін даними між пристроєм і сервером. Використання підходу Zero Trust у процесі автентифікації підвищує рівень безпеки системи, оскільки кожен запит додатково перевіряється незалежно від попереднього стану з'єднання. Крім того, така модель дозволяє масштабувати систему шляхом додавання нових пристроїв без зміни загальної логіки взаємодії. У результаті діаграма процесів відображає не лише технічну послідовність операцій, але й загальні принципи побудови безпечної та розподіленої IoT-архітектури.

4. РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ ТА ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ І ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок–схем та опис алгоритмів функціонування системи

У даному розділі наведено опис практичної реалізації системи «розумного будинку» з використанням технологій Інтернету речей, REST API, JWT–автентифікації та систем моніторингу мережевої активності. Також розглянуто алгоритми роботи основних компонентів системи, представлені у вигляді блок–схем та діаграм процесів, що дозволяє детально проаналізувати логіку функціонування розробленого програмного забезпечення.

Для програмної реалізації системи було обрано мови програмування Python та C++. Використання Python зумовлено необхідністю реалізації серверної частини системи, яка відповідає за прийом, обробку та збереження даних, а також організацію механізмів автентифікації і моніторингу. У свою чергу, мова C++ використана для програмування IoT–пристрою ESP32 в середовищі Arduino IDE, що дозволило реалізувати механізми підключення до мережі, взаємодії із сервером та передачі телеметричних даних.

Архітектурно програмне забезпечення побудоване за модульним принципом, що дозволяє логічно розділити функціональність системи на окремі незалежні компоненти. Такий підхід спрощує процес розробки, тестування, модернізації та масштабування програмного забезпечення. Крім того, модульність забезпечує можливість незалежного оновлення окремих компонентів без необхідності повного перепроєктування системи.

Розроблена система включає у себе такі основні модулі:

– модуль IoT–пристрою (ESP32);

- серверний модуль FastAPI;
- модуль автентифікації та контролю доступу;
- модуль бази даних SQLite;
- модуль моніторингу та візуалізації;
- модуль зворотного проксі-сервера HTTPS.

Одним із ключових компонентів системи є серверний модуль, реалізований на основі FastAPI. Він забезпечує прийом HTTP-запитів від IoT-пристроїв, обробку телеметричних даних, перевірку автентичності пристроїв та формування відповідей для клієнтської частини. Сервер виконує роль центрального вузла системи, через який здійснюється взаємодія між пристроями, базою даних та засобами моніторингу.

Для реалізації механізму автентифікації використано підхід Zero Trust, відповідно до якого кожен пристрій повинен пройти перевірку незалежно від розташування в мережі. Процес автентифікації реалізується через endpoint /auth/device-login, куди пристрій передає власний ідентифікатор (device_id) та секретний ключ (device_secret). У випадку успішної перевірки сервер формує JWT-токен доступу, який у подальшому використовується під час передачі телеметричних даних та отримання команд керування.

Важливим елементом архітектури є пристрій, реалізований на базі мікроконтролера ESP32. Пристрій підключається до Wi-Fi мережі, проходить процедуру автентифікації, генерації телеметрії та передачі їх на серверну частину. У програмній реалізації ESP32 періодично виконує запити до серверу, надсилаючи інформацію про стан середовища, зокрема температуру та вологість.

Передача інформації між пристроєм і сервером виконується через HTTPS-з'єднання, реалізоване за допомогою проксі-сервера Nginx. Використання TLS-сертифікатів забезпечує шифрування мережевого трафіку та захист переданих даних від перехоплення. Nginx виконує роль проміжного рівня між клієнтською та серверною частинами, забезпечуючи безпечну маршрутизацію HTTP-запитів до FastAPI-сервера.

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		27

Щоб забезпечити зберігання інформації про пристрої і телеметрію використано SQLite. У базі даних реалізовані окремі таблиці для збереження інформації про зареєстровані пристрої і отриману телеметрію. Серверна частина автоматично оновлює статус активності пристрою, час останнього підключення, отримані показники температури і вологості.

Окрему роль у системі виконує модуль моніторингу, побудований на основі Prometheus та Grafana. Prometheus здійснює збір службових метрик системи, таких як кількість HTTP-запитів, час їх виконання, кількість помилок автентифікації та останні значення температури й вологості. Для візуального представлення статистичних даних використано Grafana, що забезпечує можливість створення інформаційних панелей моніторингу в режимі реального часу.

Взаємодія між компонентами системи відбувається поступово. IoT-пристрій виконує підключення до бездротової мережі і ініціює процес автентифікації. Після успішного отримання JWT-токена пристрій переходить до циклічного режиму роботи, у межах якого виконується передача телеметричних даних, оновлення статусу активності та перевірка наявності команд керування. Отримані сервером дані зберігаються у базі даних і одночасно передаються до системи моніторингу.

Під час розробки програмного забезпечення було сформовано набір алгоритмів, які описують логіку роботи основних функціональних компонентів системи. Для підвищення наочності вони представлені у вигляді блок-схем та діаграм процесів. До основних алгоритмів належать:

- алгоритм запуску системи та підключення до мережі Wi-Fi;
- алгоритм автентифікації пристрою із використанням JWT-токена;
- алгоритм передачі телеметричних даних;
- алгоритм збереження даних та оновлення метрик;
- алгоритм отримання команд керування пристроєм;
- алгоритм збору статистичних показників системи.

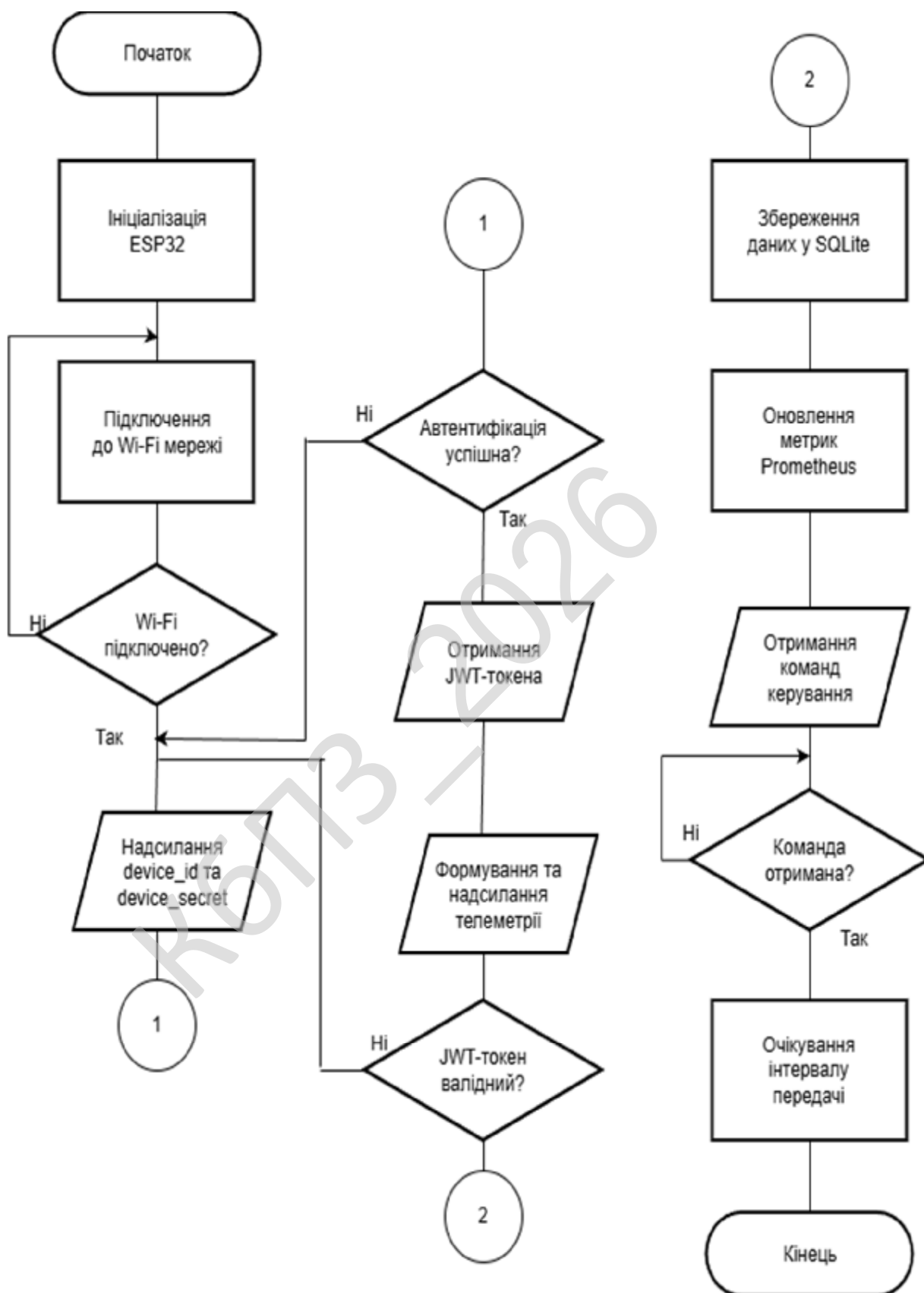


Рисунок 4.1 – Блок-схема розробленого алгоритму

Алгоритм запуску та автентифікації пристрою

Даний алгоритм описує процес ініціалізації IoT-пристрою та проходження процедури автентифікації в системі «розумного будинку». Він реалізований на стороні мікроконтролера ESP32 і забезпечує безпечне підключення пристрою до серверної частини системи з використанням підходу Zero Trust та JWT-автентифікації.

Алгоритм починається з етапу запуску пристрою, під час якого ініціалізуються основні апаратні компоненти, зокрема серійний інтерфейс для налагодження та вихідні GPIO-піни для керування виконавчими елементами. Після цього пристрій переходить до процедури підключення до бездротової мережі Wi-Fi.

У разі відсутності з'єднання з мережею виконується повторна спроба підключення до моменту успішного встановлення зв'язку. Це забезпечує стійкість роботи пристрою в умовах нестабільного мережевого середовища.

Після встановлення мережевого з'єднання пристрій ініціює процес автентифікації на сервері шляхом надсилання HTTP POST-запиту на endpoint /auth/device-login. У запиті передаються ідентифікатор пристрою (device_id) та секретний ключ (device_secret), які зберігаються локально в прошивці пристрою.

На серверній стороні виконується перевірка отриманих облікових даних шляхом їх порівняння з записами у базі даних. У разі успішної перевірки сервер формує JWT-токен доступу, який повертається пристрою у відповіді.

Отриманий JWT-токен зберігається в оперативній пам'яті пристрою та використовується для подальших запитів до серверної частини системи. У випадку невдалої автентифікації пристрій повторює процедуру логіну до моменту отримання валідного токена.

Далі алгоритм переходить у робочий режим, у межах якого пристрій періодично ініціює перевірку з'єднання та готовності до передачі телеметричних даних. Автентифікований стан пристрою є обов'язковою умовою для виконання будь-яких операцій взаємодії із сервером.

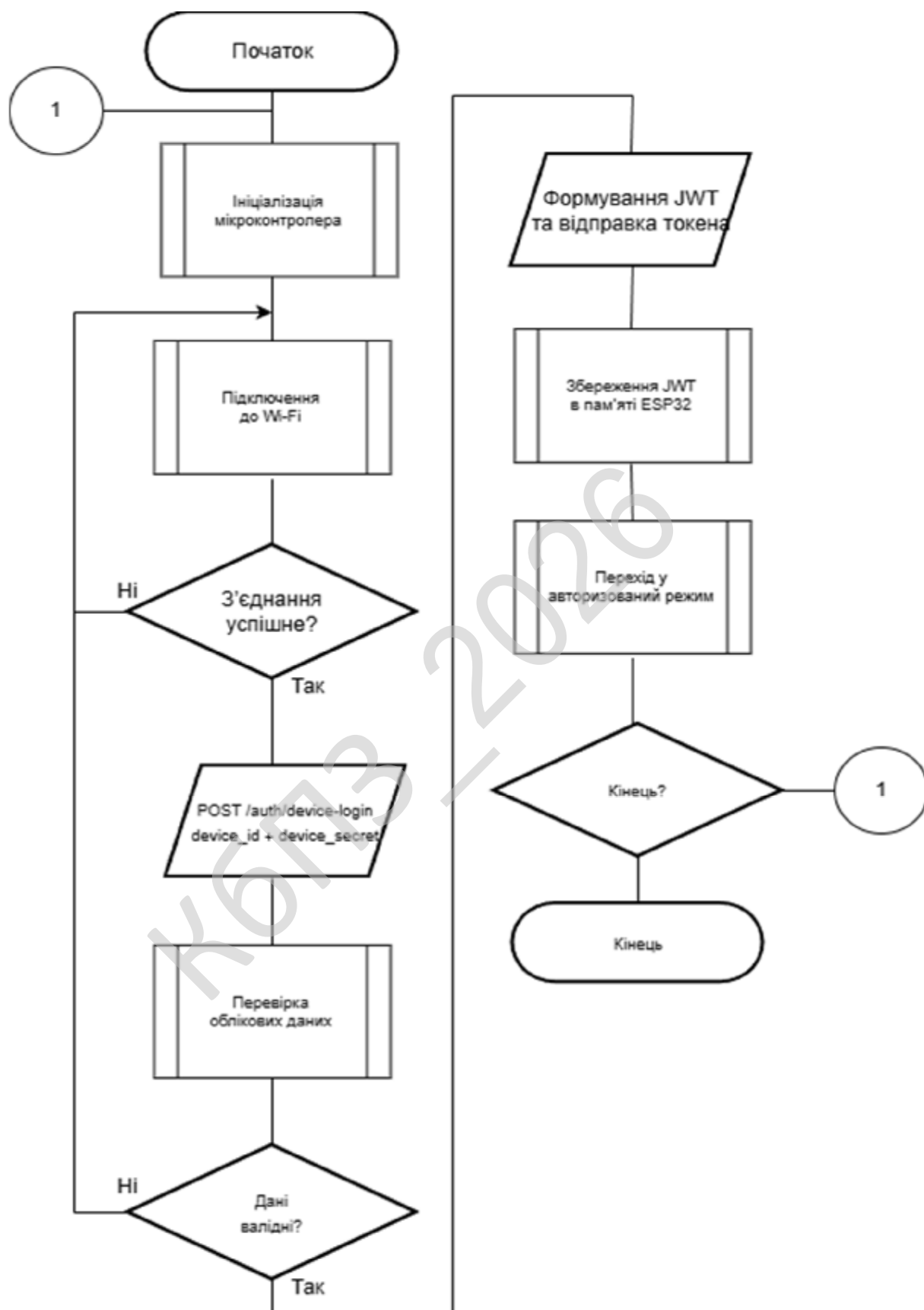


Рисунок 4.2 – Блок-схема алгоритму запуску та автентифікації пристрою

Алгоритм передачі телеметричних даних

У даному підрозділі розглядається алгоритм передачі телеметричних даних від IoT-пристрою до серверної частини системи. Основною метою даного процесу є забезпечення надійного та безпечного обміну інформацією між пристроями на базі ESP32 та сервером, реалізованим із використанням FastAPI, з подальшим збереженням даних у базі даних і їх використанням у системі моніторингу.

Після успішного проходження етапу автентифікації і отримання JWT-токена пристрій переходить у режим періодичної передачі телеметричних даних. До складу телеметрії можуть входити показники з датчиків (температура, вологість, стан пристрою, рівень сигналу Wi-Fi і інші параметри залежно від конфігурації системи).

Формування телеметричних даних здійснюється на стороні пристрою. Зібрані значення об'єднуються у структуроване повідомлення, яке включає ідентифікатор пристрою, часову мітку і корисне навантаження (payload). Для уніфікації обміну даними використовується формат JSON, що забезпечує простоту інтеграції з серверною частиною і подальшу масштабованість системи.

Сформоване повідомлення може додатково оброблятися криптографічним модулем (за наявності захищеного каналу зв'язку), після чого передається на серверну частину через HTTP-запит до REST API. У заголовках запиту обов'язково передається JWT-токен, який використовується для перевірки прав доступу пристрою.

На стороні сервера FastAPI відбувається прийом вхідного запиту і перевірка валідності JWT-токена. У разі успішної автентифікації дані приймаються до подальшої обробки. Якщо токен є недійсним або простроченим, запит відхиляється, а пристрій переходить у стан повторної автентифікації.

Після перевірки автентичності дані зберігаються у базі даних SQLite. Це дозволяє забезпечити довготривале збереження історії телеметрії і подальший аналіз стану пристроїв. Паралельно з цим відбувається оновлення метрик у

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		32

системі моніторингу Prometheus, що дозволяє в режимі реального часу відслідковувати стан IoT-інфраструктури.

Загальний алгоритм передачі телеметрії включає такі основні етапи:

- формування телеметричних даних на пристрої;
- серіалізація даних у формат JSON;
- додавання JWT-токена до запиту;
- передача даних на сервер через REST API;
- перевірка автентичності запиту;
- збереження даних у базі даних;
- оновлення метрик моніторингу.

Алгоритм обробки та виконання команд керування

У даному підрозділі розглядається алгоритм обробки та виконання команд керування IoT-пристроями в межах розробленої системи. Даний процес реалізує зворотний канал зв'язку між серверною частиною (FastAPI) та пристроями на базі ESP32, що дозволяє здійснювати дистанційне керування станом пристроїв у режимі реального часу.

Команди керування формуються на стороні серверної частини системи або ініціюються користувачем через API-інтерфейс. Кожна команда містить тип операції (наприклад, увімкнення/вимкнення пристрою, зміна режиму роботи або оновлення параметрів конфігурації), ідентифікатор цільового пристрою та додаткові параметри виконання.

Перед передачею на пристрій команда проходить етап авторизації, який базується на використанні JWT-токена. Це забезпечує захист від несанкціонованого доступу до функцій керування та гарантує, що лише авторизовані запити можуть бути виконані пристроями.

На серверній частині сформована команда передається до пристрою через REST API або постійно активне з'єднання (залежно від реалізації системи). ESP32 періодично опитує сервер або приймає вхідні повідомлення, перевіряючи наявність нових команд, призначених для нього.

Після отримання команди на стороні пристрою виконується її первинна обробка. На цьому етапі перевіряється:

- відповідність ідентифікатора пристрою;
- коректність структури команди;
- наявність валідного JWT-токена (якщо використовується в зворотному каналі);
- допустимість виконання команди в поточному стані пристрою.

У разі успішної перевірки команда передається до модуля виконання. Цей модуль реалізує безпосередню взаємодію з апаратною частиною пристрою (GPIO-піни, реле, датчики або інші виконавчі механізми). Наприклад, команда може змінювати стан вихідного сигналу, активувати підключений пристрій або змінювати режим роботи системи.

Після виконання команди формується службове повідомлення про результат виконання (успішно або з помилкою), яке передається назад на сервер для фіксації стану системи та подальшого моніторингу. Це дозволяє реалізувати повний цикл зворотного зв'язку між сервером і IoT-пристроєм.

Основні етапи алгоритму виконання команд керування включають:

- отримання команди від серверної частини;
- перевірка автентичності та прав доступу;
- аналіз структури та параметрів команди;
- передача команди до виконавчого модуля;
- виконання дії на апаратному рівні;
- формування статусу виконання;
- передача відповіді на сервер.

4.2 Захист розробленого програмного забезпечення

У межах розробленої системи для IoT-пристроїв реалізовано багаторівневу модель захисту, яка поєднує механізми автентифікації,

авторизації, контролю доступу і моніторингу стану системи. Архітектура програмного забезпечення побудована за принципом Zero Trust, відповідно до якого кожен пристрій та кожен запит перевіряється незалежно від попереднього стану довіри.

Автентифікація та управління доступом (JWT)

Основним механізмом автентифікації пристроїв у системі є використання JWT (JSON Web Token). Після успішної ідентифікації IoT-пристрою сервер генерує токен доступу, який використовується для подальших запитів до API.

Генерація токена виконується у функції `create_token()`:

```
def create_token(data: dict):
    payload = data.copy()
    payload["exp"] = datetime.utcnow() + timedelta(hours=6)
    return jwt.encode(payload, SECRET_KEY, algorithm=ALGORITHM)
```

Сформований токен додається часовий параметр `exp`, який обмежує термін дії і зменшує ризики повторного використання (replay attacks). У `payload` також передаються ідентифікатор пристрою (`sub`) та його роль (`role`).

Процес входу IoT-пристрою реалізовано через ендпоінт `/auth/device-login`. Пристрій передає свій ідентифікатор і секрет, після чого сервер виконує перевірку у базі даних.

Фрагмент реалізації:

```
@app.post("/auth/device-login")
def device_login(data: DeviceLogin, db: Session = Depends(get_db)):
    device = db.query(Device).filter(Device.device_id ==
data.device_id).first()

    if not device:
        FAILED_LOGINS.inc()
        raise HTTPException(status_code=401, detail="Unknown device")

    if device.device_secret != data.device_secret:
        FAILED_LOGINS.inc()
        raise HTTPException(status_code=401, detail="Invalid secret")
```

При успішній перевірці пристрою встановлюється статус `online`, оновлюється час останньої активності та генерується JWT-токен.

Передача телеметрії реалізована через ендпоінт /telemetry, який приймає лише авторизовані запити з дійсним токеном.

Перевірка токена виконується функцією require_device():

```
def require_device(token: str, db: Session):  
    try:  
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])  
    except:  
        return None
```

Додатково реалізовано перевірку відповідності пристрою (device mismatch), що унеможливлює підміну джерела даних.

Після успішної автентифікації система зберігає телеметрію у базі даних та оновлює оперативні метрики.

```
t = Telemetry(  
    device_id=data.device_id,  
    temperature=data.temperature,  
    humidity=data.humidity  
)
```

```
db.add(t)  
db.commit()
```

Значення температури та вологості передаються у Prometheus-графіки через Gauge-метрики:

```
TEMPERATURE_GAUGE.labels(device_id=data.device_id).set(data.temperature)  
HUMIDITY_GAUGE.labels(device_id=data.device_id).set(data.humidity)
```

У системі реалізовано повноцінний рівень спостереження за станом API за допомогою Prometheus метрик. Відстежуються:

- кількість HTTP-запитів;
- час обробки запитів;
- кількість невдалих логінів;
- останні значення сенсорів.

```
REQUEST_COUNT = Counter("http_requests_total", "Requests", ["method",  
"endpoint", "status"])  
REQUEST_TIME = Histogram("http_request_duration_seconds", "Request time")  
FAILED_LOGINS = Counter("failed_logins_total", "Failed logins")
```

Ендпоінт /metrics експортує дані у форматі, сумісному з системами моніторингу:

```
@app.get("/metrics")
def metrics():
    return Response(generate_latest(), media_type="text/plain")
```

Розгортання програмного комплексу здійснюється за допомогою контейнеризації (Docker), що забезпечує переносимість та ізоляцію компонентів системи.

У складі інфраструктури використовуються:

- FastAPI – backend API.
- Prometheus – збір метрик.
- Grafana – візуалізація даних.
- Nginx – HTTPS проксі.

Docker Compose конфігурація забезпечує одночасний запуск усіх сервісів:

```
services:
  backend:
  prometheus:
  grafana:
  nginx:
```

Для захисту мережевого трафіку використовується TLS-шифрування через Nginx як reverse проху. Усі запити до API проходять через HTTPS, що виключає можливість перехоплення JWT-токенів або телеметрії.

```
server {
    listen 443 ssl;

    ssl_certificate /etc/nginx/certs/cert.pem;
    ssl_certificate_key /etc/nginx/certs/key.pem;

    location / {
        proxy_pass http://backend:8000;
    }
}
```

Клієнтська частина системи реалізована на базі ESP32 та виконує роль емулятора IoT-вузла. Пристрій здійснює:

- підключення до Wi-Fi мережі;

- автентифікацію на сервері;
- періодичну передачу телеметрії;
- отримання команд керування.

Фрагмент реалізації передачі даних:

```
int code = http.POST(body);
```

Після отримання команди пристрій може керувати виконавчими механізмами (наприклад, LED):

```
if (command == "LED_ON") {
    digitalWrite(LED_PIN, HIGH);
}
```

К6ПЗ_2026

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		38

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблена система захищеного обміну даними для IoT-пристроїв орієнтована на використання в умовах реального часу, обмежених ресурсів мікроконтролерів та необхідності забезпечення безпечної передачі телеметрії. Архітектура рішення базується на взаємодії ESP32-пристроїв із серверною частиною, реалізованою на FastAPI, із використанням JWT-автентифікації, бази даних SQLite та систем моніторингу Prometheus і Grafana.

Завдяки впровадженню принципів Zero Trust Security модель системи передбачає, що кожен запит від пристрою або клієнта перевіряється незалежно від попереднього стану з'єднання. Це дозволяє мінімізувати ризики несанкціонованого доступу навіть у випадку компрометації окремих компонентів мережі.

Основними напрямками практичного застосування розробленої системи:

- IoT-пристрої промислового моніторингу, де критично важливим є контроль стану обладнання та захист телеметричних даних від підміни або перехоплення;
- системи «розумного будинку», у яких необхідно забезпечити конфіденційність персональних даних користувачів та безпечне керування пристроями;
- системи моніторингу інфраструктури (енергетика, водопостачання, логістика), де дані передаються в реальному часі та потребують гарантії цілісності;
- навчальні та тестові IoT-платформи, які використовуються для моделювання мереж пристроїв без необхідності фізичного розгортання великої кількості обладнання.

Важливим етапом впровадження системи є тестування її основних компонентів та перевірка коректності роботи в умовах, наближених до реальної

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		39

експлуатації. У процесі тестування кожен модуль системи був перевірений окремо, а також у складі цілісної архітектури.

Процес тестування включав такі етапи:

- перевірка коректності ініціалізації ESP32 та встановлення з'єднання з Wi-Fi мережею;
- тестування механізму JWT-автентифікації між пристроєм та сервером FastAPI;
- перевірка передачі телеметричних даних у форматі JSON та їх коректної обробки сервером;
- тестування роботи бази даних SQLite при збереженні великого обсягу телеметрії;
- перевірка інтеграції з Prometheus для збору метрик та Grafana для візуалізації даних;
- оцінка стійкості системи до втрати пакетів та повторного підключення пристроїв.

Окрему увагу було приділено оцінці продуктивності системи при збільшенні кількості одночасно підключених пристроїв. Результати показали, що використання FastAPI як асинхронного фреймворку дозволяє обробляти велику кількість запитів без суттєвого зростання затримок.

Також було проведено аналіз навантаження на серверну частину при інтенсивній передачі телеметрії. Встановлено, що використання JSON-структур для повідомлень забезпечує баланс між зручністю обробки даних і накладними витратами на передачу.

Розроблена система має низку переваг, серед яких слід виділити модульність архітектури, що дозволяє легко масштабувати рішення та інтегрувати додаткові сервіси. Використання FastAPI забезпечує високу продуктивність, а JWT-автентифікація – контроль доступу на рівні кожного запиту.

Перспективи розвитку системи включають:

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		40

- [illegible]

New panel

Legend:

- (endpoint="/auth/device-log n", method="POST", status="200")
- (endpoint="/commands/exp32_001", method="GET", status="200")
- (endpoint="/metrics", method="GET", status="200")
- (endpoint="/Accelerometry", method="POST", status="200")
- (endpoint="/to energy", method="POST", status="403")
- (name="/device.temperature", device_id="/exp32_001", instance="172170.18030", job="fastapi")
- (name="/device.humidity", device_id="/exp32_001", instance="172170.18030", job="fastapi")

Рисунок 5.2 – Результати Grafan

К6ПЗ_2026

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		42

ОСНОВНІ ВИСНОВКИ

У результаті виконання бакалаврської дипломної роботи було досягнуто поставленої мети – розроблено та реалізовано програмну систему захищеного обміну даними для IoT-пристроїв із підтримкою автентифікації, контролю доступу та моніторингу в реальному часі.

У ході дослідження проведено аналіз сучасних підходів до побудови IoT-систем та методів забезпечення їх кібербезпеки. На основі цього було обґрунтовано доцільність використання архітектури клієнт-сервер із застосуванням REST API, JWT-автентифікації та принципів Zero Trust Security, що забезпечує перевірку кожного запиту незалежно від попереднього стану сесії.

Розроблена система базується на взаємодії ESP32-пристроїв із серверною частиною, реалізованою на FastAPI. Для зберігання даних використовується SQLite, що дозволяє забезпечити простоту розгортання та достатню продуктивність для IoT-сценаріїв. Передача телеметрії здійснюється у форматі JSON, що забезпечує універсальність та легку інтеграцію з іншими сервісами.

Окрему увагу приділено механізму автентифікації та авторизації користувачів і пристроїв. Використання JWT-токенів дозволяє забезпечити безпечний доступ до API та мінімізує ризики несанкціонованого використання системи. Запропонований підхід забезпечує баланс між рівнем безпеки та продуктивністю обробки запитів.

Практична цінність роботи полягає у створенні модульної архітектури, яка може бути легко масштабована та інтегрована у різні сфери застосування IoT. Система придатна для використання у промисловому моніторингу, «розумних будинках», системах віддаленого контролю обладнання та інших розподілених середовищах.

Важливим результатом є також інтеграція системи моніторингу на основі Prometheus та Grafana, що дозволяє здійснювати збір, аналіз та візуалізацію

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		43

метрик у режимі реального часу. Це забезпечує можливість контролю стану серверної частини, навантаження системи та кількості активних пристроїв.

Проведені експериментальні перевірки підтвердили стабільність роботи системи при навантаженні, а також коректність обробки запитів при одночасному підключенні декількох пристроїв. Система демонструє стійкість до втрати з'єднання та здатність до відновлення роботи без втрати критичних даних.

Перспективи подальшого розвитку полягають у впровадженні механізмів автоматичного оновлення та ротації JWT-токенів, розширенні підтримки хмарних IoT-платформ, а також інтеграції алгоритмів виявлення аномалій у телеметричних даних із використанням методів машинного навчання.

У підсумку, розроблена система є завершеним функціональним рішенням для захищеної взаємодії IoT-пристроїв із серверною інфраструктурою та може бути використана як основа для подальших досліджень і практичних впроваджень у сфері кібербезпеки Інтернету речей.

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		44

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) IoT security and smart systems architecture // Wiley Online Library. – Режим доступу: <https://onlinelibrary.wiley.com/doi/full/10.1155/2022/6476274> (дата звернення: 22.05.2026).

2) Security mechanisms in IoT networks // Wiley Online Library. – Режим доступу: <https://onlinelibrary.wiley.com/doi/full/10.1155/2022/6367579> (дата звернення: 22.05.2026).

3) NIST SP 800–183: Networks of ‘Things’. – Режим доступу: <https://csrc.nist.gov/publications/detail/sp/800–183/final> (дата звернення: 22.05.2026).

4) NIST SP 800–53 Security Controls. – Режим доступу: <https://csrc.nist.gov/publications/detail/sp/800–53/rev–5/final> (дата звернення: 22.05.2026).

5) OpenSSL Documentation. – Режим доступу: <https://docs.openssl.org/> (дата звернення: 22.05.2026).

6) OpenSSL EVP Interface. – Режим доступу: <https://www.openssl.org/docs/manmaster/man7/evp.html> (дата звернення: 22.05.2026).

7) ESP–IDF Programming Guide (Security). – Режим доступу: <https://docs.espressif.com/projects/esp-idf/> (дата звернення: 22.05.2026).

8) ACM IoT Security Frameworks. – Режим доступу: <https://dl.acm.org/doi/10.1145/3491202> (дата звернення: 22.05.2026).

9) IoT Intrusion Detection Systems Survey. – Режим доступу: <https://doi.org/10.1016/j.adhoc.2023.103320> (дата звернення: 22.05.2026).

10) IoT Systems and Applications // Techsphere Journals. – Режим доступу: <https://stem.techspherejournals.com/index.php/tsjpas/article/view/1/1> (дата звернення: 22.05.2026).

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		45

- 11) Token-Based Authentication in IoT Networks // ResearchGate. – Режим доступу: https://www.researchgate.net/profile/Akhilesh-Wao-2/publication/357157112_DESIGN_A_NOVEL_APPROACH_FOR_TOKEN_BASED_AUTHENTICATION_IN_IOT_NETWORKS (дата звернення: 22.05.2026).
- 12) Flash Encryption Guide for ESP32 // Espressif Systems. – Режим доступу: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/security/flash-encryption.html> (дата звернення: 22.05.2026).
- 13) ESP-TLS Secure Communication // Espressif Systems. – Режим доступу: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_tls.html (дата звернення: 22.05.2026).
- 14) RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3 // IETF. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc8446> (дата звернення: 22.05.2026).
- 15) RFC 2104: HMAC Keyed-Hashing for Message Authentication // IETF. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc2104> (дата звернення: 22.05.2026).
- 16) RFC 7519: JSON Web Token (JWT) // IETF. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 22.05.2026).
- 17) RFC 6749: The OAuth 2.0 Authorization Framework // IETF. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc6749> (дата звернення: 22.05.2026).
- 18) SHA-256 Cryptographic Standard // NIST. – Режим доступу: <https://csrc.nist.gov/publications/detail/fips/180/4/final> (дата звернення: 22.05.2026).
- 19) RSA Cryptography Guide // Stanford University. – Режим доступу: <https://crypto.stanford.edu/~dabo/cryptobook/> (дата звернення: 22.05.2026).
- 20) Introduction to Elliptic Curve Cryptography // neuromancer.sk. – Режим доступу: <https://neuromancer.sk/std/> (дата звернення: 22.05.2026).
- 21) IoT Reference Architecture (IIRA) // Industrial Internet Consortium. – Режим доступу: <https://www.iiconsortium.org/pdf/IIRA-v1.9.pdf> (дата звернення: 22.05.2026).

- 22) EVP Symmetric Encryption and Decryption // OpenSSL Wiki. – Режим доступу: https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption (дата звернення: 22.05.2026).
- 23) Elliptic Curve Cryptography on ESP32 // GitHub. – Режим доступу: <https://github.com/Bizyroth/Elliptic-Curve-on-ESP32> (дата звернення: 22.05.2026).
- 24) Threat Modeling Tool for IoT Security // Microsoft Learn. – Режим доступу: <https://learn.microsoft.com/en-us/security/engineering/threat-modeling-tool> (дата звернення: 22.05.2026).
- 25) Applied Cryptography: Protocols, Algorithms and Source Code // Bruce Schneier. – Режим доступу: https://www.schneier.com/books/applied_cryptography/ (дата звернення: 22.05.2026).
- 26) Zero Trust Architecture (NIST SP 800-207) – <https://csrc.nist.gov/publications/detail/sp/800-207/final> (дата звернення: 22.05.2026).
- 27) FastAPI Documentation – <https://fastapi.tiangolo.com/> (дата звернення: 22.05.2026).
- 28) NGINX Reverse Proxy Guide – <https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/> (дата звернення: 22.05.2026).
- 29) Prometheus Monitoring System Documentation – <https://prometheus.io/docs/introduction/overview/> (дата звернення: 22.05.2026).
- 30) Grafana Documentation – <https://grafana.com/docs/> (дата звернення: 22.05.2026).
- 31) SQLite Official Documentation – <https://www.sqlite.org/docs.html> (дата звернення: 22.05.2026).
- 32) SQLAlchemy Documentation – <https://docs.sqlalchemy.org/> (дата звернення: 22.05.2026).

33) Uvicorn ASGI Server Documentation – <https://www.uvicorn.org/> (дата звернення: 22.05.2026).

34) ASGI Specification (Python Web Stack) – <https://asgi.readthedocs.io/en/latest/> (дата звернення: 22.05.2026).

35) JWT Best Practices (Auth0 Docs) – <https://auth0.com/docs/secure/tokens/json-web-tokens> (дата звернення: 22.05.2026).

36) OAuth 2.0 in Practice (Auth0 Guide) – <https://auth0.com/docs/get-started/authentication-and-authorization-flow> (дата звернення: 22.05.2026).

37) TLS Configuration Best Practices – <https://ssl-config.mozilla.org/> (дата звернення: 22.05.2026).

38) Docker Documentation – <https://docs.docker.com/> (дата звернення: 22.05.2026).

39) Docker Compose Specification – <https://docs.docker.com/compose/> (дата звернення: 22.05.2026).

40) Linux Networking Concepts (Netfilter/NAT) – <https://www.netfilter.org/documentation/> (дата звернення: 22.05.2026).

41) Wi-Fi Security IEEE 802.11 Standards Overview – https://standards.ieee.org/standard/802_11-2020.html (дата звернення: 22.05.2026).

42) ESP32 Wi-Fi and Networking Stack – <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/index.html> (дата звернення: 22.05.2026).

43) ESP-IDF HTTP Client API – https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_http_client.html (дата звернення: 22.05.2026).

44) REST API Design Best Practices – <https://restfulapi.net/> (дата звернення: 22.05.2026).

45) OpenAPI Specification – <https://swagger.io/specification/> (дата звернення: 22.05.2026).

46) Pydantic Documentation – <https://docs.pydantic.dev/> (дата звернення: 22.05.2026).

47) Python Requests Library – <https://requests.readthedocs.io/en/latest/> (дата звернення: 22.05.2026).

48) Linux System Monitoring Basics – <https://www.kernel.org/doc/html/latest/> (дата звернення: 22.05.2026).

49) IoT Edge Security Overview (Microsoft Learn) – <https://learn.microsoft.com/en-us/azure/iot-edge/> (дата звернення: 22.05.2026).

50) Observability in Distributed Systems (OpenTelemetry) – <https://opentelemetry.io/docs/> (дата звернення: 22.05.2026).

К6ПЗ – 2026

					ВКРБ–125.26.00.14.00.00.ПЗ	Арк.
Вим.	Арк.	№ докум.	Підпис	Дата		49