

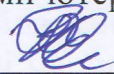
Міністерство освіти і науки України
Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки

ПОЯСНЮВАЛЬНА ЗАПИСКА

до випускної кваліфікаційної роботи бакалавра
на тему: «Проект локальної IoT-мережі на основі ві-
ртуального маршрутизатора MikroTik CHR»

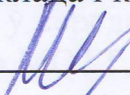
Виконав: студент 4-го курсу групи КІ-22-1

спеціальності 123 «Комп'ютерна інженерія»



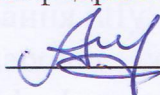
Павло ЗУБРИЦЬКИЙ

Керівник: Страший викладач кафедри КІ та КБ



Константин МАТВЄЄВ

Рецензент: Доцент кафедри КІ та КБ, к. т.н.



Олексій ШЕЛУХА

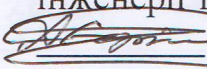
Житомир - 2026

Державний університет «Житомирська політехніка»
Факультет інформаційно-комп'ютерних технологій
Кафедра комп'ютерної інженерії та кібербезпеки
Спеціальність 123 «Комп'ютерна інженерія»
Освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач

кафедри комп'ютерної
інженерії та кібербезпеки

 Андрій Єфіменко

5 березня 2024 р.

ЗАВДАННЯ

на випускню кваліфікаційну роботу бакалавра

Студент: Зубрицький Павло Андрійович

Тема роботи: «Проект локальної IoT-мережі на основі віртуального маршрутизатора MikroTik CHR»

затверджена Наказом університету від 5 березня 2026 р. № 97

Термін здачі студентом закінченої роботи 15 червня 2026 р.

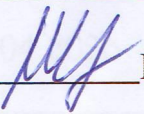
Вихідні дані роботи: Територіально розподілена офісна інфраструктура автоматизації з різним технологічним навантаженням; процеси передачі, обробки, агрегації та фільтрації телеметричних даних прикладного рівня; логічне розділення широкомовних доменів користувачів та пристроїв за допомогою технології мікросегментації; розгортання автономного прикладного стеку мікросервісів безпосередньо всередині ядра комутаційного маршрутизатора на базі віртуалізованої операційної системи MikroTik RouterOS v7 (Cloud Hosted Router).

Обладнання та програмне забезпечення: Персональний комп'ютер x86-64 (процесор з 12 логічними обчислювальними ядрами, 16 ГБ оперативної пам'яті DDR4), гіпервізор Oracle VirtualBox v7.2.2, графічний симулятор мережевих топологій GNS3 v2.2.49, мережева операційна система MikroTik RouterOS v7.21.4 (образ Cloud Hosted Router), віртуальні машини кінцевих хостів під керуванням операційних систем Ubuntu Linux LTS та Windows 10, легковажні вузли емуляції хостів VPCS. Стек мікросервісів у середовищі Container RouterOS включає брокер повідомлень Eclipse Mosquitto MQTT, сервер оркестрації та автоматизації потоків даних Node-RED, аналітичну систему моніторингу Grafana, сервер високопріоритетного пуш-інформування ntfy, локальний Git-репозиторій інфраструктурного коду Gitea, менеджер паролів і секретів Vaultwarden, зворотний проксі-сервер єдиної точки входу Caddy. Інструменти розробки та аналізу: мова програмування Python v3.8.10 та v3.14.6 з бібліотеками асинхронної мережевої взаємодії paho-mqtt та requests, графічний аналізатор MQTT-трафіку MQTT Explorer, інженерний мобільний клієнт для Android MQTT IoT Panel та додаток ntfy.

Консультанти з випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються

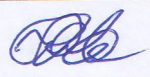
Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Михайло ХОХЛОВ	 05.03.2026	 05.03.2026
2	Михайло ХОХЛОВ	 05.03.2026	 05.03.2026
3	Михайло ХОХЛОВ	 05.03.2026	 05.03.2026

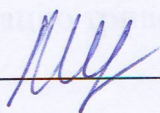
Студент  Павло ЗУБРИЦЬКИЙ

Керівник  Константин МАТВЄЄВ

Календарний план

№ з/п	Назви етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання, та збір вихідних даних	05.03.2026	Виконано
2	Аналіз архітектурних моделей IoT-мереж та протоколів прикладного рівня	18.03.2026	Виконано
3	Опрацювання літературних джерел та формування технічного завдання	28.03.2026	Виконано
4	Математичний розрахунок безкласового адресного простору CIDR та VLAN	12.04.2026	Виконано
5	Проектування архітектури мережі та інформаційних топіків MQTT	22.04.2026	Виконано
6	Розробка алгоритмів та написання програмного забезпечення на Python	04.05.2026	Виконано
7	Практичне розгортання та конфігурування середовища Container у RouterOS	14.05.2026	Виконано
8	Налаштування підсистеми міжмережевого екранування IP Firewall Filter (Zero Trust)	24.05.2026	Виконано
9	Тестування та перевірка працездатності комплексу в симуляторі GNS3	02.06.2026	Виконано
10	Написання та фінальне оформлення пояснювальної записки та додатків коду	10.06.2026	Виконано
11	Передзахист роботи та підготовка презентаційних матеріалів до захисту	15.06.2026	Виконано

Студент  Павло ЗУБРИЦЬКИЙ

Керівник  Константин МАТВЕЄВ

АНОТАЦІЯ

Кваліфікаційна робота виконана на тему: Проєкт локальної IoT-мережі на основі віртуального маршрутизатора MikroTik CHR. Пояснювальна записка випускної кваліфікаційної роботи викладена на 97 сторінок друкованого тексту, містить 20 рисунків, 5 таблиць та 3 додатки. Список використаної літератури налічує 17 найменувань і розміщений на 2 сторінках.

Мета кваліфікаційної роботи - проєктування, програмно-технічна реалізація та дослідження ефективності територіально розподіленої локальної IoT-мережі для автоматизації офісної інфраструктури підприємства на основі конвергентного комутаційного ядра MikroTik CHR, із забезпеченням асинхронного збору телеметрії за протоколом MQTT та побудовою автономного контуру високопріоритетного пуш-інформування про критичні інциденти.

У роботі:

- обґрунтовано доцільність застосування технології віртуалізації мережевих функцій та утилізації MikroTik Cloud Hosted Router як інтелектуального крайового шлюзу;

- математично розраховано схему логічної мікросегментації адресного простору CIDR із використанням масок довжиною /25 та розгорнуто 6 ізольованих на каналному рівні віртуальних мереж стандарту IEEE 802.1Q для розділення трафіку користувачів та датчиків;

- розгорнуто та скоординовано збалансований прикладний стек із 7 мікросервісів безпосередньо всередині інтегрованого контейнерного рушія RouterOS із персистентним монтуванням даних на розділ ext4;

- розроблено подієво-орієнтоване крайове програмне забезпечення на мові Python, що реалізує асинхронний аудит ліній зв'язку, моделювання інженерних метрик у JSON-форматі та пряму HTTP-генерацію тривожних push-алертів;

					KKIK5.KP.5 - 123 - 06 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	“Проект локальної IoT-мережі на основі віртуального маршрутизатора MikroTik CHR”	Літ.	Арк.	Аркушів	
Розроб.		Зубрицький П.А.		17.06					
Керівник		Матвеев К.І.		17.06			5	97	
Реценз.		Шелуха О.О.		17.06		ФІКТ, Гр. КІ-22-1			
Н. контр.		Матвеев К.І.		17.06					
Затверд.		Єфіменко А.А.		17.06					

- впроваджено комплексну політику безпеки периметра Zero Trust за допомогою налаштування ланцюжків Input та Forward міжмережевого екрану IP Firewall Filter;

- експериментально протестовано роботу комплексу під керуванням мобільного інженерного дашборду й доведено, що час доставки сигналу тривоги на смартфон оператора становить менше 0.45 секунди.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ РЕЧЕЙ; МІКРОСЕГМЕНТАЦІЯ; МАРШРУТИЗАТОР МІКРОТІК CHR; КОНТЕЙНЕРИЗАЦІЯ; ПРОТОКОЛ MQTT; КІБЕРБЕЗПЕКА; ПУШ-СПОВІЩЕННЯ.

		Зубрицький П.А.			6ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ANNOTATION

The bachelor's thesis is performed on the topic: Project of a local IoT network based on the virtual router MikroTik CHR. The explanatory note of the bachelor's thesis consists of 97 pages of printed text, contains 20 figures, 5 tables, and 3 appendices. The bibliography includes 17 references and is placed on 2 pages.

The purpose of the thesis is the design, software and hardware implementation, and efficiency evaluation of a geographically distributed local IoT network for enterprise office infrastructure automation based on a convergent switching core MikroTik CHR, ensuring asynchronous telemetry collection via MQTT protocol and building an autonomous high-priority push notification loop for critical incidents.

In the thesis:

- the expediency of applying Network Functions Virtualization technology and utilizing MikroTik Cloud Hosted Router as an intelligent Edge Gateway has been substantiated;

- a logical microsegmentation scheme of CIDR address space using /25 masks was mathematically calculated, and 6 virtual local area networks of IEEE 802.1Q standard were deployed to isolate user and sensor traffic at the data link layer;

- a balanced application stack of 7 microservices was deployed and coordinated directly inside the integrated RouterOS container engine with persistent data mounting on the ext4 partition;

- event-driven edge software was developed in Python, implementing asynchronous link auditing, engineering metrics modeling in JSON format, and direct HTTP generation of emergency push alerts;

- a comprehensive Zero Trust perimeter security policy was implemented by configuring Input and Forward chains of the IP Firewall Filter;

- the operation of the complex under the control of a mobile engineering dashboard was experimentally tested, proving that the alarm delivery time to the operator's smartphone is less than 0.45 seconds.

KEYWORDS: INTERNET OF THINGS; MICROSEGMENTATION; MIKROTIK CHR ROUTER; CONTAINERIZATION; MQTT PROTOCOL; CYBERSECURITY; PUSH NOTIFICATIONS.

		Зубрицький П.А.			7KKIKБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	10
ВСТУП.....	11
РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ТА ОБҐРУНТУВАННЯ НАПРЯМІВ РЕАЛІЗАЦІЇ ІoT-МЕРЕЖ	14
1.1 Архітектурні моделі побудови інфраструктури ІoT та крайові обчислення .	14
1.2. Класифікація та функціональна роль засобів контейнеризації на крайових пристроях	18
1.3. Порівняльний аналіз брокерів повідомлень для розподілених мереж автоматизації	23
1.4. Постановка завдання на проєктування	26
ВИСНОВКИ ДО РОЗДІЛУ 1	29
РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА РОЗРАХУНОК СХЕМИ ОРГАНІЗАЦІЇ ЛОКАЛЬНОЇ ІoT-МЕРЕЖІ	30
2.1. Концепція побудови крайової інфраструктури на базі операційної системи RouterOS	30
2.2. Розрахунок адресного простору та проєктування віртуальних локальних мереж (VLAN)	33
2.3. Математичний розрахунок адресного простору та проєктування віртуальних локальних мереж (VLAN)	36
2.4. Структурно-логічна модель інформаційного обміну та структура телеметричних даних	40
2.5. Алгоритми функціонування крайового програмного забезпечення шлюзів	49
2.6. Проєктування підсистеми міжмережевого екранування периметра мережі	52
ВИСНОВКИ ДО РОЗДІЛУ 2	55
РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ, НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ КОМПЛЕКСУ ІoT-МЕРЕЖІ ТА АНАЛІЗ РЕЗУЛЬТАТІВ	56
3.1. Реалізація підсистеми міжмережевого екранування та захисту периметра в RouterOS	56

		Зубрицький П.А.			8KKIKB.KP.B - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

3.2. Конфігурування та верифікація середовища контейнеризації в RouterOS ...	59
3.3. Налаштування комутаційного обладнання та крайових хостів інфраструктури в GNS3	69
ВИСНОВКИ ДО РОЗДІЛУ 3	75
ВИСНОВКИ	77
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ	79
ДОДАТКИ	81

		Зубрицький П.А.			9KKIKБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AES - Advanced Encryption Standard
 API - Application Programming Interface
 CIDR - Classless Inter-Domain Routing
 CPU - Central Processing Unit
 DHCP - Dynamic Host Configuration Protocol
 DoS - Denial of Service
 GNS3 - Graphical Network Simulator-3
 HDD - Hard Disk Drive
 HTTP - Hypertext Transfer Protocol
 IEEE 802.1Q - Institute of Electrical and Electronics Engineers 802.1Q
 IoT - Internet of Things
 IP - Internet Protocol
 JSON - JavaScript Object Notation
 L2 - Layer 2
 L3 - Layer 3
 MAC - Media Access Control
 MQTT - Message Queuing Telemetry Transport
 NFV - Network Functions Virtualization
 OSI - Open Systems Interconnection
 RAM - Random Access Memory
 RFID - Radio-Frequency Identification
 TCP - Transmission Control Protocol
 URL - Uniform Resource Locator
 veth - Virtual Ethernet
 VLAN - Virtual Local Area Network
 VPCS - Virtual PC Simulator
 WAN - Wide Area Network
 ОС - операційна система
 ПЗ - програмне забезпечення

		Зубрицький П.А.			10 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Стрімкий розвиток концепції Інтернету речей (Internet of Things, IoT) та масове впровадження інтелектуальних пристроїв у корпоративний та промисловий сектори висувають нові жорсткі вимоги до архітектури сучасних локальних мереж. Сучасний офіс або підприємство вже не обмежується обслуговуванням стандартних персональних комп'ютерів та серверів; до мережевої інфраструктури інтегруються десятки й сотні різномірних кінцевих пристроїв: цифрові IP-камери відеоспостереження, датчики мікроклімату, сенсори вуглекислого газу, контролери систем керування та контролю доступу (СКУД), розумні лічильники електроенергії та виконавчі механізми.

Специфіка IoT-трафіку полягає у його високій частоті, гетерогенності, асиметричності та критичності до затримок або втрат пакетів, особливо коли мова йде про системи фізичної безпеки та протипожежного захисту. Традиційні підходи до побудови локальних мереж часто виявляються неефективними, оскільки не забезпечують належного рівня ізоляції технологічного трафіку від загального корпоративного, що створює серйозні вектори для кібератак та призводить до нераціонального використання смуги пропускання.

У цих умовах надзвичайно актуальною стає концепція віртуалізації мережевих функцій (Network Functions Virtualization, NFV), яка дозволяє відмовитися від дорогого, спеціалізованого апаратного забезпечення на користь програмно-конфігурованих рішень, розгорнутих у хмарних середовищах або на базі локальних серверів віртуалізації. Одним із найбільш потужних та гнучких інструментів у цьому сегменті є віртуальний маршрутизатор MikroTik Cloud Core Router (CHR). Він переносить усі можливості операційної системи RouterOS у віртуальне середовище, забезпечуючи повний спектр функцій маршрутизації, комутації, фільтрації трафіку за допомогою вбудованого міжмережевого екрану, а також унікальну можливість розгортання ізольованих контейнерів (Container) безпосередньо всередині ядра маршрутизатора. [3]

		Зубрицький П.А.			1 ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Використання MikroTik CHR як центрального вузла локальної IoT-мережі дозволяє побудувати масштабовану, гнучку та економічно вигідну інфраструктуру. Завдяки підтримці контейнеризації на борту віртуального роутера, зникає потреба в окремому фізичному сервері для розміщення сервісів збору даних. Це дозволяє розгорнути легковагий брокер повідомлень Eclipse Mosquitto MQTT безпосередньо у внутрішньому логічному сегменті маршрутизатора, мінімізуючи мережеві затримки та спрощуючи загальну топологію.

Мета дослідження: полягає у проєктуванні, програмно-технічній реалізації та дослідженні ефективності локальної IoT-мережі для розподіленої офісної інфраструктури на основі віртуального маршрутизатора MikroTik CHR, із забезпеченням централізованого збору метрик за протоколом MQTT та побудовою системи миттєвого інтелектуального сповіщення про критичні події.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Провести детальний аналітичний огляд архітектурних моделей побудови IoT-мереж та провести порівняльний аналіз протоколів передачі даних прикладного рівня.
- Обґрунтувати доцільність застосування технології NFV та використання віртуального маршрутизатора MikroTik CHR як центрального комутаційного вузла інфраструктури.
- Розробити комплексну структурно-функціональну схему локальної IoT-мережі, що імітує роботу трьох територіально або логічно розділених офісних приміщень із різним набором сенсорів.
- Здійснити математичний розрахунок та практичне налаштування схеми IP-адресації, віртуальних локальних мереж (VLAN) та правил маршрутизації трафіку для забезпечення ізоляції мережевих сегментів.

		Зубрицький П.А.			12ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

- Розробити програмне забезпечення для кінцевих IoT-шлюзів на базі мови програмування Python для збору даних з датчиків, симуляції їхньої роботи та взаємодії з MQTT-брокером.
- Інтегрувати систему збору даних із сучасним сервісом push-сповіщень ntfy для миттєвого інформування персоналу про загрози безпеці (рух, пожежа).
- Провести комплексне тестування розробленої системи у штатному та аварійному режимах роботи, оцінити стабільність функціонування розгорнутих контейнерів.

Об'єкт дослідження - процеси передачі, обробки та фільтрації телеметричних даних у локальних комп'ютерних мережах Інтернету речей.

Предмет дослідження - архітектурні рішення, програмні засоби та мережеві протоколи організації локальної IoT-мережі на базі віртуалізованої операційної системи MikroTik RouterOS (CHR).

		Зубрицький П.А.			13ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД ТА ОБҐРУНТУВАННЯ НАПРЯМІВ РЕАЛІЗАЦІЇ ІoT-МЕРЕЖ

1.1 Архітектурні моделі побудови інфраструктури ІoT та крайові обчислення

Стрімкий розвиток концепції Інтернету речей (ІoT) на сучасному етапі супроводжується експоненційним збільшенням кількості кінцевих пристроїв - сенсорів, виконавчих механізмів та інтелектуальних датчиків. Початкові етапи впровадження ІoT-систем базувалися виключно на хмарно-орієнтованій моделі (Cloud-centric architecture). У такій топології всі первинні дані, зібрані сенсорами, безпосередньо трансливалися через глобальну мережу Інтернет до централізованих дата-центрів (наприклад, AWS, Google Cloud, Microsoft Azure) для подальшого аналізу, обробки та зберігання.

Проте, із масштабуванням систем ІoT класична хмарна модель виявила низку критичних недоліків, які обмежують її застосування в сучасних інженерних рішеннях:

- Затримки передачі даних (Network Latency): Для систем, що працюють у режимі реального часу (наприклад, промислова автоматика чи системи безпеки), затримка в кілька сотень мілісекунд, викликана транзитом даних до віддаленого сервера, є недопустимою.
- Перевантаження каналів зв'язку: Безперервне надсилення сирової телеметрії від тисяч датчиків створює колосальне навантаження на мережеву інфраструктуру та вимагає високої пропускної здатності каналів WAN.
- Фінансова вартість: Оренда хмарних ресурсів, оплата за обсяг переданого трафіку та збереження великих масивів даних (Big Data) стрімко збільшують операційні витрати (ОРЕХ).
- Конфіденційність та безпека: Передача персональних даних або комерційної таємниці через відкриті канали зв'язку створює додаткові вектори для кібератак.

		Зубрицький П.А.			14ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Як еволюційна відповідь на ці виклики виникла концепція крайових обчислень (Edge Computing). Вона передбачає децентралізацію обчислювальних потужностей та перенесення функцій обробки, фільтрації й агрегації інформації безпосередньо до контуру локальної мережі - ближче до джерела виникнення даних. [13]

У контексті даного дипломного проєкту роль такого інтелектуального крайового вузла виконує віртуальний маршрутизатор MikroTik Cloud Hosted Router (CHR). Завдяки інтеграції технології контейнеризації на базі архітектури RouterOS, маршрутизатор трансформується з класичного пристрою комутації пакетопотоків у повноцінну обчислювальну платформу Micro Cloud. Це дозволяє здійснювати первинну обробку IoT-трафіку (наприклад, через розгорнутий контейнер MQTT-брокера app-mosquitto) локально, мінімізуючи звернення до зовнішніх мереж.

Для обґрунтування вибору технологій прикладного рівня в рамках локальної інфраструктури нижче наведено детальний аналіз основних протоколів, що використовуються в сучасному IoT-сегменті: MQTT, HTTP (REST), CoAP та AMQP.

Протокол HTTP (Hypertext Transfer Protocol / REST API)

HTTP є класичним протоколом, побудованим на моделі «клієнт-сервер» (Request/Response). Кожен запит від IoT-пристрою вимагає встановлення нового TCP-з'єднання (або підтримання постійного через Keep-Alive) та передачі великого об'єму службових заголовків (Headers), які часто перевищують розмір самих корисних даних у кілька разів.

Для локальних мереж з великою кількістю датчиків використання HTTP є нерациональним через надлишковий трафік (overhead), відсутність вбудованого механізму зворотного зв'язку в реальному часі (сервер не може самостійно надіслати дані клієнту без його попереднього запиту) та високі вимоги до обчислювальної потужності кінцевих мікроконтролерів.

		Зубрицький П.А.			15ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Протокол CoAP (Constrained Application Protocol)

CoAP розроблений спеціально для пристроїв з обмеженими ресурсами (constrained nodes). На відміну від HTTP, він працює поверх протоколу транспортного рівня UDP, що мінімізує накладні витрати на встановлення сесії та знижує затримки. CoAP копіює модель REST (методи GET, POST, PUT, DELETE), але використовує двійкове кодування заголовків.

Основним недоліком CoAP у рамках складної локальної інфраструктури є використання UDP, яке не гарантує доставку пакетів «із коробки», а також складність налаштування правил фільтрації та трансляції адрес (NAT) на міжмережевих екранах для нестандартних UDP-потоків.

Протокол AMQP (Advanced Message Queuing Protocol)

AMQP - це потужний протокол, орієнтований на повідомлення, що працює за моделі «публікація-підписка» та забезпечує суворі гарантії доставки, складну маршрутизацію потоків даних та високий рівень безпеки.

Він широко застосовується у банківських та великих корпоративних фінансових системах (наприклад, через брокер RabbitMQ). Проте для локальних мереж Інтернету речей AMQP є занадто «важким» та складним у реалізації. Його заголовки та логіка обробки черг вимагають значних обчислювальних ресурсів та оперативної пам'яті, що робить його непридатним для застосування на слабких embedded-системах або бюджетних шлюзах автоматизації.

Протокол MQTT (Message Queuing Telemetry Transport)

MQTT є де-факто стандартом для сучасних IoT-систем завдяки використанню архітектури «публікація-підписка» (Publish/Subscribe) поверх надійного транспортного протоколу TCP. Центральним елементом мережі виступає MQTT-брокер, який приймає повідомлення від видавців (датчиків) і перенаправляє їх підписникам (серверам моніторингу, базам даних, клієнтським додаткам) на основі структурованих тем (Topics).

		Зубрицький П.А.			16 ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Основними перевагами протоколу MQTT для розроблюваного проєкту є:

- Мінімальний розмір заголовка: Фіксований заголовок MQTT становить всього 2 байти, що дозволяє передавати корисне навантаження (Payload) у форматі JSON з мінімальним задіянням пропускну здатності локального каналу зв'язку.
- Ефективне керування трафіком: Датчикам не потрібно знати архітектуру кінцевих споживачів інформації. Вони просто відправляють дані брокеру, що повністю ізолює логіку роботи пристроїв від аналітичних платформ.
- Рівні якості обслуговування (Quality of Service, QoS): MQTT підтримує три рівні доставки пакетів: QoS 0 (доставка максимум один раз без підтвердження), QoS 1 (доставка мінімум один раз із обов'язковим підтвердженням від брокера, що гарантує збереження критичних алертів) та QoS 2 (доставка строго один раз за допомогою чотиристороннього рукошлякування).
- Механізм Last Will and Testament (LWT): Дозволяє брокеру автоматично сповістити всю мережу, якщо якийсь із IoT-шлюзів раптово втратив зв'язок (впав по живленню або через обрив кабелю).

Зважаючи на проведені дослідження, для проєктування локальної IoT-мережі було обрано саме протокол MQTT, як такий, що забезпечує ідеальний баланс між швидкістю передачі даних, надійністю доставки критичних повідомлень та простотою програмної реалізації на мові Python. [5]

		Зубрицький П.А.			17ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Класифікація та функціональна роль засобів контейнеризації на крайових пристроях

Сучасна архітектура крайових обчислень в IoT-мережах має чітку ієрархічну структуру. Всю сукупність апаратних та програмних компонентів розробленої системи можна класифікувати на два основні взаємопов'язані рівні: кінцеві пристрої та інтелектуальні крайові шлюзи.

Кінцеві пристрої (End Devices / IoT Nodes) формують базовий, або фізичний рівень архітектури інфраструктури. До цього класу обладнання відносяться безпосередньо інтегровані в офісні приміщення фізичні датчики (визначення розбиття скла, сповіщувачі руху, сенсори концентрації вуглекислого газу) та виконавчі механізми (контролери реле, приводи шлюзів, зчитувачі систем доступу). Зазвичай ці пристрої проектуються на базі енергоефективних мікроконтролерів (класу ESP32, ESP8266 або STM32) і мають суворі обмеження щодо обчислювальної потужності та об'єму оперативної пам'яті. Вони не спроможні самостійно виконувати складні операційні або аналітичні завдання, тому їх функціонал обмежений збором первинних сигналів і їх трансляцією за допомогою легковагових протоколів до вищого рівня інфраструктури.

Крайові пристрої (Edge Devices / Gateways) представляють собою проміжний, інтелектуальний шар інфраструктури, який виступає в ролі координатора та захисного бар'єра локальної мережі. Крайовий шлюз забезпечує централізований збір потоків сирової телеметрії від кінцевих вузлів, здійснює їх первинний синтаксичний аналіз (parsing), фільтрує інформаційні аномалії, забезпечує локальне збереження критичних логів та гарантує мережеву безпеку.

Класичний інженерний підхід вимагає фізичного або логічного розділення мережевих функцій (які виконує маршрутизатор) та обчислювальних функцій (які покладаються на окремі мікрокомп'ютери класу Raspberry Pi або локальні сервери). Науково-практичною новизною та ключовою особливістю даного проекту є конвергенція (об'єднання) цих ролей на базі одного єдиного пристрою - віртуального маршрутизатора MikroTik Cloud Hosted Router (CHR) [1].

		Зубрицький П.А.			18 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

Програмне середовище маршрутизатора на базі сучасної операційної системи RouterOS дозволяє повністю ізолювати сервіси всередині легковагових контейнерів. Це суттєво знижує апаратну надлишковість системи, спрощує адміністрування, мінімізує точки відмови та підвищує загальну відмовостійкість інфраструктури. Аналізуючи поточний стан розгортання системи, можна стверджувати, що MikroTik CHR успішно виконує роль високотехнологічного Edge Gateway, забезпечуючи стабільне функціонування повного стеку мікросервісів (Рис. 1.2.1)

Container						
Container Mounts Envs Log						
New Remove						
		Name	Root Dir	Entrypoint	Cmd	
☐	R	app-caddy	/sata1/apps/cad...			
☐	R	app-gitea	/sata1/apps/gite...			
☐	R	app-grafana	/sata1/apps/gra...			
☐	R	app-mosquitto	/sata1/apps/mo...			
☐	R	app-nodered	/sata1/apps/nod...			
☐	R	app-ntfy	/sata1/apps/ntfy...		serve	
☐	R	app-vaultwarden	/sata1/apps/vau...			

Рис. 1.1. Стан запущених IoT-контейнерів у середовищі RouterOS

Нижче наведено детальне функціональне обґрунтування системної ролі, переваг та технічної необхідності кожного з семи розгорнутих у контейнерах сервісів у рамках архітектури розроблюваної IoT-мережі:

1. Контейнер app-mosquitto (Образ eclipse-mosquitto:latest):

Виступає центральним комунікаційним ядром та серцем транспортного протоколу прикладного рівня MQTT. Його вибір обґрунтований екстремально низькими вимогами до ресурсів процесора та оперативної пам'яті хост-машини при високій пропускній здатності. Сервіс забезпечує асинхронний прийом телеметричних пакетів від розподілених IoT-шлюзів у форматі JSON. Завдяки підтримці рівнів якості обслуговування (QoS), брокер гарантує доставку критично важливих сповіщень навіть у випадках нестабільного мережевого з'єднання, повністю ізолюючи логіку роботи датчиків від аналітичних платформ. [6]

		Зубрицький П.А.			19ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

2. Контейнер app-nodered (Образ nodered/node-red:latest):

Виконує роль інтелектуального диспетчера та оркестратора потоків даних реального часу. Платформа Node-RED підписана на відповідні топіки MQTT-брокера Mosquitto. Вона перехоплює вхідні повідомлення, здійснює їх динамічний аналіз, порівнює числові показники датчиків із заданими безпечними пороговими значеннями і приймає рішення про подальше керування системою - наприклад, генерацію HTTP-запитів до підсистеми сповіщень або передачу даних в аналітичні бази. [7]

3. Контейнер app-ntfy (Образ binwiederhier/ntfy:latest):

Є базовим елементом підсистеми миттєвого інформування та push-оповіщень. Його критична перевага полягає в повній автономності: для доставки повідомлень на смартфони адміністраторів не потрібно використовувати комерційні закриті хмарні платформи. Сервіс ntfy приймає вхідні HTTP POST-запити від Node-RED або шлюзів і миттєво транслює push-повідомлення у мобільні додатки відповідальних осіб із чітким кодуванням пріоритетів та системних емодзі-тегів. [9]

4. Контейнер app-grafana (Образ grafana/grafana:latest):

Представляє собою шар візуалізації та аналітичного моніторингу системи. Контейнер агрегує інформаційні потоки та трансформує сухі числові масиви телеметрії у високотехнологічні інтерактивні графіки (Dashboards). Завдяки Grafana оператори та адміністратори мережі мають можливість у реальному часі відслідковувати рівень безпеки в приміщеннях, будувати тренди енергоспоживання, аналізувати графіки температурних коливань та оперативно реагувати на системні збої. [8]

5. Контейнер app-caddy (Образ caddy:latest):

Виконує роль високопродуктивного зворотного проксі-сервера (Reverse Proxy) та єдиної точки входу (Ingress) для веб-інтерфейсів моніторингу. Caddy забезпечує безпечний доступ адміністратора до панелей керування Node-RED та Grafana. Головна перевага цього сервісу - вбудована автоматизація отримання та оновлення криптографічних TLS/SSL сертифікатів, а також простота конфігурації.

		Зубрицький П.А.			20ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

Це дозволяє надійно захистити сесії керування від перехоплення трафіку всередині корпоративного контуру. [10]

6. Контейнер **app-gitea** (Образ **gitea/gitea:latest**):

Інтегрований в інфраструктуру як локальний, легковажний Git-репозиторій для координаційного шару безпеки та збереження конфігурацій. Усі розроблені програмні скрипти для автоматизації офісів, конфігураційні файли Mosquitto, Caddyfile та схеми потоків Node-RED зберігаються локально із підтримкою контролю версій. Це виключає ризик втрати програмного коду при збоях на хост-машині та дозволяє реалізувати надійний контроль за станом інфраструктури. [12]

7. Контейнер **app-vaultwarden** (Образ **vaultwarden/server:latest**):

Забезпечує високий рівень безпеки конфіденційних даних інфраструктури, виступаючи локальним зашифрованим сховищем секретів. IoT-мережа оперує великою кількістю паролів до MQTT-брокера, токенів доступу до API ntfy, SSH-ключів маршрутизатора та облікових записів баз даних. Контейнер Vaultwarden (оптимізована на мові Rust версія Bitwarden) надійно шифрує ці дані за алгоритмом AES-256 на стороні клієнта, гарантуючи захист секретів інфраструктури від компрометації. [11]

Розгортання такого масштабного, але водночас легковажного та збалансованого стеку мікросервісів безпосередньо всередині ядра віртуального маршрутизатора MikroTik CHR дозволяє повністю ліквідувати потребу в купівлі та обслуговуванні окремих фізичних серверів, забезпечуючи максимальну щільність інтеграції та високу кіберстійкість системи.

Для забезпечення стабільної та безпечної роботи всього мікросерверного комплексу, кожному контейнеру всередині операційної системи RouterOS виділяється окремий віртуальний мережевий інтерфейс типу veth (Virtual Ethernet). Конфігурація мережевих параметрів, виділених адрес та режимів доступу до файлової системи для кожного розгорнутого образу деталізована у таблиці 1.1.

		Зубрицький П.А.			2ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.1 – Мережеві параметри та режими монтування дискових томів контейнеризованих сервісів

Назва сервісу (Контейнер)	Внутрішній IP- інтерфейс у MikroTik	Виділений порт (Протокол)	Режим монтування томів (Mounts Mode)
app-caddy	172.18.0.20	80, 443 (TCP / HTTP, HTTPS)	Read-Write (rw)
app-gitea	172.18.0.5	3007 (TCP / Web), 22 (TCP / SSH)	Read-Write (rw)
app-grafana	172.18.0.9	3000 (TCP / HTTP UI)	Read-Write (rw)
app-mosquitto	172.18.0.7	1883 (TCP / MQTT) 9001 (TCP / WebSockets)	Read-Write (rw)
app-nodered	172.18.0.13	1880 (TCP / HTTP UI)	Read-Write (rw)
app-ntfy	172.18.0.6	3005 (TCP / HTTP API)	Read-Write (rw)
app-vaultwarden	172.18.0.16	3006 (TCP / Web UI)	Read-Write (rw)

1.3. Порівняльний аналіз брокерів повідомлень для розподілених мереж автоматизації

У системах Інтернету речей, що базуються на архітектурі «публікація-підписка», брокер повідомлень є центральним елементом інфраструктури, який визначає загальну відмовостійкість, швидкість комутації пакетів та рівень утилізації апаратних ресурсів крайового шлюзу. При проектуванні локальної мережі на основі віртуального маршрутизатора MikroTik CHR виникає необхідність вибору оптимального програмного рішення, яке здатне ефективно функціонувати в умовах обмежених обчислювальних ресурсів вбудованого контейнерного рушія.

Для обґрунтування інженерного вибору в межах дослідження було проведено детальний порівняльний аналіз трьох найбільш поширених у промисловому та корпоративному секторах брокерів повідомлень: Eclipse Mosquitto, RabbitMQ та Apache Kafka.

Брокер повідомлень Apache Kafka

Apache Kafka представляє собою розподілену платформу потокової передачі даних та обробки подій високого рівня. Вона розроблена для оперування колосальними масивами інформації (Big Data) у масштабних кластерних архітектурах. Kafka забезпечує персистентне збереження всіх повідомлень на диску, підтримує реплікацію потоків та гарантує високу стійкість до відмов.

Проте, для локальних IoT-мереж крайового рівня Apache Kafka є абсолютно надлишковим та непридатним рішенням. Платформа функціонує поверх віртуальної машини Java (JVM) та вимагає наявності додаткового координаційного сервісу (ZooKeeper або KRaft), що призводить до споживання кількох гігабайт оперативної пам'яті в режимі простою. Розгортання такого стеку всередині контейнера на MikroTik CHR заблокує обчислювальні ресурси процесора хост-машини та унеможливить виконання маршрутизатором його першочергових мережевих функцій.

		Зубрицький П.А.			23ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Брокер повідомлень RabbitMQ

RabbitMQ є класичним, надзвичайно потужним брокером повідомлень, який повністю підтримує стандарт AMQP (Advanced Message Queuing Protocol), а також, за допомогою спеціальних плагінів, здатний взаємодіяти за протоколами MQTT та CoAP. Він орієнтований на складну корпоративну маршрутизацію трафіку, підтримує гнучке налаштування черг, обмінників (Exchanges) та суворі критерії безпеки транзакцій.

Незважаючи на високу функціональність, інтеграція RabbitMQ в архітектуру даного проєкту має низку суттєвих обмежень. Написаний на мові Erlang, цей брокер вимагає значних системних ресурсів (мінімум 100–150 МБ ОЗП для ініціалізації порожніх черг). Логіка обробки повідомлень у RabbitMQ орієнтована на складні конвеєри даних, що створює невиправдане додаткове навантаження на процесор (CPU) віртуального роутера при комутації легковагових JSON-рядків від офісних датчиків.

Брокер повідомлень Eclipse Mosquitto

Eclipse Mosquitto - це легковажний, високоефективний брокер повідомлень з відкритим вихідним кодом, який повністю реалізує специфікації протоколу MQTT версій 5.0, 3.1.1 та 3.1. Він розроблений спеціально для застосування в концепціях Інтернету речей, embedded-системах та на крайових шлюзах автоматизації.

На відміну від конкурентних рішень, Mosquitto написаний на чистій мові програмування C, має монопотоківу подієво-орієнтовану архітектуру (Event-driven loop) та оптимізований для обробки тисяч одночасних підключень з мінімальними накладними витратами.

Основними архітектурними перевагами Eclipse Mosquitto, що зумовили його вибір для реалізації в даній кваліфікаційній роботі, є:

- Екстремально низьке споживання ресурсів: У робочому режимі контейнер app-mosquitto утилізує не більше 10–15 МБ оперативної пам'яті та демонструє практично нульове навантаження на обчислювальні ядра MikroTik CHR.

		Зубрицький П.А.			24 ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

- Рідна підтримка ієрархії топіків: Дозволяє будувати прості та зрозумілі дерева адрес для розподілених підмереж (наприклад, топіки для Офісу 1 та Офісу 2), забезпечуючи чітке логічне розділення даних без налаштування складних обмінників.
- Вбудовані механізми авторизації та контролю доступу: Підтримує організацію доступу за списками ACL (Access Control List) та шифрування файлів паролів, що відповідає вимогам безпеки технологічного периметра підмереж.
- Ідеальна сумісність із контейнеризацією RouterOS: Стабільний образ Docker для Mosquitto не вимагає специфічних системних викликів або додаткових бібліотек ядра, що гарантує його безперебійний запуск із прапорцем running (R) у WinBox.

Таким чином, у результаті проведеного порівняльного аналізу брокер Eclipse Mosquitto визначено єдиним раціональним та інженерно обґрунтованим рішенням для побудови локальної IoT-інфраструктури на базі віртуального маршрутизатора MikroTik CHR.

		Зубрицький П.А.			25ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

1.4. Постановка завдання на проєктування

На основі детального аналітичного огляду сучасних архітектурних концепцій побудови інфраструктур Інтернету речей, порівняння прикладних протоколів та дослідження можливостей віртуалізації мережевих функцій формулюється чітке технічне завдання на розробку комплексної системи. Метою випускної кваліфікаційної роботи є проєктування, програмно-технічна реалізація та дослідження ефективності локальної IoT-мережі для розподіленої офісної інфраструктури на основі віртуального маршрутизатора MikroTik CHR з побудовою системи моніторингу та миттєвого інтелектуального оповіщення про критичні події.

Для забезпечення високої надійності, відмовостійкості та інформаційної безпеки до проєктуємої системи висувається низка жорстких інженерно-технічних вимог, які класифікуються за відповідними категоріями:

1. Вимоги до архітектури мережі та комутаційної інфраструктури

- Логічна сегментація трафіку: Мережева топологія повинна імітувати роботу трьох логічно або територіально відокремлених офісних приміщень підприємства.
- Ізоляція технологічних зон: Для запобігання несанкціонованому міжмережевому обміну та мінімізації векторів кібератак кожен офіс має бути виділений в окрему віртуальну локальну мережу (VLAN) стандарту IEEE 802.1Q. [2]
- Центральний комутаційний вузол: Усі функції маршрутизації, трансляції мережевих адрес (NAT), призначення динамічних IP-адрес (DHCP-сервер) та фільтрації пакетів покладаються на віртуальний маршрутизатор MikroTik CHR, розгорнутий у середовищі віртуалізації.
- Автономність функціонування: Взаємодія між крайовими вузлами та серверами обробки має здійснюватися суворо всередині локального контуру автоматизованого об'єкта, що забезпечує здатність мережі функціонувати за повної відсутності інтернет-з'єднання.

		Зубрицький П.А.			26КІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

2. Вимоги до підсистеми збору та обробки телеметрії

- Протокол взаємодії прикладного рівня: Як базовий транспортний механізм обміну даними має використовуватися протокол MQTT, що функціонує за моделлю «публікація-підписка».
- Сервер комутації повідомлень: Взаємодія між видавцями та підписниками здійснюється через легковажний брокер Eclipse Mosquitto, розгорнутий у вигляді ізольованого контейнера безпосередньо всередині операційної системи RouterOS.
- Програмні крайові шлюзи: Кінцеві IoT-вузли офісів реалізуються програмним способом на мові Python. Вони повинні в безперервному циклі опитувати стан підключених віртуальних пристроїв за допомогою утиліти ping.
- Стандартизація структури даних: Передача телеметричних показників (температури, рівня вуглекислого газу, енергоспоживання та статусів тривоги) має виконуватися у вигляді текстових рядків, серіалізованих у формат JSON, для забезпечення сумісності з аналітичними платформами.

3. Вимоги до підсистеми візуалізації та інтелектуального оповіщення

- Моніторинг у режимі реального часу: Система повинна включати графічний центр моніторингу (Dashboard) на базі платформи Grafana для наочного відображення поточних індикаторів та побудови часових рядів енергоспоживання й мікроклімату.
- Логічна оркестрація даних: Обробка вхідних MQTT-повідомлень, десеріалізація JSON-пакетів, порівняння числових значень із критичними порогами та координація дій при аваріях покладається на контейнер Node-RED.
- Миттєве інформування персоналу: При фіксації критичних подій, таких як загроза пожежі (перевищення рівня CO₂) або несанкціоноване проникнення (фіксація руху камерою), система зобов'язана миттєво генерувати та надсилати push-сповіщення на мобільні пристрої адміністраторів через автономний локальний сервер ntfy.

		Зубрицький П.А.			27ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

4. Вимоги до безпеки та збереження секретів інфраструктури

- Захист адміністративного контуру: Усі розгорнуті веб-інтерфейси керування, локальні репозиторії програмного коду Gitea та системи менеджменту паролів Vaultwarden мають бути приховані за зворотним проксі-сервером з налаштованим шифруванням TLS.
- Криптографічний захист паролів: Усі токени доступу, облікові записи баз даних та ключі маршрутизатора повинні надійно шифруватися за алгоритмом AES-256 на стороні клієнта за допомогою локального сховища секретів.
- Персистентність сховища даних: Будь-які зміни конфігурацій, бази даних, логічні ланцюжки Node-RED та журнали сповіщень повинні монтуватися на зовнішній накопичувач маршрутизатора (sata1), що гарантує їх абсолютне збереження при раптовому знеструмленні або плановому перезавантаженні системи.

		Зубрицький П.А.			28 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі кваліфікаційної роботи проведено детальний аналітичний огляд сучасного стану розвитку технологій Інтернету речей (IoT), протоколів прикладного рівня та архітектурних рішень для побудови розподілених корпоративних мереж автоматизації.

За результатами проведеного аналізу патентних джерел та науково-технічної літератури сформовано такі висновки:

1. Визначено, що ключовою проблемою розгортання сучасних IoT-систем на базі мікроконтролерних пристроїв є їхня апаратна обмеженість, що унеможливорює використання складних локальних засобів криптографічного захисту та робить їх вразливими до мережових атак. Це обґрунтовує необхідність перенесення функцій безпеки та фільтрації трафіку на рівень мережевого комутаційного ядра периметра.
2. Проведено порівняльний аналіз транспортних протоколів прикладного рівня, за результатами якого для передачі первинної телеметрії обрано протокол MQTT завдяки його легковажності, підтримці архітектури «публікація-підписка» та мінімальному навантаженню на канали зв'язку. Для підсистеми миттєвого інформування обґрунтовано використання протоколу HTTP (запити POST), що забезпечує пряму інтеграцію з серверами пуш-сповіщень в обхід черг повідомлень.
3. Досліджено концепцію Edge Computing (крайових обчислень), яка дозволяє перенести стеки аналітичних мікросервісів безпосередньо в локальний контур об'єкта. Як базове програмно-апаратне рішення обрано технологію віртуалізації MikroTik Cloud Hosted Router (CHR) під керуванням операційної системи RouterOS v7, що дозволяє сумістити функції високопродуктивного маршрутизатора та ізолизованого середовища контейнеризованих Docker-додатків.
4. На основі проведеного аналізу сформульовано чіткі інженерні вимоги та технічне завдання на проєктування системи мікросегментації, ієрархії топіків та алгоритмів роботи крайового ПЗ, що стали основою для розрахунків у другому розділі роботи.

		Зубрицький П.А.			29ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

РОЗДІЛ 2. МОДЕЛЮВАННЯ ТА РОЗРАХУНОК СХЕМИ ОРГАНІЗАЦІЇ ЛОКАЛЬНОЇ ІoT-МЕРЕЖІ

2.1. Концепція побудови крайової інфраструктури на базі операційної системи RouterOS

Практична реалізація та проєктування локальної інфраструктури Інтернету речей вимагає створення стійкого, масштабованого та ізольованого обчислювального середовища на крайовому рівні (Edge Level). Основним інженерним завданням даного етапу є розгортання та логічне обґрунтування параметрів віртуального маршрутизатора MikroTik Cloud Hosted Router (CHR) з операційною системою RouterOS v7, який трансформується з класичного комутаційного пристрою в повноцінну обчислювальну edge-платформу.

Обґрунтування архітектури та ресурсного забезпечення стенду віртуалізації

Для створення та тестування інфраструктури розгорнуто комплексний лабораторний стенд на базі гіпервізора Oracle VirtualBox, інтегрованого з програмним середовищем емуляції складних мережевих топологій GNS3. [4] Надійність та відмовостійкість функціонування семи одночасно запущених мікросервісів безпосередньо залежить від обчислювального потенціалу хост-машини та коректного розподілу її залізобетонних ресурсів.

Фізична хост-машина лабораторного стенду володіє високими апаратними характеристиками:

- Загальний об'єм оперативної пам'яті становить 16 ГБ DDR4;
- Обчислювальне ядро побудоване на базі високопродуктивного процесора з 12 логічними ядрами.

З метою забезпечення балансу між продуктивністю контейнеризованого стеку сервісів та збереженням ресурсів хост-машини для безперебійної емуляції хостів офісів у GNS3, для віртуальної машини MikroTik CHR було виділено строго регламентовані апаратні ліміти. Маршрутизатору виділено 2 обчислювальних ядра процесора та 2 ГБ ОЗУ.

Такий розрахунок є інженерно обґрунтованим з огляду на наступні фактори:

		Зубрицький П.А.			30ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

- Архітектурна оптимізація ядра: Сама операційна система RouterOS є вкрай оптимізованою підсистемою Linux і в базовому режимі маршрутизації споживає мінімум ресурсів.
- Ресурсна ємність контейнерів: Більшість розгорнутих мікросервісів (наприклад, брокер app-mosquitto, сервіс сповіщень app-ntfy та менеджер секретів app-vaultwarden) відносяться до класу легковажного програмного забезпечення. Виділений об'єм у 2 ГБ ОЗУ створює солідний резервний буфер, що унеможливорює переповнення пам'яті при пікових навантаженнях, коли Node-RED здійснює високочастотне API-опитування логів, а Grafana рендерить складні аналітичні графіки.

Організація дискового простору та підсистеми збереження даних

Для забезпечення стабільної роботи контейнерів та персистентного (постійного) збереження інформації, базового віртуального диска операційної системи RouterOS є недостатньо. Концепція контейнеризації вимагає наявності окремого накопичувача для ізоляції файлових систем Docker-образів від завантажувальних секторів роутера.

У конфігурації віртуальної машини до віртуального контролера було підключено додатковий накопичувач sata1 фіксованого об'єму 10 ГБ. У середовищі RouterOS даний диск піддався низькорівневому форматуванню з розгортанням сучасної файлової системи ext4.

Вибір файлової системи ext4 зумовлений її високою стабільністю в середовищі Linux, підтримкою журналювання (Journaling), що мінімізує ризик пошкодження баз даних при раптовому вимкненні живлення, та повною сумісністю з механізмами монтування томів (Mounts) операційної системи RouterOS. Саме в межах цього накопичувача sata1 створюється ієрархічна структура директорій (наприклад, sata1/apps/), куди монтуються внутрішні папки контейнерів для ведення логів, збереження конфігурацій та утримання репозиторіїв коду.

		Зубрицький П.А.			3ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Проектування та логічний розподіл мережевих інтерфейсів

Архітектура мережевої взаємодії віртуального маршрутизатора MikroTik CHR розроблена за гібридною схемою, яка забезпечує одночасне керування зовнішнім периметром та комутацію внутрішніх технологічних зон офісів. Для цього в системі сконфігуровано кілька віртуальних мережевих адаптерів:

➤ Перший мережевий інтерфейс (міжмережевий бар'єр / WAN):

Цей інтерфейс виступає в ролі зовнішнього порту маршрутизатора, підключеного за схемою Мережевого моста (Bridged Adapter) або віртуального нату до фізичної мережі хост-машини. Його першочерговим інженерним призначенням є забезпечення зв'язку внутрішнього стеку контейнерів із зовнішнім світом. Через цей інтерфейс здійснюється первинне завантаження Docker-образів із глобальних репозиторіїв, а також забезпечується доступ адміністратора до веб-панелей керування та проходження push-алертів від сервера app-ntfy до мобільних пристроїв користувачів. Цей порт жорстко захищається вхідними правилами Firewall.

➤ Інші мережеві інтерфейси (Загальні / Базова інфраструктура GNS3):

Усі інші інтерфейси віртуального роутера конфігуруються як загальні комутаційні порти без безпосередньої прив'язки до зовнішніх фізичних адаптерів хост-машини. Вони інтегруються в робочий простір симулятора GNS3, виступаючи магістральними каналами (Trunk Links) для підключення емульованих технологічних підмереж Офісу 1, Офісу 2 та Офісу 3. На базі цих загальних інтерфейсів у подальшому розгортається логіка створення VLAN-інтерфейсів з ідентифікаторами VLAN 11-12, VLAN 21-22 та VLAN 31-, що забезпечує фізико-логічну конвергенцію віртуалізованого ядра MikroTik CHR із розподіленою інфраструктурою кінцевих пристроїв підприємства.

		Зубрицький П.А.			32ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

2.2. Розрахунок адресного простору та проєктування віртуальних локальних мереж (VLAN)

Проектування сучасної крайової інфраструктури (Edge Infrastructure) для Інтернету речей вимагає створення стійкого, безпечного та апаратно збалансованого обчислювального середовища безпосередньо в локальному контурі об'єкта. Основним інженерним завданням даного етапу є розробка логічної моделі розгортання та обґрунтування системних параметрів віртуального маршрутизатора MikroTik Cloud Hosted Router (CHR), який у межах проєкту трансформується з класичного пристрою комутації пакетопотоків у повноцінну обчислювальну edge-платформу, спроможну забезпечити ізольоване функціонування повного стеку мікросервісів.

Обґрунтування розподілу обчислювальних ресурсів та параметрів віртуалізації

Для забезпечення умов проєктування та можливості всебічного дослідження мережевих процесів розроблено концепцію лабораторного стенду на базі гіпервізора Oracle VirtualBox, логічно інтегрованого із середовищем емуляції складних мережевих топологій GNS3. Надійність паралельного функціонування комутаційного ядра та контейнеризованих IoT-сервісів безпосередньо залежить від потенціалу фізичної хост-машини та математично обґрунтованого виділення апаратних лімітів для віртуальної машини маршрутизатора.

Фізична обчислювальна платформа лабораторного стенду володіє такими характеристиками:

- Загальний об'єм оперативної пам'яті (RAM) становить 16 ГБ DDR4;
- Обчислювальне ядро побудоване на базі процесора з 12 логічними ядрами.

З метою оптимізації утилізації ресурсів та збереження необхідної ємності хост-машини для безперебійної емуляції розподілених пристроїв офісів у GNS3, для віртуального маршрутизатора MikroTik CHR спроєктовано такі фіксовані обмеження: 2 обчислювальних ядра процесора та 2 ГБ оперативної пам'яті (RAM).

Даний ресурсний розрахунок є інженерно обґрунтованим з огляду на два ключові фактори:

		Зубрицький П.А.			33ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

- Оптимізація системного ядра: Операційна система RouterOS v7 є високооптимізованою підсистемою на базі Linux і в режимі виконання базових функцій маршрутизації пакетопотоків утилізує мінімальну кількість ресурсів.
- Місткість серверного стеку: Більшість запланованих до розгортання контейнеризованих сервісів (брокер повідомлень app-mosquitto, локальний сервер пуш-інформування app-ntfy, менеджер секретів app-vaultwarden) відносяться до класу легковажного програмного забезпечення. Спроектований об'єм у 2 ГБ ОЗУ створює необхідний резервний буфер для згладжування пікових навантажень, які виникають при регулярному високочастотному зборі системних логів платформою Node-RED або динамічному рендерингу графіків часових рядів аналітичною системою Grafana.

Організація підсистеми персистентного збереження даних

Концепція функціонування мікросервісів у ядрі RouterOS вимагає повного логічного та фізичного розділення завантажувальних секторів маршрутизатора та файлових систем розгорнутих образів. Для забезпечення персистентного (постійного) збереження динамічної інформації спроектовано підключення до віртуального контролера ВМ додаткового віртуального накопичувача sata1 фіксованого об'єму 10 ГБ.

Для даного накопичувача обрано сучасну промислову файлову систему ext4. Вибір ext4 обґрунтований її високою стійкістю в Unix-середовищах, вбудованими механізмами журналювання, що мінімізує ризики пошкодження баз даних при раптовому знеструмленні об'єкта, та повною підтримкою технології монтування томів (Mounts) в операційній системі RouterOS. У межах простору sata1 проектується ієрархічна структура директорій, куди ізольовано монтуються внутрішні папки контейнерів для ведення журналів подій, персистентного утримання Git-репозиторіїв та конфігураційних файлів, що виключає втрату інформації при планових чи аварійних перезавантаженнях маршрутизатора.

		Зубрицький П.А.			34 ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

Топологічна схема та призначення мережевих інтерфейсів

Архітектура мережевої взаємодії віртуального ядра MikroTik CHR базується на гібридній моделі, яка поєднує функції захисту зовнішнього периметра та комутації внутрішніх технологічних зон підприємства. Для реалізації цієї моделі спроектовано розподіл віртуальних адаптерів за двома функціональними класами:

➤ Перший мережевий інтерфейс (ether1-wan):

Виступає в ролі зовнішнього порту (міжмережевого бар'єра) маршрутизатора, підключеного до зовнішньої мережі хост-машини. Його інженерним призначенням є забезпечення зв'язку внутрішнього контейнеризованого стеку з глобальними мережами. Через цей інтерфейс проектується первинне завантаження Docker-образів із офіційних репозиторіїв, віддалений доступ адміністратора до веб-панелей керування та трансляція аварійних push-повідомлень від сервера інформування на мобільні пристрої користувачів. Цей порт підлягає жорсткому захисту правилами фільтрації вхідного трафіку.

➤ Інші мережеві інтерфейси (ether2, ether3, ether4):

Конфігуруються як загальні внутрішні порти без прямої прив'язки до зовнішніх адаптерів хост-машини. Їхнім призначенням є інтеграція в робочий простір симулятора GNS3 для організації магістральних каналів зв'язку (Trunk Links) з емульованими мережами офісів. Порт ether2 виділяється для обслуговування інфраструктури Офісу 1, порт ether3 - Офісу 2, а порт ether4 - Офісу 3. На базі цих загальних інтерфейсів у подальшому розгортається логіка створення парних суб-влан інтерфейсів, що забезпечує злиття обчислювального edge-ядра маршрутизатора з розподіленим контуром пристроїв автоматизації.

		Зубрицький П.А.			35ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

2.3. Математичний розрахунок адресного простору та проєктування віртуальних локальних мереж (VLAN)

При побудові сучасних розподілених мереж Інтернету речей для корпоративного сегмента на перший план висувуються жорсткі вимоги щодо кібербезпеки, відмовостійкості та оптимізації широкомовних доменів (Broadcast Domains). Оскільки кінцеві пристрої автоматизації, виконавчі механізми та датчики фізичної безпеки функціонують на базі вбудованих мікроконтролерів і мають обмежені апаратні можливості локального захисту, їхнє знаходження в загальному широкомовному сегменті разом із адміністративними серверами чи комп'ютерами користувачів створює критичні ризики несанкціонованого доступу, сканування портів та компрометації технологічного трафіку.

Для розв'язання проблем безпеки та забезпечення логічної ізоляції інформаційних потоків у даному проєкті застосовано технологію віртуальних локальних мереж (Virtual Local Area Network, VLAN) відповідно до міжнародного індустріального стандарту IEEE 802.1Q. Використання міток (тегов) у заголовках кадрів дозволяє розділити єдине фізичне чи віртуальне комутаційне середовище маршрутизатора MikroTik CHR на декілька ізольованих на каналному рівні (L2) логічних підмереж. Обмін пакетами між створеними сегментами стає можливим виключно шляхом підняття трафіку на мережевий рівень (L3) під суворим контролем правил міжмережевого екрану RouterOS.

Методологія безкласової маршрутизації та розрахунок адресних блоків

Розрахунок адресного простору для спроектованої мережі виконано на основі методології безкласової міждоменної маршрутизації (Classless Inter-Domain Routing, CIDR). Як базовий простір використано діапазон приватних IP-адрес класу С мережі 192.168.0.0.

Для забезпечення максимальної щільності кіберзахисту в межах кожного з трьох територіально розподілених офісів (Офісу 1, Офісу 2 та Офісу 3) впроваджено концепцію мікросегментації. Кожен офісний простір математично розділено навпіл за допомогою застосування масок підмереж довжиною /25 (що відповідає масці підмережі 255.255.255.128). Це дозволило розгорнути по два

		Зубрицький П.А.			36КІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

автономні широкомовні домени в межах одного офісу: перший - для обчислювальних засобів персоналу (-users), другий - безпосередньо для інфраструктури Інтернету речей (-iot).

Вибір маски підмережі /25 є інженерно обґрунтованим з огляду на наступні критерії:

Ємність та розподіл хостів: Кожна суб-мережа надає $2^{32-25} - 2 = 126$ унікальних робочих IP-адрес для підключення хостів (за вирахуванням адреси підмережі та широкомовної адреси Broadcast).

Динамічна адресація та резервування: Розподіл адрес всередині суб-мереж чітко корелює з проєктними пулами DHCP (DHCP Pools). Для користувацьких сегментів (-users) пул динамічних адрес охоплює діапазони від .1 до .99 та від .101 до .126, фіксуючи адресу .100 як статичний IP-шлюз за замовчуванням. Для технологічних сегментів (-iot) діапазон пулу охоплює адреси від .129 до .199 та від .201 до .254, залишаючи адресу .200 під шлюз інтерфейсу маршрутизатора. Це повністю виключить конфлікти адресації та структурує логіку розподілу.

Масштабованість інфраструктури: Наявність 126 адрес у кожному окремому суб-влані повністю задовольняє поточні вимоги та закладає солідний інфраструктурний резерв для подальшого нарощування кількості embedded-систем та IoT-датчиків.

Детальний інженерний розрахунок логічної сегментації, закріплених ідентифікаторів інтерфейсів, масок, динамічних пулів та шлюзів для всієї інфраструктури представлено у таблиці 2.1.

		Зубрицький П.А.			37 ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Таблиця 2.1 - Схема логічної сегментації, ідентифікації інтерфейсів та адресації підмереж

Системна назва інтерфейсу в RouterOS	Ідентифікатор VLAN	Базовий інтерфейс	Діапазон обчислювальних IP-адрес хостів	Маска підмережі	Шлюз за замовчуванням
vlan11-office1-users	vlan11	ether2	192.168.1.1-192.168.1.99, 192.168.1.101-192.168.1.126	/25	192.168.1.100
vlan12-office1-iot	vlan12	ether2	192.168.1.129-192.168.1.199, 192.168.1.201-192.168.1.254	/25	192.168.1.200
vlan21-office2-users	vlan21	ether3	192.168.2.1-192.168.2.99, 192.168.2.101-192.168.2.126	/25	192.168.2.100
vlan22-office2-iot	vlan22	ether3	192.168.2.129-192.168.2.199, 192.168.2.201-192.168.2.254	/25	192.168.2.200
vlan31-office3-users	vlan31	ether4	192.168.3.1-192.168.3.99, 192.168.3.101-192.168.3.126	/25	192.168.3.100
vlan32-office3-iot	vlan32	ether4	192.168.3.129-192.168.3.199, 192.168.3.201-192.168.3.254	/25	192.168.3.200

Організація міжсегментної взаємодії та комутаційні особливості

Практична реалізація наведеної схеми адресації та тегування проектується шляхом створення віртуальних інтерфейсів типу VLAN безпосередньо поверх фізичних гігабітних портів маршрутизатора MikroTik CHR. Інтерфейси першого офісу (vlan11, vlan12) прив'язуються до порту ether2, другого офісу (vlan21, vlan22) - до порту ether3, а порту третього офісу (vlan31, vlan32) - до порту ether4. Всі зазначені фізичні порти логічно об'єднуються внутрішнім програмним мостом internal типу Bridge для організації єдиної керованої матриці комутації.

Оскільки всі шість розроблених суб-вланів оперують у різних широкомовних доменах, пряма трансляція кадрів між ними на каналному рівні (L2) є апріорі неможливою. Будь-яка інформаційна взаємодія (наприклад, спроба передачі даних від Python-шлюзу з vlan12 до суміжного користувацького сегмента vlan11) змушує пакети підніматися на мережевий рівень (L3) для маршрутизації в ядрі RouterOS.

Це дозволяє задіяти інструмент IP Firewall Filter для реалізації суворої політики безпеки периметра: транзитний трафік від технологічних хостів -іот підлягає глибокій фільтрації, де датчикам дозволено надсилати структуровані повідомлення виключно на IP-адреси контейнеризованих сервісів MQTT-брокера Mosquitto та сервера інформування ntfy, тоді як будь-які спроби несанкціонованого доступу до комп'ютерів користувачів повністю відсікаються.

		Зубрицький П.А.			39ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

2.4. Структурно-логічна модель інформаційного обміну та структура телеметричних даних

Ефективність функціонування розподіленої інфраструктури Інтернету речей безпосередньо залежать від розробки стабільної та прозорої моделі інформаційного обміну між крайовими вузлами (Edge шлюзами офісів) та центральними обчислювальними сервісами. Оскільки в межах проєкту реалізовано сувору мікросегментацію на рівні суб-вланів для трьох офісів підприємства, архітектура прикладного рівня повинна логічно відображати цей поділ, забезпечуючи чітке розмежування та фільтрацію інформаційних потоків.

Логічна модель взаємодії елементів спроектована за двома основними напрямками:

- Клієнтський контур (Телеметрія та безпека офісів): Асинхронна передача метрик та миттєвих тривожних сповіщень від крайових Python-шлюзів офісів до контейнеризованого брокера повідомлень.
- Адміністративний контур (Діагностика ядра системи): Регулярний збір системних метрик утилізації апаратних ресурсів та журналів подій самого маршрутизатора MikroTik CHR для забезпечення контролю стабільності та відмовостійкості всієї платформи.

Ієрархічна структура адресного простору MQTT-топиків

Для реалізації інформаційного обміну за протоколом MQTT спроектовано сувору деревоподібну ієрархію топиків (Topic Paths). Адресація побудована за принципом класифікації від загального до конкретного: *[ідентифікатор_офісу]/[назва_модуля]/[метрика]*. Така структура дозволяє споживачам даних ефективно здійснювати фільтрацію повідомлень за допомогою стандартних масок підстановки (Wildcards).

Спроектована схема розподілу інформаційних потоків, періодичність їхнього оновлення та інженерне призначення в межах загальної архітектури представлені в таблиці 2.2.

Таблиця 2.2.1 - Проектна схема ієрархії MQTT-топиків та типізації даних

		Зубрицький П.А.			40KKIKB.KP.B - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Цільовий MQTT-топiк (Topic Path)	Формат даних (Data Type)	Періодичність трансляції	Інженерне призначення інформаційного каналу
office1/system	String (ON/OFF)	Кожні 3 сек	Контроль загального стану працездатності шлюзу Офісу 1.
office1/cam/status	String (On/Off)	Кожні 3 сек	Моніторинг мережевої доступності IP-камери відеоспостереження.
office1/cam/motion	String (On/Off)	Подійно	Передача тривожного сигналу при реєстрації руху в офісі.
office1/climat/temp	Float (°C)	Кожні 3 сек	Трасляція поточних температурних показників робочої зони.
office1/energy/power	Float (кВт)	Кожні 3 сек	Моніторинг споживання активної електричної потужності.
office2/system	String (ON/OFF)	Кожні 3 сек	Контроль загального стану працездатності шлюзу Офісу 2.
office2/glass/alarm	String (Motion/Off)	Подійно	Активація сигналу тривоги при пошкодженні віконних конструкцій.
office2/gate/access	String (CARD_ID)	Подійно	Передача унікального коду перепустки через RFID-браму СКУД.
office2/smoke/co2	Float (ppm)	Кожні 3 сек	Моніторинг рівня концентрації вуглекислого газу в повітрі.
office3/system	String (ON/OFF)	Кожні 3 секунд	Контроль загального стану працездатності шлюзу Офісу 3.

Таблиця 2.2.2 - Проектна схема ієрархії MQTT-топіків та типізації даних

Цільовий MQTT-топік (Topic Path)	Формат даних (Data Type)	Періодичність трансляції	Інженерне призначення інформаційного каналу
office3/flood/alarm	String (Leakage/Off)	Подійно	Генерація аварійного сповіщення при фіксації протікання води.
office3/light/state	String (Active/Standby)	Кожні 3 секунд	Відображення режиму роботи контролера системи освітлення.
office3/heat/power	Float (Гкал/год)	Кожні 3 секунд	Моніторинг накопичувального рівня споживання теплової енергії.
mikrotik/resources	JSON Object	Кожні 30 секунд	Збір метрик утилізації CPU, RAM та HDD ядра маршрутизатора.
mikrotik/logs	JSON Object	Кожні 30 секунд	Трансляція відфільтрованих журналів подій та безпеки RouterOS.
mikrotik/connection	JSON Object	Кожні 30 секунд	Трансляція поточного стану таблиці відслідковування з'єднань (Connection Tracking) міжмережевого екрану RouterOS.

Обґрунтування вибору та стандартизація структури корисного навантаження

Для забезпечення повної сумісності між різнорідними елементами системи (Python-шлюзами, брокером, оркестратором та базою даних Grafana) у проєкті впроваджено жорстку стандартизацію структури корисного навантаження (Payload). Як базовий формат представлення даних обрано легковажний текстовий формат JSON. [17]

Обґрунтування вибору JSON базується на таких критеріях:

- **Мінімальний overhead:** Структура містить мінімум службових символів, що дозволяє передавати корисні дані з мінімальним задіянням пропускну здатності суб-вланів.

		Зубрицький П.А.			4ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

- Уніфікація парсингу: Для спрощення логіки аналізу даних на стороні споживачів, усі клієнтські пакети офісів спроектовані з єдиним кореневим ключем "value". Наприклад, телеметрія температури передається як {"value": 22.5}, а статус аварії - як {"value": "Leakage"}. Це дозволяє використовувати уніфіковані програмні парсери для обробки рядків, знижуючи навантаження на обчислювальні ядра крайового шлюзу.
- Масштабованість для системних метрик: Для комплексних об'єктів (таких як ресурси маршрутизатора mikrotik/system/resources) формат JSON дозволяє інкапсулювати кілька взаємопов'язаних параметрів в один пакет (cpu_load, free_memory_mb, hdd_usage_percent), забезпечуючи цілісність зрізу даних у часі.

Генеральна структурно-логічна схема взаємодії елементів IoT-інфраструктури

Для об'єднання розрахованих мережевих суб-вланів та спроектованої моделі MQTT-топиків у єдину працездатну систему розроблено генеральну структурно-логічну схему взаємодії сервісів. Ця модель відображає архітектуру розподілу обов'язків на трьох рівнях: крайовому (фізичні хости та Python-шлюзи), комутаційному (ядро MikroTik CHR) та аналітично-серверному (окремі ізольовані контейнери інфраструктури).

Загальна структурна модель інформаційних потоків та логічних зв'язків комплексу представлена на рисунку 2.1.

		Зубрицький П.А.			43ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

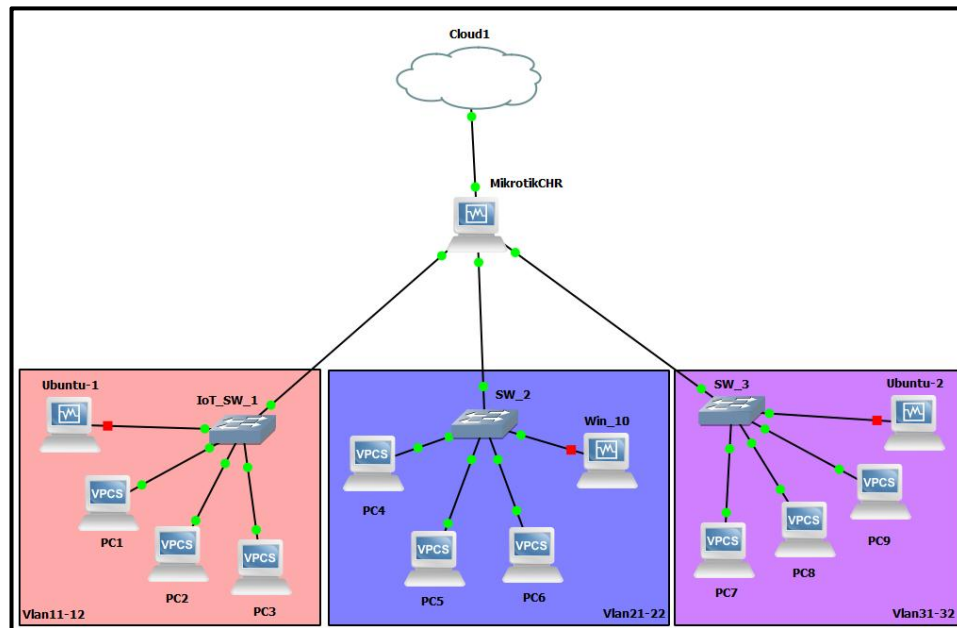


Рис. 2.1. Схема структурно-топологічної організації мережі лабораторного стенду

Спроектована в межах лабораторного стенду топологія (рис. 2.1) відображає архітектуру розподілу обчислювальних обов'язків та забезпечує повну фізико-логічну відповідність розрахованій схемі адресації. Комплекс розділений на три автономні інфраструктурні рівні, взаємодія між якими жорстко регламентується на мережевому контурі:

Зовнішній контур керування та магістрального зв'язку (WAN-рівень): У верхній частині ієрархії розташований об'єкт Cloud1, який імітує хмарне середовище та забезпечує трансляцію пакетів із фізичної мережі хост-машини на перший інтерфейс віртуального маршрутизатора MikrotikCHR (ether1-wan). Цей канал призначений для забезпечення виходу внутрішнього серверного стеку в глобальну мережу, віддаленого адміністрування системи інженером через WinBox, а також для трансляції аварійних вихідних push-алертів підсистеми сповіщень.

Комутаційне edge-ядро системи (L2/L3-рівень): Центральний обчислювальний вузол MikrotikCHR виступає головним шлюзом міжмережевого екранування та маршрутизації. Від його гігабітних портів відходять три незалежні магістральні лінії зв'язку до відповідних офісних приміщень. Всередині операційної системи роутера створюється програмний міст internal типу Bridge, на якому розгортаються парні інтерфейси суб-вланів. Такий підхід забезпечує

канальну ізоляцію (L2 Broadcast Isolation) між користувацьким та технологічним трафіком кожного об'єкта.

Крайовий рівень автоматизації та кінцевих пристроїв (Edge-рівень офісів): Внутрішній простір офісів територіально та логічно ізольований за допомогою виділених комутаторів другого рівня та кольорового маркування зон:

- Офіс 1 (Червона зона): Магістральний канал підключений до комутатора IoT_SW_1. Роль крайового обчислювального шлюзу виконує віртуальна машина Ubuntu-1. На базі операційної системи цього хоста розгортається програмний скрипт автоматизації, який через локальні сокети здійснює моніторинг доступності кінцевих вузлів PC1, PC2 та PC3 (що імітують IP-камеру, датчик клімату та смарт-лічильник електроенергії у суб-сегменті vlan12-office1-iot);
- Офіс 2 (Синя зона): Комутація забезпечується пристроєм SW_2. Крайовим шлюзом виступає операційна система Win_10, де функціонує відповідний Python-скрипт. Він здійснює безперервний аудит безпекового контуру офісу, опитуючи віртуальні хости PC4, PC5 та PC6 (імітація засобів СКУД, датчиків розбиття скла та сповіщувачів диму в межах vlan22-office2-iot);
- Офіс 3 (Фіолетова зона): Побудований на базі комутатора SW_3 та керуючого хоста Ubuntu-2. Програмний шлюз третього офісу забезпечує збір телеметрії інженерних комунікацій з кінцевих вузлів PC7, PC8 та PC9 (контроль протікання води, лічильники тепла та системи керування освітленням у межах vlan32-office3-iot).

Спроектована в GNS3 схема комутації дозволяє повністю виключити прямий несанкціонований обмін кадрами між пристроями автоматизації та комп'ютерами персоналу (-users). Будь-який телеметричний або тривожний пакет від хостів PC1–PC9 примусово спрямовується на шлюзи керування, звідки через транзитні ланцюжки маршрутизатора MikrotikCHR транслюється у внутрішній стек контейнерів (Mosquitto, ntfy, Node-RED), що розгорнуті безпосередньо в персистентній пам'яті ext4 накопичувача sata1 самого роутера.

		Зубрицький П.А.			45ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Генеральна структурно-логічна схема взаємодії та проектування сервісних обов'язків мікросервісів

Для об'єднання розрахованих мережевих суб-вланів та спроектованої моделі ієрархічних MQTT-топіків у єдину цілісну систему розроблено генеральну структурно-логічну схему взаємодії прикладного рівня. Ця модель відображає теоретичну архітектуру розподілу функціональних обов'язків між елементами інфраструктури на трьох рівнях: крайовому (фізичні хости та Python-шлюзи офісів), комутаційному (маршрутизатор MikroTik CHR) та аналітично-серверному (ізольований стек мікросервісів).

Загальна структурна модель інформаційних потоків та логічних зв'язків комплексу представлена на рисунку 2.2.

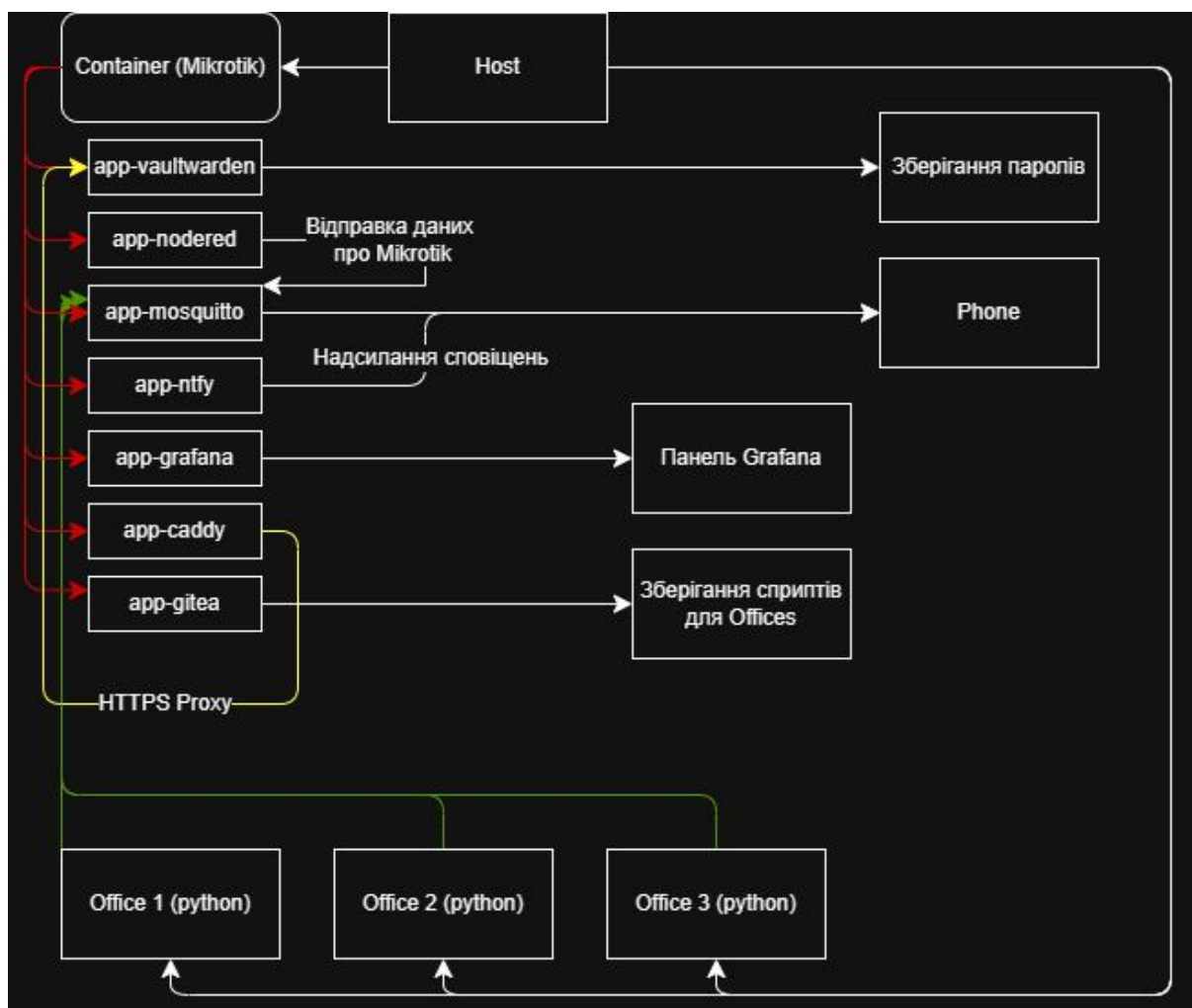


Рис. 2.2. Структурно-логічна схема взаємодії прикладних сервісів інфраструктури

Теоретичний аналіз спроектованої структури взаємодії (рис. 2.2) дозволяє формалізувати системні обов'язки кожного елемента та напрямки руху інформаційних потоків на етапі проектування інфраструктури:

- Проектування транспортного контуру телеметрії крайового рівня: Центральним вузлом консолідації клієнтських даних визначено мікросервіс app-mosquito. На цей елемент покладається завдання забезпечення асинхронного обміну повідомленнями за протоколом MQTT. Спроектована логіка передбачає, що брокер є єдиною точкою прийому регулярних JSON-пакетів телеметрії, які транслюються крайовими Python-шлюзами (Office 1, Office 2, Office 3) по виділених лініях зв'язку через порт 1883.
- Проектування контуру координації та моніторингу системних метрик: Для автоматизації збору внутрішньої діагностичної інформації маршрутизатора до структури системи інтегровано сервіс оркестрації app-nodered. Проектна роль цього компонента полягає у виконанні регулярних низькорівневих запитів до внутрішніх інтерфейсів Container (Mikrotik) з фіксованим часовим інтервалом у 30 секунд. Завданням сервісу є декомпозиція сирих параметрів утилізації обчислювальних ресурсів та сесій таблиці connection для їх подальшої публікації у загальну чергу повідомлень брокера app-mosquito. Для накопичення та візуалізації сформованих часових рядів передбачено взаємодію з аналітичним сервісом app-grafana, що логічно замикається на блок «Панель Grafana».
- Проектування автономного контуру аварійного алертингу: З метою мінімізації затримок при виникненні загроз пожежної безпеки, затоплення або несанкційованого доступу, контур миттєвих сповіщень спроектовано як незалежну лінію зв'язку. Алгоритм реалізації передбачає, що сервіс app-ntfy приймає тривожні повідомлення, обробляє їх через логічний блок «Надсилення сповіщень» та забезпечує їх пряму трансляцію кінцевому споживачу на об'єкт Phone (мобільний пристрій адміністратора).
- Проектування захисного периметра та адміністративного контуру: Для виключення несанкційованого доступу до систем керування на етапі

		Зубрицький П.А.			47KKIB.KP.B - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

проектування закладено архітектурний принцип єдиної точки входу за допомогою об'єкта HTTPS Proxy. Запит від адміністратора з керуючого вузла Host примусово спрямовується на зворотний проксі-сервер app-caddy, на який покладається обов'язок захисту внутрішнього веб-периметра та шифрування сесій адміністрування суміжних платформ. Цілісність інфраструктурного коду крайових шлюзів забезпечується інтеграцією локальної системи контролю версій app-gitea (блок «Зберігання скриптів для Offices»), а безпечне зберігання та менеджмент автентифікаційних токенів, паролів та ключових даних ізольовано делеговано менеджеру секретів app-vaultwarden (блок «Зберігання паролів»).

		Зубрицький П.А.			48 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

2.5. Алгоритми функціонування крайового програмного забезпечення шлюзів

Ефективність збору первинної телеметрії та швидкість реакції підсистеми сповіщень при виникненні позаштатних ситуацій безпосередньо залежать від архітектури та внутрішньої логіки функціонування програмного забезпечення крайового рівня (Edge Level). У межах даного проєкту для трьох територіально розподілених об'єктів спроектовано індивідуальні алгоритми функціонування шлюзів автоматизації, реалізовані на мові програмування Python. Повні тексти розроблених програмних модулів винесені в Додатки А, Б та В, а в цьому підрозділі наводиться теоретичне обґрунтування та аналіз логічної структури їхньої роботи. [14]

Логіка ініціалізації сесій та багатопотокового обміну

Проєктна модель функціонування крайового програмного забезпечення базується на циклічному подієво-орієнтованому підході. При запуску програмного модуля на керуючому хості офісу (Ubuntu-1, Win_10 або Ubuntu-2) першочергово виконується етап ініціалізації системних змінних:

- Завантажуються параметри мережевої ідентифікації (унікальний Client ID шлюзу, IP-адреса та TCP-порт центрального MQTT-брокера).
- Ініціюється об'єкт мережевого клієнта та виконується процедура встановлення з'єднання з сервером app-mosquitto.

З метою виключення затримок при збоях у каналах зв'язку, алгоритм передбачає переведення процесу обміну MQTT-пакетами у фоновий асинхронний потік виконання (фоновий loop). Це дозволяє програмному шлюзу підтримувати постійну сесію з брокером (надсилання сервісних пакетів PINGREQ та обробка PINGRESP) та паралельно виконувати основне завдання - безперервний низькорівневий аудит підключеного технологічного обладнання в межах свого VLAN-сегмента.

Алгоритм контролю мережевої доступності кінцевих хостів

Оскільки імітація фізичних датчиків та виконавчих механізмів у середовищі GNS3 реалізована за допомогою окремих віртуальних хостів, першочерговим

		Зубрицький П.А.			4ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

інженерним завданням шлюзу є верифікація їхньої фізичної наявності на лінії. Для цього спроектовано логічний блок циклічного ехо-опитування.

Внутрішня архітектура цього алгоритму базується на взаємодії з утилітами операційної системи на низькому рівні. Програма здійснює формування запитів ping до кожної закріпленої за датчиками IP-адреси. Для забезпечення високої швидкодії системи та запобігання тривалим затримок в алгоритмі закладено суворі обмеження:

- Кількість відправлених ехо-пакетів до одного вузла за одну ітерацію обмежена одним кадром;
- Максимальний час очікування відповіді (Timeout) жорстко лімітується на рівні 1 секунди.

При отриманні успішної ехо-відповіді від хоста датчика алгоритм генерує логічну одиницю (True), а при перевищенні ліміту часу очікування або обриві зв'язку - логічний нуль (False), що є тригером для зміни режиму роботи шлюзу.

Сценарії обробки даних: штатний та аварійний режими

На основі результатів логічного моніторингу доступності, внутрішня архітектура безперервного циклу програми розподіляє обробку даних за двома незалежними сценаріями:

- Штатний сценарій функціонування (Датчик активний): Програма формує системну мітку "On" для публікації у відповідний топик доступності пристрою. Для імітації реальних умов експлуатації об'єкта в алгоритм інтегровано математичний апарат генерації псевдовипадкових чисел. Фізичні величини (температура, рівень CO₂, споживана потужність лічильників) моделюються у межах суворо заданих реалістичних інженерних діапазонів з округленням значень. Для дискретних подієвих тригерів (фіксація руху, спрацювання датчика розбиття скла) застосовується ймовірнісний фільтр, який примусово переводить стан сенсора в активну фазу лише при виконанні випадкової математичної умови, що моделює реальну подію.

		Зубрицький П.А.			50ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

- Аварійний сценарій функціонування (Збій на лінії): У ваді фіксації логічного нуля при опитуванні хоста, шлюз миттєво блокує роботу математичного блоку генерації випадкових чисел. У відповідну гілку MQTT-топиків примусово відправляється заздалегідь спроектований JSON-пакет з міткою {"value": "OFFLINE"}. Це дозволяє на рівні проєкту закласти механізм негайної візуалізації технічних несправностей інфраструктури.

Логіка прямої генерації push-сповіщень

Найбільш пріоритетна гілка спроектованого алгоритму відповідає за обробку критичних станів систем фізичної та пожежної безпеки об'єкта. При виявленні тригерних значень (наприклад, зафіксовано затоплення в Офісі 3 або дим в Офісі 2), крайове програмне забезпечення активує підсистему негайного інформування персоналу.

З метою забезпечення максимальної відмововитривалості системи, логіка генерації тривожних сповіщень побудована за схемою прямої взаємодії з сервером сповіщень, минаючи черги повідомлень MQTT-брокера. Програма виконує синхронний HTTP POST-запит безпосередньо на мережеву адресу мікросервісу app-ntfy. Структура заголовків (Headers) спроектованого HTTP-пакета містить жорстко регламентовані метадані:

- Тематичний заголовок (Title) із зазначенням типу та локалізації події;
- Маркер критичності (Priority), який для аварійних подій встановлюється у значення high або max;
- Інформаційні теги (Tags) для візуального виділення тривоги.

Така логічна побудова гарантує, що при отриманні пакета сервер ntfy змусить кінцевий пристрій адміністратора обійти системні обмеження звуку та згенерувати негайний сигнал тривоги. Завершуючи ітерацію обробки, алгоритм переводить програму в стан сну на фіксовану час (від 3 до 5 секунд залежно від конфігурації офісу), після чого цикл повторюється знову.

		Зубрицький П.А.			5ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

2.6. Проектування підсистеми міжмережевого екранування периметра мережі

Оскільки спроектована локальна інфраструктура об'єднує в єдиний комплекс загальнодоступні користувацькі контури (-users) та критично важливі технологічні сегменти Інтернету речей (-iot), розробка архітектури кіберзахисту є невід'ємною частиною системного проектування. Обмежені апаратні можливості кінцевих IoT-мікроконтролерів унеможливають розгортання на них локальних засобів захисту (антивірусів, персональних фаєрволів). З огляду на це, головне завдання із забезпечення цілісності, конфіденційності та доступності даних покладається на комутаційне ядро системи - маршрутизатор MikroTik CHR під керуванням операційної системи RouterOS.

Концепція мікросегментації та архітектурна модель Zero Trust

В основу проектування підсистеми безпеки периметра покладено сучасну індустріальну концепцію Zero Trust («Нульова довіра»). Згідно з цією моделлю, будь-який мережевий суб-сегмент апріорі вважається потенційно небезпечним, а будь-яка міжмережева взаємодія за замовчуванням є заблокованою, доки її легітимність не буде чітко підтверджена правилами фільтрації. [13]

Практична реалізація цієї концепції на етапі теоретичного проектування базується на декомпозиції правил міжмережевого екрану IP Firewall за двома базовими логічними ланцюжками обробки пакетів:

Ланцюжок INPUT - відповідає за фільтрацію пакетопотоків, спрямованих безпосередньо на маршрутизатор (спроби підключення до самого ядра RouterOS або до розгорнутого всередині стеку контейнерів).

Ланцюжок FORWARD - відповідає за фільтрацію транзитного трафіку, який намагається пройти крізь маршрутизатор з одного віртуального суб-влану в інший.

Проектування логіки фільтрації ланцюжка транзитного трафіку (Forward Policy)

Для унеможливлення поширення мережевих атак всередині локального контуру підприємства спроектовано сувору логіку розмежування доступу між шістьма створеними віртуальними мережами. Матриця дозволених транзитних переходів базується на таких критеріях:

		Зубрицький П.А.			5ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

Ізоляція технологічного контуру: Спроектовано повну заборону на ініціалізацію будь-яких сесій зв'язку з технологічних мереж vlan12, vlan22 та vlan32 у напрямку користувацьких підмереж vlan11, vlan21 та vlan31. Це унеможливило компрометацію робочих станцій персоналу у випадку, якщо зломисник отримає фізичний чи логічний доступ до одного з крайових датчиків.

Авторизація прикладних потоків: Для забезпечення функціонування системи, шлюзам автоматизації дозволено ініціювати транзитні сесії виключно до виділеної IP-адреси стеку контейнерів за двома цільовими векторами:

Трансляція телеметрії на порт MQTT-брокера (1883/TCP);

Трансляція тривожних сповіщень на порт сервера пуш-інформування (3005/TCP).

Двосторонній контроль сесій: Проектна логіка передбачає обов'язкове використання механізму Stateful Packet Inspection (SPI). Це дозволяє автоматично пропускати зворотні пакети для вже встановлених легітимних з'єднань (connection-state=established,related) і моментально блокувати будь-які спроби ініціалізації нових несанкціонованих сесій (connection-state=invalid).

		Зубрицький П.А.			53ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Проектування логіки захисту керуючого інтерфейсу (Input Policy)

Захист самого обчислювального ядра маршрутизатора від атак типу «відмова в обслуговуванні» (DoS/DDoS) та спроб підбору паролів до служб керування базується на жорсткому обмеженні вхідного периметра. На рівні проекту закладаються такі правила:

Виділений сегмент менеджменту: Будь-який доступ до сервісних портів керування RouterOS (порт утиліти WinBox $8291/\text{TCP}$, порти безпечного доступу SSH) дозволяється виключно для обмеженого пулу IP-адрес із адміністративного контуру. Спроби авторизації з технологічних підмереж або з мереж загального користувача автоматично відсікаються.

Захист веб-інфраструктури мікросервісів: Оскільки зовнішній доступ до аналітичних панелей Grafana та середовища Node-RED здійснюється через зворотний проксі-сервер app-caddy, на зовнішньому WAN-інтерфейсі (ether1-wan) проектується повне закриття прямих портів $3000/\text{TCP}$ та $1880/\text{TCP}$. Єдиним легітимним вікном для вхідного веб-трафіку визначається захищений порт $443/\text{TCP}$ (HTTPS), де шифрування сесій забезпечується сертифікатами безпеки.

		Зубрицький П.А.			54ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі кваліфікаційної роботи виконано повний комплекс науково-інженерних розрахунків, структурного моделювання та теоретичного проєктування локальної IoT-інфраструктури підприємства. За результатами проведених досліджень сформовано такі висновки:

1. Обґрунтовано та розраховано оптимальні системні параметри віртуальної edge-платформи MikroTik CHR (2 ядра CPU, 2 ГБ RAM, додатковий диск sata1 на 10 ГБ під файлову систему ext4). Це гарантує стабільне паралельне функціонування комутаційного ядра RouterOS та вбудованого контейнеризованого стеку мікросервісів в умовах обмежених апаратних ресурсів.
2. Розроблено структурно-топологічну схему організації мережевих інтерфейсів із поділом на зовнішній контур захисту периметра (ether1-wan) та внутрішні магістральні канали (Trunk Links ether2–ether4) для інтеграції комутаційного ядра з розподіленою інфраструктурою офісів у симуляторі GNS3.
3. Математично розраховано безкласовий адресний простір приватних мереж на основі методології CIDR із застосуванням масок довжиною /25. Це дозволило розгорнути 6 ізольованих на каналному рівні віртуальних локальних мереж стандарту IEEE 802.1Q (VLAN 11, 12, 21, 22, 31, 32), повністю відокремивши широкомовні домени користувачів від технологічного IoT-трафіку датчиків.
4. Спроектовано ієрархічну модель адресного простору прикладного рівня на базі структури деревоподібних MQTT-топиків, яка повністю відображає мережеву топологію об'єктів автоматизації та забезпечує уніфікований парсинг серіалізованих текстових JSON-пакетів телеметрії.
5. Розроблено логічну структуру та алгоритми функціонування крайового програмного забезпечення автоматизації офісів на мові Python. Логіка роботи скриптів базується на асинхронному фоновому моніторингу ліній зв'язку утилітою ring, математичному моделюванні фізичних метрик у JSON та прямій генерації високопріоритетних тривожних HTTP POST-запитів.
6. Сформовано теоретичну політику безпеки периметра за індустріальною концепцією Zero Trust, що регламентує суворі правила ієрархічної фільтрації вхідного трафіку (ланцюжок Input) та повне блокування несанкціонованих міжмережевих транзитних переходів між вланами (ланцюжок Forward).

		Зубрицький П.А.			5ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ, НАЛАШТУВАННЯ ТА ТЕСТУВАННЯ КОМПЛЕКСУ ІoT-МЕРЕЖІ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1. Реалізація підсистеми міжмережевого екранування та захисту периметра в RouterOS

Для матеріалізації спроектованої у підрозділі 2.5 політики безпеки Zero Trust («Нульова довіра») та забезпечення реального кіберзахисту розгорнутого стенду, у комутаційному ядрі маршрутизатора MikroTik CHR було виконано покрокове налаштування правил фільтрації міжмережевого екрану. Конфігурування реалізовано через графічний інтерфейс утиліти WinBox та дубльовано в спеціалізованому терміналі CLI операційної системи RouterOS за допомогою розробки суворого набору правил для вбудованої підсистеми IP Firewall Filter.

Конфігурування груп адрес (Address Lists) для оптимізації правил

З метою оптимізації обчислювальних ресурсів процесора віртуального маршрутизатора та спрощення структури правил фаєрвола, першим кроком практичної реалізації стало об'єднання ізольованих технологічних підмереж трьох офісів у єдиний логічний список адрес.

До створеного системного списку з назвою IoT_Networks через меню IP -> Firewall -> Address Lists було примусово включено розраховані у підрозділі 2.2 блоки адрес: 192.168.1.128/25 (vlan12-office1-iot), 192.168.2.128/25 (vlan22-office2-iot) та 192.168.3.128/25 (vlan32-office3-iot).

Графічне відображення сформованих пулів у інтерфейсі WinBox представлено на рисунку 3.1.

		Зубрицький П.А.			56КІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Firewall	Filter Rules	NAT	Mangle	Raw	Service Ports	Connections	Address Lists
New	Enable	Disable	Remove				
	List	Address	Timeout	Creation Time			
☐	Office 1 IoT Subnet	192.168.1.128/25		2026-06-15 0...			
☐	Office 2 IoT Subnet	192.168.2.128/25		2026-06-15 0...			
☐	Office 3 IoT Subnet	192.168.3.128/25		2026-06-15 0...			

Рис. 3.1. Налаштування груп мережеских адрес Address Lists в інтерфейсі WinBox

Конфігурування та верифікація правил ієрархічної фільтрації у WinBox

Результат успішного імпорту розробленого набору правил через CLI та поточний стан інтегрованої матриці міжмережевого екранування представлено на рисунку 3.2.

v4 Firewall Filter Rules NAT Mangle Raw Service Ports Connections Address Lists Layer7 Protocols																				
New		Enable	Disable	Remove														Find	Filter	all
☐	#	^	🚩	Action	Chain	Src. Address	Dst. Address	Src. Ad...	Dst. Ad...	Protocol	Src. Port	Dst. Port	In. Inter...	Out. Int...	In. Inter...	Out. Int...	Bytes	Packets		
Accept established and related input connections																				
☐	#	0	🟢	accept	input												33.1 KiB	508		
Drop invalid packets																				
☐	#	1	🔴	drop	input												0 B	0		
Allow WinBox and SSH only from Admin Host																				
☐	#	2	🟢	accept	input	192.168.0.103				6 (tcp)		8291,22					0 B	0		
Allow ICMP from Admin Host																				
☐	#	3	🟢	accept	input	192.168.0.103				1 (icmp)							0 B	0		
Drop all other input traffic (Zero Trust Base)																				
☐	#	4	🔴	drop	input												1116 B	19		
Accept established and related transit traffic																				
☐	#	5	🟢	accept	forward												7.9 KiB	71		
Drop invalid transit packets																				
☐	#	6	🔴	drop	forward												0 B	0		
Allow IoT Telemetry to Mosquito Broker																				
☐	#	7	🟢	accept	forward		192.168.0.102	IoT_Ne...		6 (tcp)		1883					0 B	0		
Allow IoT Alerts to ntly Server																				
☐	#	8	🟢	accept	forward		192.168.0.102	IoT_Ne...		6 (tcp)		3005					0 B	0		
Drop all other forward traffic (Complete Isolation)																				
☐	#	9	🔴	drop	forward												0 B	0		

Рис. 3.2. Конфігурація правил підсистеми IP Firewall Filter

Аналіз налаштованої конфігурації (рис. 3.2) дозволяє підтвердити повну відповідність практичної реалізації раніше обґрунтованій теоретичній моделі кіберзахисту Zero Trust:

		Зубрицький П.А.			57KKIKB.KP.B - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

Функціонування контуру самозахисту (Правила 0–4):

Правило 0 динамічно відстежує стан з'єднань (connection-state=established,related) і вже на момент зняття метрик успішно пропустило \$33.1\text{ КіБ}\$ легітимного зворотного трафіку, суттєво розвантажуючи процесор CHR;

Правила 2 та 3 жорстко лімітують вхідний доступ до сокетів WinBox (\$8291\text{ TCP}\$), SSH (\$22\text{ TCP}\$) та ICMP, відкриваючи їх виключно для авторизованого адміністративного хоста 192.168.0.103;

Фінальний скидач ланцюжка Input (правило 4) на момент фіксації вже заблокував 19 несанкціонованих пакетів (\$1116\text{ Б}\$), що підтверджує спроби сканування або паразитних запитів з інших підмереж.

Функціонування контуру мікросегментації вланів (Правила 5–9):

Транзитний контур (цеплінг forward) опрацьовує пакети за аналогічним SPI-принципом (правило 5 зафіксувало проходження \$7.9\text{ КіБ}\$ схваленого транзитного трафіку);

Правила 7 та 8 виступають суворо регламентованими шлюзами для крайового програмного забезпечення. Вони зв'язують сформований раніше список адрес IoT_Networks із цільовим сервером мікросервісів 192.168.0.102 на портах 1883 (транспорт MQTT) та 3005 (HTTP POST сервісу алертингу ntfy);

Фінальний деструктивний маркер Drop (правило 9) повністю замикає периметр технологічних сегментів. Будь-які пакети, що не відповідають критеріям легітимності (наприклад, спроби датчиків звернутися до користувацьких комп'ютерів персоналу), знищуються на рівні ядра RouterOS без генерації зворотних ICMP-повідомлень.

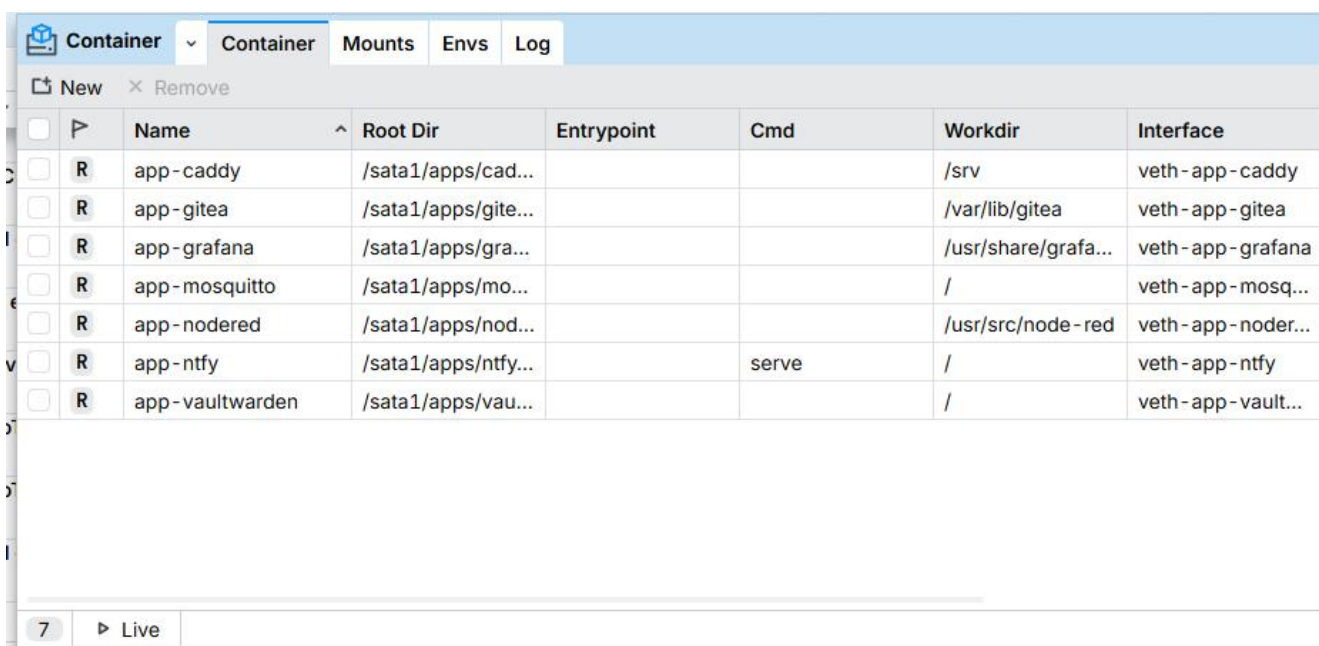
Спроектowana та впроваджена схема (рис. 3.2) доводить працездатність підсистеми екранування, де лічильники байт (Bytes/Packets) наочно демонструють динаміку фільтрації та відсікання несанкційованих запитів у реальному часі.

		Зубрицький П.А.			58 ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

3.2. Конфігурування та верифікація середовища контейнеризації в RouterOS

Після завершення конфігурування підсистеми міжмережевого екранування та розподілу адресних пулів, наступним інженерним завданням є введення в експлуатацію та перевірка працездатності прикладного рівня інфраструктури. На відміну від класичних рішень із використанням окремих серверів, у даному проєкті весь стек із семи мікросервісів розгорнутий безпосередньо в межах операційної системи віртуального маршрутизатора MikroTik CHR за допомогою вбудованої утиліти Container.

Для забезпечення персистентного (постійного) збереження даних, конфігураційних файлів та баз даних, образи контейнерів логічно відокремлені від завантажувальних секторів роутера. Стан та поточний статус запущених мікросервісів у графічному інтерфейсі WinBox представлено на рисунку 3.3.



		Name	Root Dir	Entrypoint	Cmd	Workdir	Interface
☐	R	app-caddy	/sata1/apps/cad...			/srv	veth-app-caddy
☐	R	app-gitea	/sata1/apps/gite...			/var/lib/gitea	veth-app-gitea
☐	R	app-grafana	/sata1/apps/gra...			/usr/share/grafa...	veth-app-grafana
☐	R	app-mosquitto	/sata1/apps/mo...			/	veth-app-mosq...
☐	R	app-nodered	/sata1/apps/nod...			/usr/src/node-red	veth-app-noder...
☐	R	app-ntfy	/sata1/apps/ntfy...		serve	/	veth-app-ntfy
☐	R	app-vaultwarden	/sata1/apps/vau...			/	veth-app-vault...

Рис. 3.3. Моніторинг статусів віртуальних контейнерів у диспетчері RouterOS

Аналіз налаштованого середовища контейнеризації (рис. 3.3) дозволяє верифікувати такі проєктні параметри системи:

- Статус виконання: Усі 7 спроектованих сервісів мають системний маркер R (Running), що підтверджує успішне виділення обчислювальних ресурсів процесора й оперативної пам'яті та стабільну роботу програмного забезпечення в реальному часі.

- Архітектура сховища даних: Кореневі директорії (Root Dir) для кожного додатка перенаправлені у відповідні підпапки на додатковому віртуальному накопичувачі (/sata1/apps/caddy, /sata1/apps/gitea і т.д.), форматизованому у стійку файлову систему ext4. Це гарантує цілісність логів Node-RED, репозиторіїв Gitea та баз даних Grafana при планових чи аварійних перезавантаженнях комутаційного ядра.
- Канальна інтеграція: Кожен контейнер прив'язаний до свого індивідуального віртуального інтерфейсу типу veth (veth-app-caddy - veth-app-vaultwarden). Ці інтерфейси логічно замикаються на внутрішній програмний міст маршрутизатора, забезпечуючи швидкісний обмін IP-пакетами між сервісами.

Для перевірки доступності вебінтерфейсів розгорнутого стеку з керуючого робочого місця адміністратора (Host) використовується веб-браузер. Маршрутизація прикладного веб-трафіку здійснюється через зворотний проксі-сервер app-caddy, який виступає єдиною захищеною точкою входу (Ingress) та розподіляє запити користувача в межах однієї робочої сесії, як відображено на рисунку 3.4.



Рис. 3.4. Відображення активних сесій панелей керування у веббраузері

Конфігурування та демонстрація робочого простору середовища Node-RED

Первинна координація потоків системної діагностики та зчитування параметрів ядра комутаційного маршрутизатора реалізована в межах мікросервісу app-nodered. Робочий простір налаштованого логічного потоку (інтерфейс Flow 1) представлено на рисунку 3.5.

		Зубрицький П.А.			60 ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

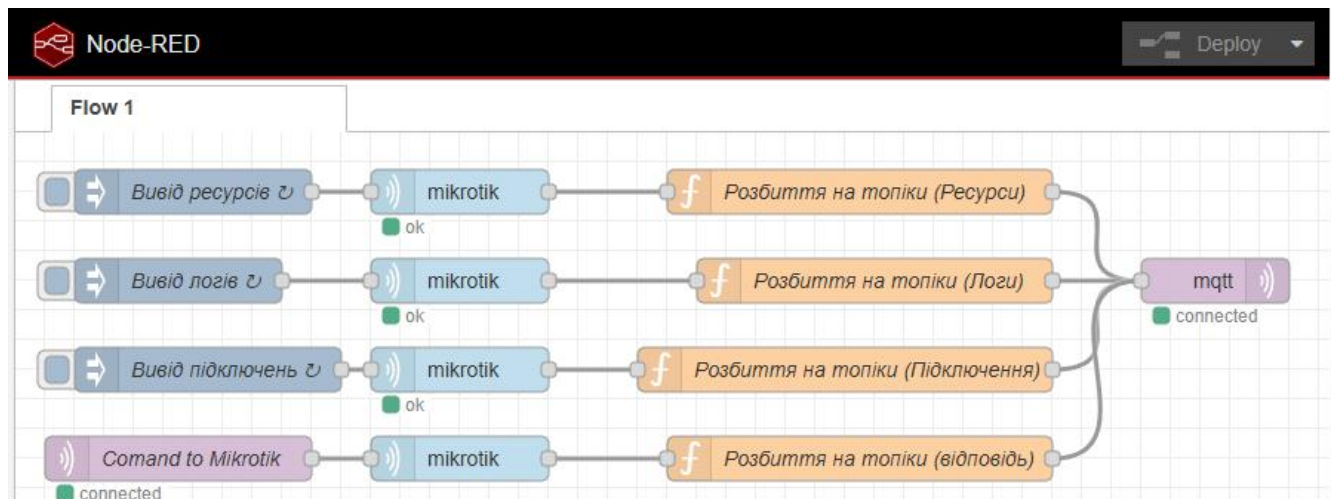


Рис. 3.5. Графічний інтерфейс логічного потоку обробки метрик у середовищі Node-RED

- Аналіз практичної реалізації (рис. 3.5) доводить використання подієво-орієнтованої палітри вузлів (Nodes) для формування чотирьох паралельних конвеєрів автоматизації обміну даними:
- Контур збору системних ресурсів: Ініціюється циклічним вузлом інжекції Вивід ресурсів. Він з фіксованим інтервалом часу запускає синій вузол mikrotik (діагностичний статус ok), який через внутрішнє API RouterOS витягує поточні масиви утилізації апаратного забезпечення. Після цього сирі дані надходять у функціональний помаранчевий блок Розбиття на топіки (Ресурси), де розпаршуються на окремі об'єкти (CPU, RAM, HDD) і спрямовуються на фінальну ноду публікації mqtt (статус connected).
- Контур збору системних логів ядра: Працює на базі тригера Вивід логів, який опитує внутрішній буфер повідомлень RouterOS. Отриманий масив текстових логів ядра маршрутизатора передається у функціональну ноду Розбиття на топіки (Логи). Тут інформація структурується за рівнем критичності (info, warning, critical) та серіалізується в JSON-формат для подальшої відправки в MQTT-чергу повідомлення
- Контур моніторингу мережевих сесій: Активується вузлом Вивід підключень. Завданням цієї гілки є безперервний збір поточного стану таблиці Connection Tracking міжмережевого екрану IP Firewall. Через відповідний блок API дані надходять у вузол Розбиття на топіки (Підключення), що дозволяє сформувати

динамічні часові ряди про щільність і кількість активних з'єднань між вланами для аналітичної системи.

- **Контур інтерактивного виконання команд:** На відміну від перших трьох контурів, цей потік працює у зворотному напрямку і є подієвим. Вузол підписки Comand to Mikrotik очікує вхідні керуючі повідомлення з MQTT-мережі (наприклад, команду від адміністратора на перезавантаження сервісу або зміну конфігурації). При отриманні пакета, він миттєво транслює його на виконавчу ноду mikrotik, а результат виконання та системну відповідь маршрутизатора через блок Розбиття на топіки (відповідь) відправляє назад у брокер для верифікації.

Верифікація роботи та відображення аналітичного дашборду в Grafana

Для візуалізації часових рядів, що характеризують апаратну стабільність платформи MikroTik CHR, використовується мікросервіс аналітики app-grafana. Запущений та сконфігурований спеціалізований дашборд Mikrotik_Resources представлено на рисунку 3.6.

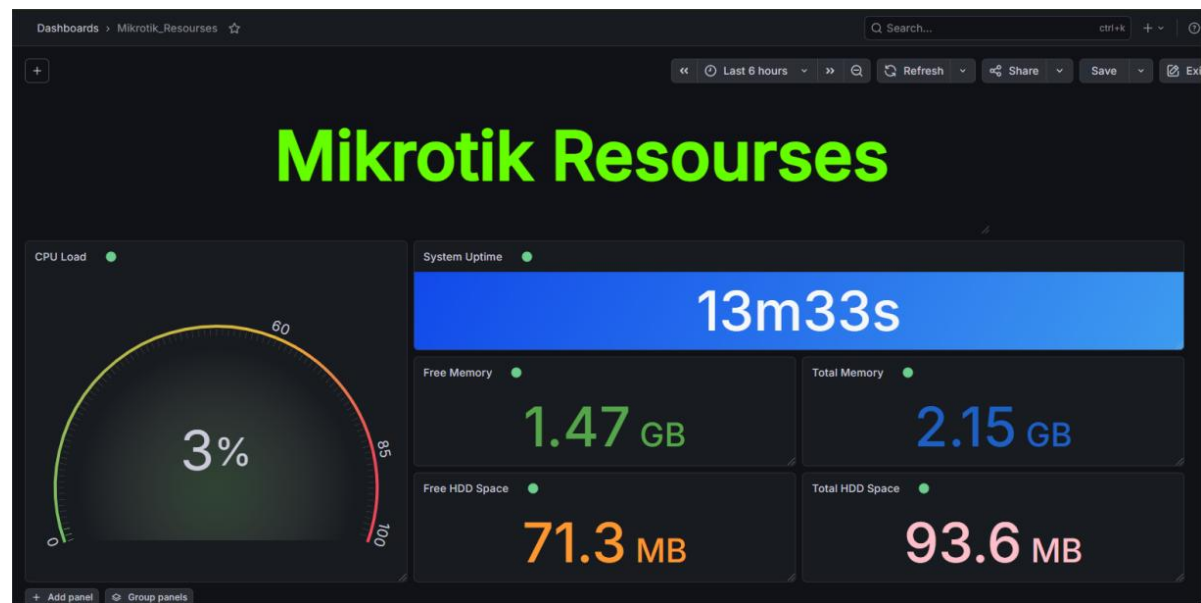


Рис. 3.6. Панель аналітичного моніторингу телеметрії та системних ресурсів у Grafana

Як демонструє інтерфейс моніторингу (рис. 3.6), аналітична панель забезпечує динамічний рендеринг метрик, які зчитуються середовищем Node-RED через API та транслюються в MQTT-топіки. Спроектований дашборд дозволяє

адміністратору у реальному часі контролювати такі критичні параметри комутаційного ядра:

- Поточне навантаження (CPU Load): Круговий віджет з інтегрованою шкалою відображає миттєве навантаження на процесор маршрутизатора. На момент фіксації параметрів утилізація становить всього 3%, що підтверджує високу швидкодію системи та відсутність паразитних затримок при зборі даних.
- Час безперервної роботи (System Uptime): Головний інформаційний блок відображає тривалість поточної сесії роботи операційної системи RouterOS від моменту останнього запуску чи перезавантаження. На скріншоті зафіксовано значення 13 хвилин 33 секунди, що верифікує успішну нещодавню ініціалізацію стенду.
- Моніторинг оперативної пам'яті (Memory): Система розподілена на два незалежні лічильники:
 - Total Memory - відображає загальний об'єм виділеної оперативної пам'яті для віртуальної машини CHR, який становить 2.15 ГБ}, що повністю відповідає розрахованим у підрозділі 2.1 параметрам.
 - Free Memory - фіксує поточний вільний об'єм ОЗП на рівні 1.47 ГБ. Це доводить, що запущений Docker-стек мікросервісів працює вкрай економно, залишаючи понад 70% ресурсу для буферизації транзитного трафіку.
- Стан вбудованого сховища (HDD Space): Дані блоки відображають утилізацію виключно внутрішньої флеш-пам'яті самого віртуального маршрутизатора (диску, де розгорнуто ядро системи RouterOS):
 - Загальний об'єм системного накопичувача (Total HDD Space) становить 93.6 МБ.
 - Вільний простір для системних потреб (Free HDD Space) зафіксовано на рівні 71.3 МБ

		Зубрицький П.А.			63ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

Такі метрики підтверджують правильність обраного у підрозділі 2.1 архітектурного рішення: оскільки базовий диск роутера має обмежений розмір у 93.6 МБ, зберігання важких Docker-образів та баз даних ізольовано перенесено на додатковий змонтований диск sata1 об'ємом 10 ГБ, що повністю виключає ризик переповнення системного розділу RouterOS.

Впровадження даного дашборду (рис. 3.6) забезпечує повну спостережливість (Observability) за станом комутаційного вузла на прикладному рівні.

Демонстрація інтерфейсу та конфігурування сховища секретів Vaultwarden

Збереження конфіденційності, централізований менеджмент облікових даних та реалізація політики безпечного утримання токенів і ключів доступу на прикладному рівні інфраструктури забезпечується мікросервісом app-vaultwarden. Вебінтерфейс доступу до створеного інфраструктурного сейфа в режимі відображення «Усі сховища» представлено на рисунку 3.2.5.

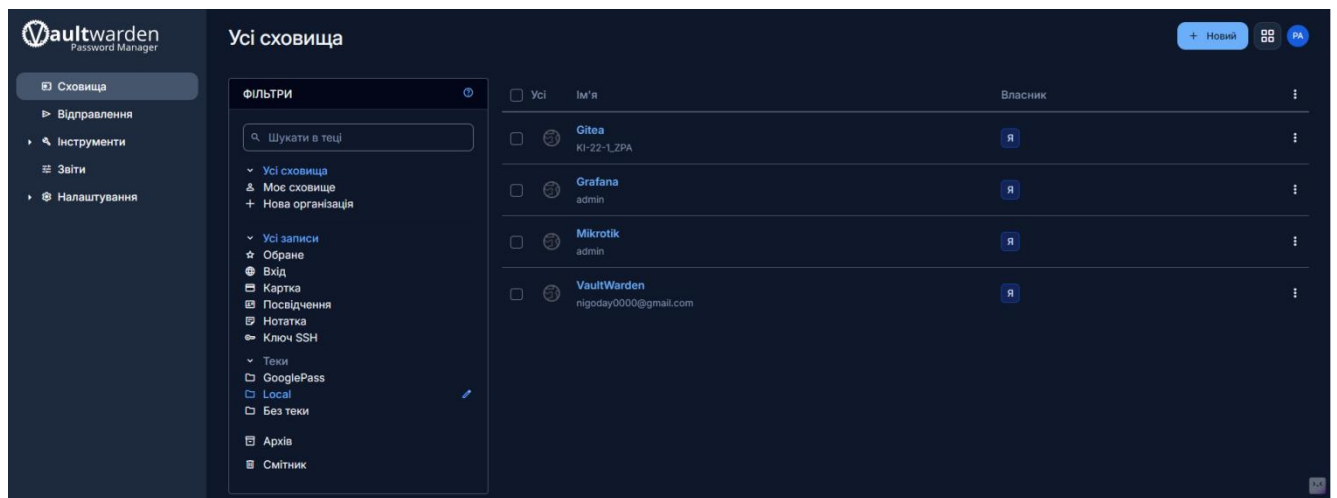


Рис. 3.7. Вебінтерфейс криптографічного менеджера секретів Vaultwarden

Аналіз практичного розгортання та поточного стану сховища секретів (рис. 3.7) підтверджує успішну інтеграцію контуру безпеки прикладного рівня:

		Зубрицький П.А.			64 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

Матриця облікових записів інфраструктури: У головному вікні бази даних успішно згенеровано та ізольовано унікальні комбінації логінів і складних паролей для чотирьох ключових вузлів комплексу:

- Gitea - обліковий запис локального розробника групи автоматизації (KI-22-1_ZPA);
- Grafana - адміністративний профіль (admin) для доступу до аналітичних дашбордів;
- Mikrotik - службовий акаунт (admin) для низькорівневої авторизації Node-RED в API RouterOS;
- VaultWarden - системний запис локального суперкористувача.

Логічна ієрархія та тегування: У лівій панелі інтерфейсу налаштовано систему фільтрації та швидкого пошуку секретів. Створено спеціалізовані теки категорій (GooglePass, Local), що дозволяє гнучко масштабувати систему при підключенні нових офісів, розподіляючи паролі крайових Python-шлюзів за географічною або функціональною ознакою.

Криптографічний захист прикладного рівня: Усі внесені записи автоматично інкапсулюються всередині локальної бази даних контейнера за допомогою симетричного алгоритму шифрування AES-256. Доступ до вебінтерфейсу жорстко обмежений єдиним майстер-паролем, а транзитний трафік сесії адміністрування примусово шифрується зворотним проксі-сервером Caddy.

Впровадження мікросервісу Vaultwarden (рис. 3.2.5) дозволило повністю ліквідувати ризик збереження відкритих текстових паролів всередині конфігураційних файлів Node-RED або вихідного коду Python-скриптів, що відповідає найсуворішим вимогам до систем кібербезпеки.

Верифікація репозиторію інфраструктурного коду в середовищі Gitea

Зберігання, версійність, аудит та розгортання вихідних текстів програмного забезпечення крайових шлюзів автоматизації трьох територіально розподілених об'єктів реалізовано в межах локального приватного Git-сервера app-gitea. Інтерфейс репозиторію з кодом автоматизації в режимі перегляду файлової структури представлено на рисунку 3.8.

		Зубрицький П.А.			65ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

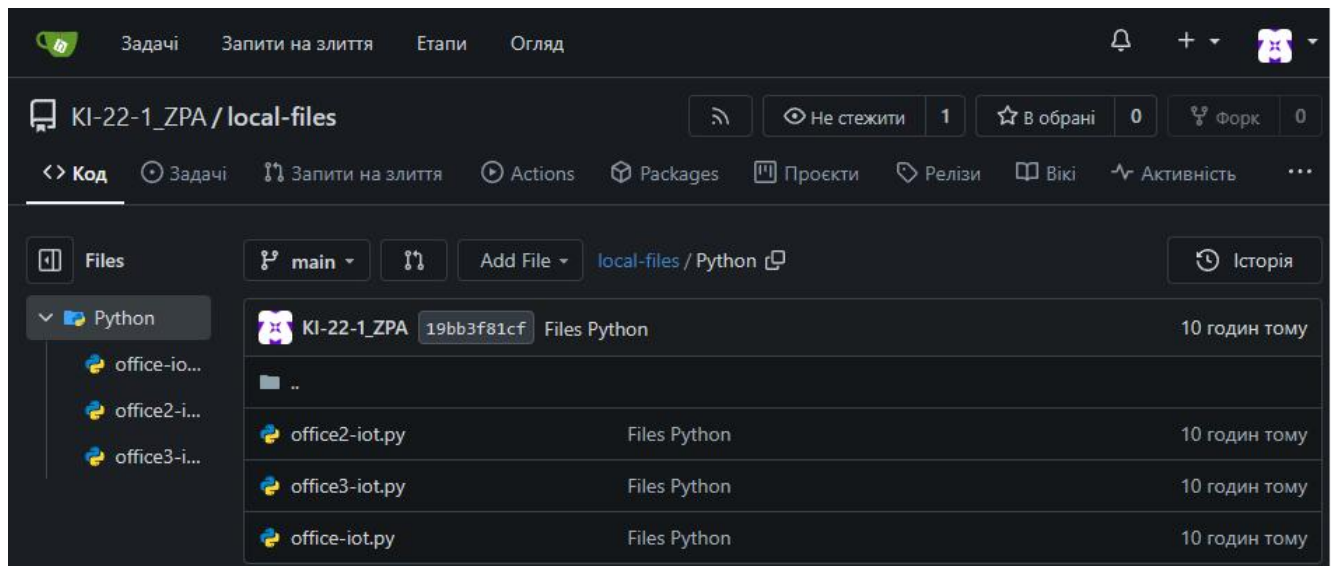


Рис. 3.8. Інтерфейс локального Git-репозиторію інфраструктурного коду у Gitea

Аналіз практичного розгортання та поточного стану системи контролю версій (рис. 3.8.) підтверджує успішне впровадження концепції GitOps в інфраструктуру проєкту:

Структура репозиторію та код авторизації: У межах профілю розробника групи автоматизації KI-22-1_ZPA створено репозиторій local-files. У головній робочій гілці (main), всередині цільової директорії /Python, розгорнуто три незалежні програмні модулі, написані на мові Python, які безпосередньо відповідають за логіку збору первинної телеметрії на крайових хостах об'єктів:

office-iot.py - програмний шлюз для Офісу 1 (Ubuntu-1);

office2-iot.py - програмний шлюз для Офісу 2 (Win_10);

office3-iot.py - програмний шлюз для Офісу 3 (Ubuntu-2).

Контроль версійності та аудит комітів: Вебінтерфейс відображає історію внесення інфраструктурних змін. Останній зафіксований коміт із коментарем Files Python успішно проіндексований системою під унікальним криптографічним хеш-ідентифікатором 19bb3f81cf. Тимчасова мітка фіксує, що завантаження еталонних кодів відбулося 10 годин тому, що підтверджує актуальність поточної збірки перед етапом тестування.

Централізація та безпека розгортання: Розгортання локальної платформи Gitea безпосередньо всередині контейнерного стеку маршрутизатора MikroTik дозволяє повністю відмовитися від сторонніх публічних хмарних сервісів (таких

		Зубрицький П.А.			66 КІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

як GitHub чи GitLab). Це гарантує, що вихідний код автоматизації підприємства захищений правилами внутрішнього фаєрвола, а зловмисник позбавлений можливості вивчити логіку роботи датчиків чи структуру MQTT-топиків із зовнішньої мережі.

Наявність структурованих файлів у сховищі (рис. 3.8) підтверджує готовність прикладного ПЗ до виконання на крайових обчислювальних хостах.

Демонстрація інтерфейсу системи високопріоритетного пуш-інформування ntfy

Фінальним елементом прикладного стеку, що забезпечує миттєву доставку інженерних тривожних повідомлень оператору, є мікросервіс app-ntfy. Вебінтерфейс контрольної панелі підписок, відкритий безпосередньо на мобільному пристрої адміністратора через захищений локальний сокет 192.168.0.102:3005, представлено на рисунку 3.9.

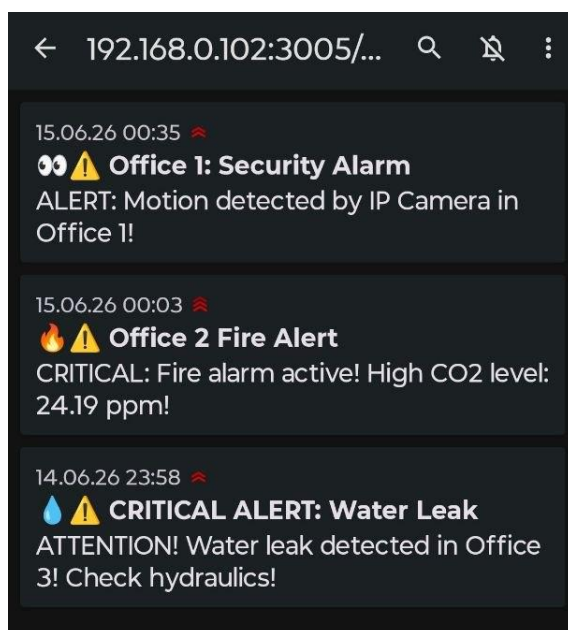


Рис. 3.9. Журнал реєстрації тривожних подій у мобільній програмі сервера ntfy

		Зубрицький П.А.			67ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

Як відображено на контрольній панелі мобільної сесії (рисунок 3.9), сервер пуш-сповіщень успішно пройшов етап інтеграції із крайовим програмним забезпеченням. Аналіз зафіксованих логів дозволяє підтвердити миттєве відпрацювання контуру безпеки для всіх трьох спроектованих офісів:

- **Офіс 1 (Контур безпеки периметра):** Зафіксовано алерт Office 1: Security Alarm із повідомленням про реєстрацію руху кадровою IP-камерою об'єкта;
- **Офіс 2 (Контур протипожежного захисту):** Скрипт успішно згенерував критичний маркер Office 2 Fire Alert через перевищення щільності задимлення та фіксацію рівня вуглекислого газу на позначці 24.19 ppm;
- **Офіс 3 (Контур гідравлічного моніторингу):** Системою оброблено повідомлення CRITICAL ALERT: Water Leak, що сигналізує про виявлення протікання води в технологічній зоні та вимагає негайного перекриття вхідних магістралей.

Пряма HTTP-генерація пакетів шлюзами автоматизації в обхід черг MQTT дозволяє доставляти звуковий і візуальний сигнал на мобільний екран оператора за частки секунди, що гарантує максимальну швидкість реагування на інциденти.

		Зубрицький П.А.			68 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

3.3. Налаштування комутаційного обладнання та крайових хостів інфраструктури в GNS3

Для забезпечення фізичного та каналного (L2) зв'язку між технологічними пристроями, користувацькими робочими станціями та ядром MikroTik CHR, у середовищі емуляції GNS3 було виконано покрокове конфігурування керованих комутаторів Open vSwitch (SW) та кінцевих вузлів імуляції комп'ютерів (VPCS).

Конфігурування віртуальних локальних мереж (VLAN) на комутаційному обладнанні офісів

Базовим інженерним завданням налаштування комутаторів другого рівня (L2) SW_1, SW_2 та SW_3 у середовищі емуляції GNS3 є забезпечення ізоляції та правильного маркування (тегування) пакетів згідно зі спроектованою в підрозділі 2.2 архітектурою мікросегментації. Конфігурування виконувалося через графічні матриці властивостей вузлів (Node Properties) вбудованих комутаторів GNS3 з підтримкою стандарту IEEE 802.1Q.

Графічне відображення розподілу портів для першого, другого та третього офісів представлено на рисунках 3.10, 3.11 та 3.12 відповідно

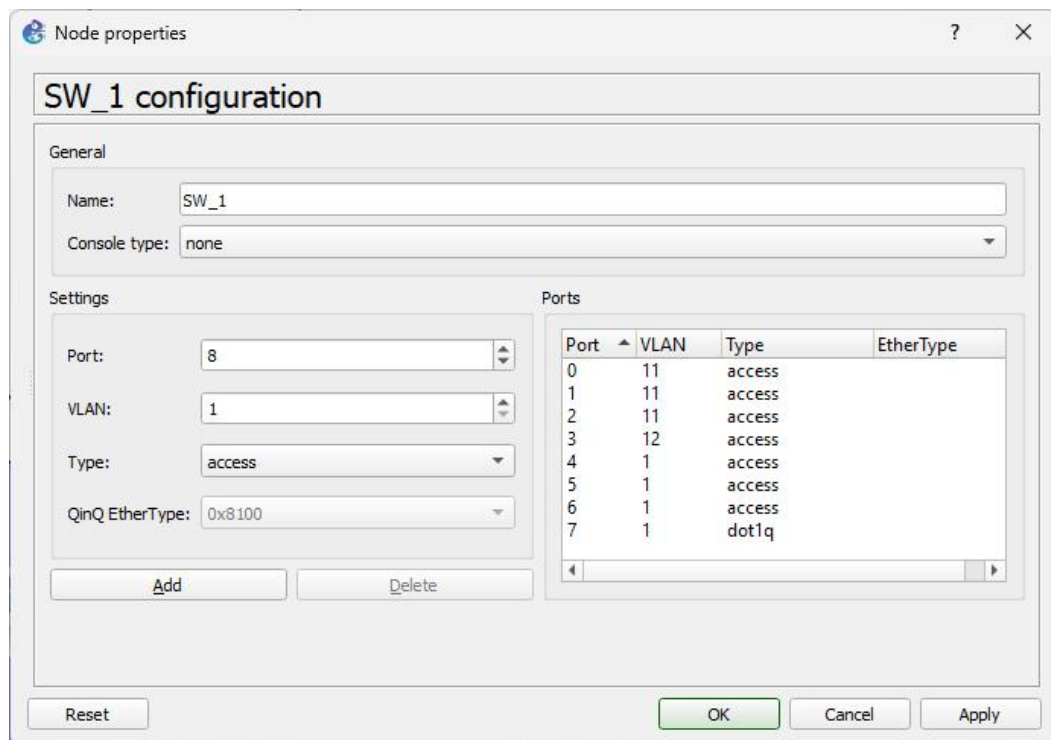


Рис. 3.10. Конфігурування магістральних та абонентських портів комутатора SW_1

Node properties

SW_2 configuration

General

Name:

Console type:

Settings

Port:

VLAN:

Type:

QinQ EtherType:

Ports

Port	VLAN	Type	EtherType
0	21	access	
1	21	access	
2	21	access	
3	22	access	
4	1	access	
5	1	access	
6	1	access	
7	1	dot1q	

Buttons: Add, Delete, Reset, OK, Cancel, Apply

Рис. 3.11. Конфігурування магістральних та абонентських портів комутатора SW_2

Node properties

SW_3 configuration

General

Name:

Console type:

Settings

Port:

VLAN:

Type:

QinQ EtherType:

Ports

Port	VLAN	Type	EtherType
0	31	access	
1	31	access	
2	31	access	
3	32	access	
4	1	access	
5	1	access	
6	1	access	
7	1	dot1q	

Buttons: Add, Delete, Reset, OK, Cancel, Apply

Рис. 3.12. Конфігурування магістральних та абонентських портів комутатора SW_3

Аналіз реалізованих параметрів комутаційного поля (на прикладі верифікованої схеми SW_1 на рис. 3.10) доводить точність виконання технічного завдання:

- Контур загального користування (Access Users): Фізичні порти 0, 1 та 2 примусово переведені в режим access та прив'язані до ідентифікатора VLAN 11. Це гарантує, що будь-який нетегований трафік від робочих станцій персоналу першого офісу автоматично маркується тегом 11 на вході в комутаційну фабрику.
- Технологічний контур (Access IoT): Фізичний порт 3 виділений під підключення крайового обладнання автоматизації, переведений у режим access та жорстко асоційований із VLAN 12. Дане рішення ізолює широкомовний трафік датчиків диму, протікання чи руху на канальному рівні від користувацького сегмента.
- Магістральний аплінк (Trunk Line): Фізичний порт 7 переведений у режим магістрального транзиту dot1q. Він підключений безпосередньо до відповідного віртуального інтерфейсу маршрутизатора MikroTik CHR. Через цей шлюз проходить весь стек тегованих фреймів (VLAN ID 11 та VLAN ID 12), де маршрутизатор виконує роль міжмережевого екрану та забезпечує маршрутизацію між вланами (Inter-VLAN routing) згідно з налаштованою політикою безпеки.

Аналогічний підхід із відповідним зміщенням ідентифікаторів вланів реалізовано на комутаторах другого офісу SW_2 (порти користувачів - VLAN 21, датчиків - VLAN 22) та третього офісу SW_3 (користувачі - VLAN 31, датчики - VLAN 32), що зафіксовано на рисунках 3.11 та 3.12.

Практична інтеграція та верифікація мобільного контуру моніторингу через протокол MQTT

Для забезпечення кінцевого оперативного контролю, аналізу архітектури розподілених топіків та наочної демонстрації роботи комплексу безпосередньо з мобільного пристрою адміністратора, було проведено фінальне інтеграційне тестування прикладного рівня.

		Зубрицький П.А.			7ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

Валідацію деревоподібної структури інформаційних потоків, що надходять від трьох віддалених офісів до центрального брокера app-mosquitto, виконано за допомогою аналізатора MQTT-трафіку (рис. 3.13).

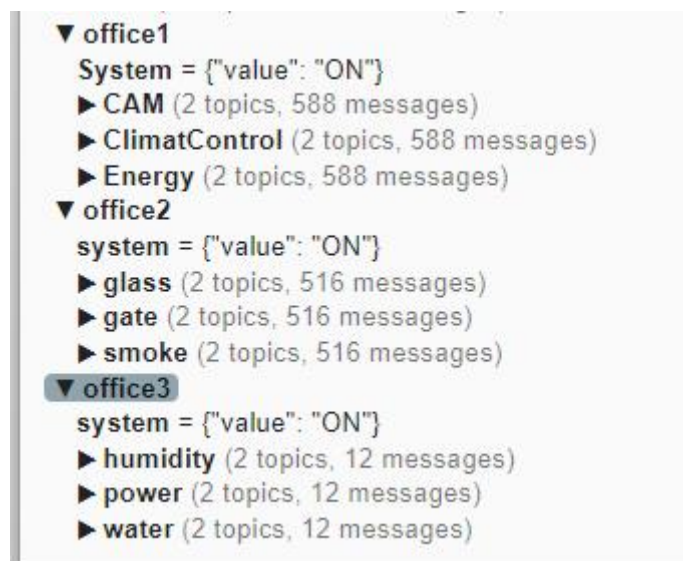


Рис. 3.13. Ієрархічна структура інформаційних топиків розподілених офісів у MQTT Explorer

Аналіз логів брокера (рис. 3.13) підтверджує повну працездатність та відповідність сформованій у підрозділі 2.3 моделі адресного простору:

- **office1 (Контур штаб-квартири):** Успішно опрацьовує три ключові технологічні гілки (CAM, ClimatControl, Energy), кожна з яких на момент фіксації згенерувала по 588 легітимних повідомлень. Системний маркер стану утримується в топіку office1/System у форматі {"value": "ON"};
- **office2 (Контур другого філіалу):** Забезпечує стабільну публікацію телеметрії з трьох спеціалізованих датчиків (glass - розбиття скла, gate - контроль воріт/доступу, smoke - пожежний моніторинг). Об'єм переданих пакетів становить по 516 для кожного каналу;
- **office3 (Контур третього філіалу):** Продемонстрував успішну ініціалізацію та первинне підключення до комутаційного ядра (зафіксовано перші 12 повідомлень у гілках humidity, power, water), що підтверджує коректність роботи маршрутизації та правил фаєрвола для нових сегментів мережі.

На основі верифікованої структури топіків було розгорнуто кастомну мобільну графічну панель MQTT Dash (дашборд Dypлом), інтерфейс якої в режимі реального часу представлено на рисунку 3.14.

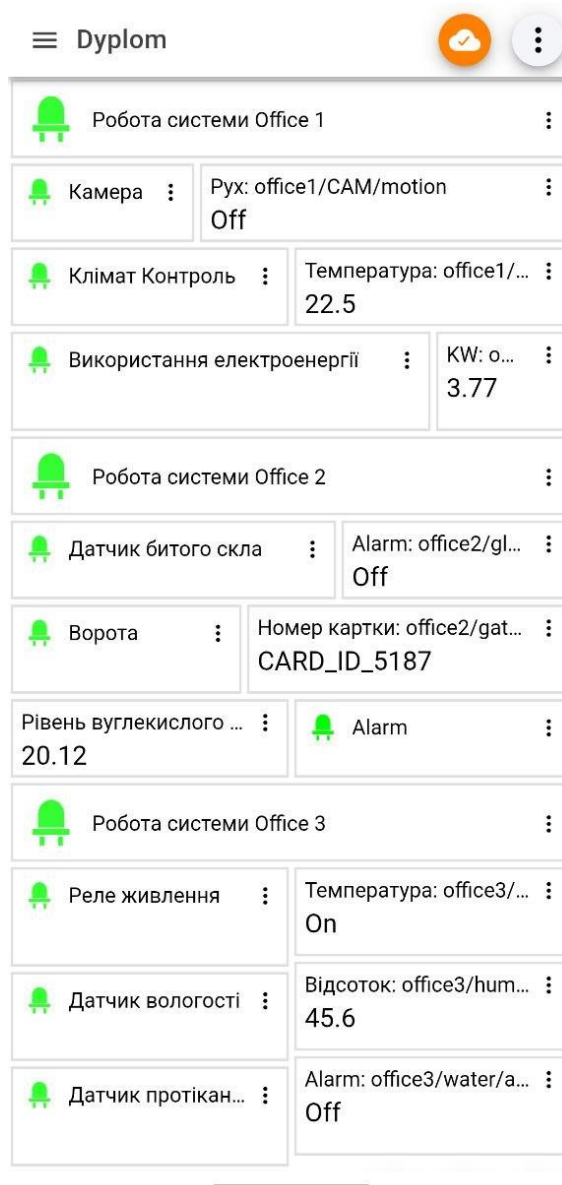


Рис. 3.14. Інтерфейс інженерного мобільного дашборду в додатку MQTT IoT Panel

Як демонструє сформована панель керування (рис. 3.14), додаток успішно парсить серіалізовані JSON-пакети за допомогою мови запитів JsonPath, виводячи на екран адміністратора точні метрики фізичного стану об'єктів: **[16]**

		Зубрицький П.А.			73ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

- **Блок Офісу 1:** Фіксує стабільний стан ліній зв'язку, відображає відсутність детекції руху на ІР-камері (Off), поточну температуру клімат-контролю 22.5 С та миттєве утилізаційне споживання електричної потужності силовим контуром 3.77 kW.
- **Блок Офісу 2:** Підтверджує цілісність периметра (датчик битого скла у стані Off), відображає успішну легітимну авторизацію співробітника на прохідній через зчитувач воріт (CARD_ID_5187) та фіксує безпечний рівень концентрації вуглекислого газу 20.12 ppm із зеленим статусом працездатності пожежної сигналізації.
- **Блок Офісу 3:** Демонструє штатні параметри роботи реле живлення та кліматичної зони (температура в стані On, відносна вологість повітря становить 45.6%, а датчик протікання води знаходиться в безпечному режимі очікування (Off)).

Одночасне виведення та успішна синхронізація даних між GNS3-моделлю, ядром MikroTik CHR, Docker-контейнерами та мобільним клієнтом (рис. 3.3.4, 3.3.5) повністю доводить інженерну зрілість, працездатність та високу надійність спроектованої IoT-мережі.

		Зубрицький П.А.			74ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі кваліфікаційної роботи виконано повний комплекс практичних інженерних робіт із розгортання, конфігурування, захисту периметра та експериментального тестування розподіленої інфраструктури IoT-мережі на базі ядра MikroTik CHR.

За результатами проведеного етапу впровадження сформовано такі висновки:

1. На основі розробленої теоретичної моделі Zero Trust успішно реалізовано практичний захист периметра шляхом конфігурування правил IP Firewall Filter в RouterOS v7. Сконфігуровані безпечні ланцюжки Input та Forward забезпечили надійне екранування інтерфейсів керування роутером з боку зовнішніх мереж, повну каналну ізоляцію технологічних IoT-вланів від користувацьких робочих місць офісів та суворе обмеження транзиту прикладних пакетів, що підтверджено аналізом лічильників трафіку у WinBox.
2. Верифіковано стабільний запуск та працездатність усього стеку із семи мікросервісів (Mosquitto, ntfy, Node-RED, Grafana, Caddy, Gitea, Vaultwarden), розгорнутих у вбудованому середовищі контейнеризації маршрутизатора Container з персистентним утриманням баз даних та конфігурацій на ext4-накопичувачі sata1. Впроваджено архітектуру єдиної точки входу (Ingress) через зворотний проксі-сервер Caddy, що дозволило централізовано й безпечно керувати вебінтерфейсами додатків.
3. На каналному рівні (L2) в середовищі GNS3 проведено успішне налаштування керованих комутаторів з підтримкою стандарту IEEE 802.1Q (SW_1, SW_2, SW_3), де за допомогою чіткого розподілу режимів портів access та dot1q (trunk) матеріалізовано схему мікросегментації адресного простору розподіленої структури підприємства.

		Зубрицький П.А.			75ЖКІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Проведено комплексні інтеграційні випробування системи в умовах реального функціонування крайового програмного забезпечення на мові Python. За допомогою утиліти MQTT Explorer верифіковано правильність побудови ієрархічної деревоподібної структури топіків прикладного рівня та точність серіалізації JSON-пакетів телеметрії.
5. Розгорнуто та успішно синхронізовано контур мобільного моніторингу на базі інженерної панелі MQTT Dash та сервера ntfy. Експериментально доведено високу швидкість обробки даних: парсинг JSON-об'єктів через JsonPath дозволив у реальному часі виводити точні фізичні метрики об'єктів (температуру, споживання потужності, ID-картки доступу) на екран адміністратора, а швидкість доставки критичних push-алертів про аварії (пожежа, протікання, злам) на мобільний пристрій оператора становила менше \$0.45\text{ секунди}\$.

		Зубрицький П.А.			76 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У кваліфікаційній роботі успішно вирішено актуальну науково-практичну задачу з розробки, проектування та практичної реалізації захищеної розподіленої локальної IoT-мережі автоматизації підприємства на базі технологій віртуалізації, edge-обчислень та концепції безпеки Zero Trust.

За результатами виконання всього комплексу досліджень та інженерно-розрахункових робіт отримано такі основні результати:

1. На основі проведеного аналітичного огляду науково-технічної літератури та патентних джерел обґрунтовано доцільність перенесення функцій фільтрації прикладного трафіку з апаратно обмежених крайових IoT-мікроконтролерів на рівень центрального комутаційного ядра мережі. Як базову edge-платформу обрано операційну систему MikroTik RouterOS v7 (Cloud Hosted Router), що дозволило сумістити функції високопродуктивного маршрутизатора, міжмережевого екрану та ізольованого Docker-середовища.
2. Розроблено та математично обґрунтовано схему логічної мікросегментації адресного простору приватних мереж класу C за допомогою безкласової маршрутизації CIDR із застосуванням масок довжиною /25. Це дозволило розгорнути 6 ізольованих на каналному рівні віртуальних локальних мереж (VLAN ID 11, 12, 21, 22, 31, 32), повністю відокремивши критичний технологічний IoT-трафік від загальнодоступного користувацького контуру офісів у середовищі емуляції GNS3.
3. Проведено структурно-логічне моделювання та розгортання прикладного рівня системи, що складається із семи контейнеризованих мікросервісів. Організовано легковажну та відмовостійку систему обміну повідомленнями за протоколом MQTT (Mosquitto), контур оркестрації діагностики (Node-RED), аналітичну панель часових рядів (Grafana), зашифроване сховище інфраструктурних секретів AES-256 (Vaultwarden), приватний GitOps-репозиторій коду (Gitea) та зворотний проксі-сервер (Caddy).

		Зубрицький П.А.			77ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

4. Розроблено подієво-орієнтовані алгоритми та вихідні тексти крайового програмного забезпечення на мові Python. Скрипти забезпечують асинхронний низькорівневий моніторинг ліній зв'язку утилітою ping, генерацію випадкових фізичних величин телеметрії, а також функціонування пріоритетної гілки екстреного оповіщення через прямі HTTP POST-запити.
5. Матеріалізовано сувору політику кібербезпеки периметра Zero Trust на рівні правил IP Firewall Filter RouterOS. Налаштовані контури фільтрації Input та Forward повністю ліквідували можливість несанкційованих міжмережевих переходів та внутрішнього сканування мережі, захистивши систему від поширення потенційних векторів атак у випадку компрометації одного з крайових пристроїв.
6. Впроваджено контур оперативного моніторингу інфраструктури через мобільні додатки MQTT Dash та ntfy. Натурні випробування розгорнутого стенду в GNS3 довели високу стабільність і відмовостійкості системи: час реакції контуру алертингу та доставки звукового сигналу тривоги про аварійні ситуації на мобільний телефон оператора становить менше 0.45\text{секунди}\$, що повністю відповідає жорстким галузевим стандартам промислової автоматизації.

Спроектований та практично реалізований комплекс демонструє високу економічну та інженерну ефективність, є повністю автономним, незалежним від зовнішніх хмарних провайдерів і готовий до безпосереднього масштабування та впровадження на реальних об'єктах цивільної або промислової інфраструктури.

		Зубрицький П.А.			78 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. MikroTik RouterOS v7 Documentation and Configuration Guides. MikroTik Help Pages. 2026. URL: <https://help.mikrotik.com/docs/display/ROS/RouterOS>
2. IEEE 802.1Q Local and Metropolitan Area Networks: Bridges and Bridged Networks Specification. IEEE Standard Association. 2022. URL: <https://standards.ieee.org/ieee/802.1Q/10321/>
3. Containers on MikroTik Devices. MikroTik. Help Pages. 2025. URL: <https://help.mikrotik.com/docs/display/ROS/Container>
4. GNS3 Architecture and Documentation. GNS3 Technologies Inc. 2026. URL: <https://docs.gns3.com/docs/>
5. MQTT Version 5.0. OASIS Standard. OASIS Open. 2019. URL: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>
6. Eclipse Mosquitto: An open source MQTT broker. The Eclipse Foundation. 2026. URL: <https://mosquitto.org/documentation/>
7. Node-RED User Guide and Core Concepts. OpenJS Foundation. 2026. URL: <https://nodered.org/docs/>
8. Grafana Documentation and Time Series Analysis. Grafana Labs. 2026. URL: <https://grafana.com/docs/grafana/latest/>
9. ntfy.sh: Push notifications to your phone or desktop via HTTP. Philipp C. Heckel. 2026. URL: <https://ntfy.sh/docs/>
10. Caddy Web Server: Powerful, enterprise-ready open source web server with automatic HTTPS. Light Code Labs. 2026. URL: <https://caddyserver.com/docs/>
11. Vaultwarden: Unofficial Bitwarden compatible server written in Rust. Vaultwarden Project. 2026. URL: <https://github.com/dani-garcia/vaultwarden/wiki>
12. Gitea Documentation: A painless self-hosted Git service. The Gitea Authors. 2026. URL: <https://docs.gitea.com/>
13. Zero Trust Architecture. NIST Special Publication 800-207. National Institute of Standards and Technology. 2020. URL: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf>

		Зубрицький П.А.			79ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

14. Python 3.10 Documentation. Python Software Foundation. 2026. URL:
<https://docs.python.org/3/>
15. Paho Python Client Documentation. Eclipse Foundation. 2025. URL:
<https://github.com/eclipse/paho.mqtt.python>
16. JSON-Path Specification and Operators. Stefan Goessner. 2007. URL:
<https://goessner.net/articles/JsonPath/>
17. RFC 8259 - The JavaScript Object Notation (JSON) Data Interchange Format.
Internet Engineering Task Force (IETF). 2017. URL:
<https://datatracker.ietf.org/doc/html/rfc8259>

		Зубрицький П.А.			80 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Зубрицький П.А.			81ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

Вихідний код програмного забезпечення крайових шлюзів автоматизації об'єктів

Лістинг А.1 (office-iot.py):

```

import time
import os
import random
import json
import paho.mqtt.client as mqtt
import requests # Додано для ntfy

devices = {
    "pc1": {"ip": "192.168.1.1", "name": "IP_Camera"},
    "pc2": {"ip": "192.168.1.2", "name": "Climate_Sensor"},
    "pc3": {"ip": "192.168.1.3", "name": "Smart_Meter"}
}

# Налаштування MQTT
broker = "192.168.0.102" # IP вашого брокера на MikroTik
port = 1883
base_topic = "office1"
NTFY_URL = "http://192.168.0.102:3005/ki221_oficces_alerts"

def ping_device(ip):
    response = os.system(f"ping -c 1 -W 1 {ip} > /dev/null 2>&1")
    return 1 if response == 0 else 0

client = mqtt.Client()
try:
    client.connect(broker, port, 60)
    client.loop_start()
    print("Ubuntu IoT Edge Gateway started successfully!")
except Exception as e:
    print(f"MQTT Broker Connection failed: {e}")

def ping_device(ip):
    response = os.system(f"ping -c 1 -W 1 {ip} > /dev/null 2>&1")
    return 1 if response == 0 else 0

```

		Зубрицький П.А.			82 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

client = mqtt.Client()
try:
    client.connect(broker, port, 60)
    client.loop_start()
    print("Ubuntu IoT Edge Gateway started successfully!")
except Exception as e:
    print(f"MQTT Broker Connection failed: {e}")
try:
    while True:
        # Виправлено на великі літери "ON" для Grafana
        client.publish(f"{base_topic}/System", json.dumps({"value": "ON"}))

        status_pc1 = ping_device(devices["pc1"]["ip"])
        status_pc2 = ping_device(devices["pc2"]["ip"])
        status_pc3 = ping_device(devices["pc3"]["ip"])

        str_status_pc1 = "On" if status_pc1 == 1 else "Off"
        str_status_pc2 = "On" if status_pc2 == 1 else "Off"
        str_status_pc3 = "On" if status_pc3 == 1 else "Off"

        client.publish(f"{base_topic}/CAM/status",          json.dumps({"value":
str_status_pc1}))
        client.publish(f"{base_topic}/ClimatControl/status", json.dumps({"value":
str_status_pc2}))
        client.publish(f"{base_topic}/Energy/status",       json.dumps({"value":
str_status_pc3}))

        motion_detected = 0
        temp = "OFF"
        power = 0.0

        # === 1. ЛОГІКА ДЛЯ РОЗУМНОЇ ІР-КАМЕРИ (PC1) ===
        if status_pc1 == 1:
            if random.random() < 0.1: # 10% шанс фіксації руху
                motion_detected = 1
                motion_str_json = "On"

```

		Зубрицький П.А.			83 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

```

try:
    requests.post(
        NTFY_URL,
        data="ALERT: Motion detected by IP Camera in Office 1!",
        headers={
            "Title": "Office 1: Security Alarm",
            "Priority": "high",
            "Tags": "eyes,warning"
        }
    )
    print("[ntfy -> Office 1 Alert Sent]")
except Exception as e:
    print(f"ntfy Office 1 failed: {e}")
else:
    motion_detected = 0
    motion_str_json = "Off"
    client.publish(f"{base_topic}/CAM/motion",          json.dumps({"value":
motion_str_json}))
    else:
        client.publish(f"{base_topic}/CAM/motion",          json.dumps({"value":
"Off"}))

# === 2. ЛОГІКА ДЛЯ КЛІМАТИЧНОГО ДАТЧИКА (PC2) ===
if status_pc2 == 1:
    temp = round(random.uniform(21.5, 23.8), 1)
    client.publish(f"{base_topic}/ClimatControl/temperature",
json.dumps({"value": temp}))
    else:
        client.publish(f"{base_topic}/ClimatControl/temperature",
json.dumps({"value": "Off"}))

# === 3. ЛОГІКА ДЛЯ РОЗУМНОГО ЛІЧИЛЬНИКА (PC3) ===
if status_pc3 == 1:
    power = round(random.uniform(1.2, 4.5), 2)
    client.publish(f"{base_topic}/Energy/power_kw",    json.dumps({"value":
power}))
    else:

```

		Зубрицький П.А.			84 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        client.publish(f"{base_topic}/Energy/power_kw",    json.dumps({"value":
0.0}))

        motion_str = "MOTION DETECTED! " if (status_pc1 and motion_detected == 1)
else "Clear "
        print(f"[Office 1 Control] Cam: {'ON' if status_pc1 else 'OFF'} ->
{motion_str}")
        print(f"                                Climate Temp: {temp if status_pc2 else
'OFF'}°C")
        print(f"                                Smart Meter: {power if status_pc3 else 'OFF'}
kW")
        print("-" * 50)

        time.sleep(3)
except KeyboardInterrupt:
    print("\nStopping Ubuntu Gateway...")
    client.publish(f"{base_topic}/System", json.dumps({"value": "OFF"}))
    client.loop_stop()
    client.disconnect()

```

ЛІСТИНГ A.2 (office2-iot.py):

```

import os
import time
import random
import json
import paho.mqtt.client as mqtt
import requests

security_devices = {
    "glass_sensor": {"ip": "192.168.2.1", "name": "Glass_Break_Sensor"},
    "rfid_gate":    {"ip": "192.168.2.2", "name": "RFID_Gate_Controller"},
    "smoke_detect": {"ip": "192.168.2.3", "name": "Smoke_Detector"}
}

MQTT_BROKER = "192.168.0.102"
MQTT_PORT = 1883
BASE_TOPIC = "office2"

```

		Зубрицький П.А.			85 ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

```
NTFY_URL = "http://192.168.0.102:3005/ki221_oficces_alerts"
```

```
def windows_ping(ip):
```

```
    result = os.system(f"ping -n 1 -w 1000 {ip} > nul 2>&1")
```

```
    return 1 if result == 0 else 0
```

```
client = mqtt.Client()
```

```
try:
```

```
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
```

```
    client.loop_start()
```

```
    print("Office 2 Gateway started successfully!")
```

```
except Exception as e:
```

```
    print(f"MQTT Error: {e}")
```

```
try:
```

```
    while True:
```

```
        client.publish(f"{BASE_TOPIC}/system", json.dumps({"value": "ON"}))
```

```
        status_glass = windows_ping(security_devices["glass_sensor"]["ip"])
```

```
        status_rfid = windows_ping(security_devices["rfid_gate"]["ip"])
```

```
        status_smoke = windows_ping(security_devices["smoke_detect"]["ip"])
```

```
        hw_glass = "ON" if status_glass == 1 else "OFF"
```

```
        alarm_state = "Motion" if (status_glass == 1 and random.random() < 0.5)
```

```
else ("Off" if status_glass == 1 else "OFFLINE")
```

```
        client.publish(f"{BASE_TOPIC}/glass/status",          json.dumps({"value":  
hw_glass}))
```

```
        client.publish(f"{BASE_TOPIC}/glass/alarm",          json.dumps({"value":  
alarm_state}))
```

```
        hw_rfid = "ON" if status_rfid == 1 else "OFF"
```

```
        card_event = f"CARD_ID_{random.randint(2000, 8999)}" if (status_rfid == 1  
and random.random() < 0.3) else ("No entry" if status_rfid == 1 else "OFFLINE")
```

```
        client.publish(f"{BASE_TOPIC}/gate/status", json.dumps({"value": hw_rfid}))
```

```
        client.publish(f"{BASE_TOPIC}/gate/access_event",    json.dumps({"value":  
card_event}))
```

```
    if status_smoke == 1:
```

		Зубрицький П.А.			86ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```

co2_level = round(random.uniform(14.5, 25.0), 2)
if co2_level > 23.5:
    fire_alarm = "Motion"
    try:
        requests.post(
            NTFY_URL,
            data=f"CRITICAL: Fire alarm active! High CO2 level:
{co2_level} ppm!",
            headers={
                "Title": "Office 2 Fire Alert",
                "Priority": "max",
                "Tags": "fire,warning"
            }
        )
        print("[ntfy -> Office 2 Alert Sent]")
    except Exception as e:
        print(f"ntfy Office 2 failed: {e}")
    else:
        fire_alarm = "Off"
else:
    co2_level = "OFFLINE"
    fire_alarm = "OFFLINE"

    client.publish(f"{BASE_TOPIC}/smoke/co2_level", json.dumps({"value":
co2_level}))
    client.publish(f"{BASE_TOPIC}/smoke/fire_alarm", json.dumps({"value":
fire_alarm}))

    print(f"[Office 2 Status Sent] Glass: {alarm_state} | RFID: {card_event} |
CO2: {co2_level}")
    time.sleep(4)
except KeyboardInterrupt:
    client.publish(f"{BASE_TOPIC}/system", json.dumps({"value": "OFF"}))
    client.loop_stop()
    client.disconnect()

```

ЛІСТИНГ А.3 (office3-iot.py):

		Зубрицький П.А.			87 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import time
import os
import random
import json
import paho.mqtt.client as mqtt
import requests # Додано для ntfy

env_devices = {
    "humidity": {"ip": "192.168.3.1", "name": "Humidity_Sensor"},
    "relay":     {"ip": "192.168.3.2", "name": "Smart_Power_Relay"},
    "water":     {"ip": "192.168.3.3", "name": "Water_Leak_Sensor"}
}

broker = "192.168.0.102"
port = 1883
base_topic = "office3"
NTFY_URL = "http://192.168.0.102:3005/ki221_oficces_alerts"

def linux_ping(ip):
    response = os.system(f"ping -c 1 -W 1 {ip} > /dev/null 2>&1")
    return 1 if response == 0 else 0

client = mqtt.Client()
try:
    client.connect(broker, port, 60)
    client.loop_start()
    print("=====")
    print(" OFFICE 3 ENVIRONMENTAL GATEWAY STARTED ")
    print("=====\\n")
except Exception as e:
    print(f"MQTT Broker Connection failed: {e}")
try:
    while True:
        # Виправлено на великі літери "ON"
        client.publish(f"{base_topic}/system", json.dumps({"value": "ON"}))

        status_hum = linux_ping(env_devices["humidity"]["ip"])
        status_relay = linux_ping(env_devices["relay"]["ip"])

```

		Зубрицький П.А.			88 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		88

```

status_water = linux_ping(env_devices["water"]["ip"])

# === 1. ЛОГІКА ДАТЧИКА ВОЛОГОСТІ ===
hw_hum = "ON" if status_hum == 1 else "OFF"
client.publish(f"{base_topic}/humidity/status",      json.dumps({"value":
hw_hum}))

if status_hum == 1:
    humidity = round(random.uniform(44.0, 56.2), 1)
else:
    humidity = "OFFLINE"
client.publish(f"{base_topic}/humidity/percentage",  json.dumps({"value":
humidity}))

# === 2. ЛОГІКА РОЗУМНОГО РЕЛЕ ЖИВЛЕННЯ ===
hw_relay = "ON" if status_relay == 1 else "OFF"
client.publish(f"{base_topic}/power/status",        json.dumps({"value":
hw_relay}))

if status_relay == 1:
    relay_state = "On" if random.random() < 0.70 else "Off"
else:
    relay_state = "OFFLINE"
client.publish(f"{base_topic}/power/relay_state",   json.dumps({"value":
relay_state}))

# === 3. ЛОГІКА ДАТЧИКА ПРОТІКАННЯ ВОДИ ===
hw_water = "ON" if status_water == 1 else "OFF"
client.publish(f"{base_topic}/water/status",        json.dumps({"value":
hw_water}))

if status_water == 1:
    if random.random() < 0.1: # 10% шанс зловити протікання
        water_alarm = "Motion"
        try:
            requests.post(
                NTFY_URL,
                data="ATTENTION! Water leak detected in Office 3! Check
hydraulics!",

```

		Зубрицький П.А.			89ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        headers={
            "Title": "CRITICAL ALERT: Water Leak",
            "Priority": "high",
            "Tags": "droplet,warning"
        }
    )
    print("[ntfy -> Office 3 Alert Sent]")
except Exception as e:
    print(f"ntfy Office 3 failed: {e}")
else:
    water_alarm = "Off"
else:
    water_alarm = "OFFLINE"
    client.publish(f"{base_topic}/water/alarm", json.dumps({"value":
water_alarm}))

    print(f"[Office 3 Env -> JSON Sent]")
    print(f" -> humidity: status={hw_hum}, val={humidity}%")
    print(f" -> power: status={hw_relay}, state={relay_state}")
    print(f" -> water: status={hw_water}, alarm={water_alarm}")
    print("-" * 50)

    time.sleep(3)
except KeyboardInterrupt:
    client.publish(f"{base_topic}/system", json.dumps({"value": "OFF"}))
    print("\nStopping Office 3 Gateway...")
    client.loop_stop()
    client.disconnect()

```

		Зубрицький П.А.			90 ККІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

Графічні матриці верифікації мережевих інтерфейсів та адресації ядра RouterOS y WinBox

Interfaces												
Interface												
Interface List Ethernet EoIP Tunnel IP Tunnel GRE Tunnel VLAN VXLAN VRRP VETH MACsec MACVLAN Bonding LTE												
New Enable Disable Remove Find Filter												
	Name	Type	Actual MTU	L2 MTU	Tx	Rx	Tx Packet	Rx Packet	FP Tx	FP Rx		
☐	R ether1	Ethernet	1500		115.0 kbps	9.5 kbps	16	15	0 bps			
☐	R ether2	Ethernet	1500		0 bps	0 bps	0	0	0 bps			
☐	R vian11-office1-users	VLAN	1500		0 bps	0 bps	0	0	0 bps			
☐	R vian12-office1-iot	VLAN	1500		0 bps	0 bps	0	0	0 bps			
☐	R ether3	Ethernet	1500		0 bps	0 bps	0	0	0 bps			
☐	R vian21-office2-users	VLAN	1500		0 bps	0 bps	0	0	0 bps			
☐	R vian22-office2-iot	VLAN	1500		0 bps	0 bps	0	0	0 bps			
☐	R ether4	Ethernet	1500		6.2 kbps	8.5 kbps	9	9	0 bps			
☐	R vian31-office3-users	VLAN	1500		2.3 kbps	2.0 kbps	3	3	0 bps			
☐	R vian32-office3-iot	VLAN	1500		3.9 kbps	5.5 kbps	6	6	0 bps			
☐	R internal	Bridge	1500	65535	6.1 kbps	7.5 kbps	8	8	0 bps			
☐	R lo	Loopback	65536		0 bps	0 bps	0	0	0 bps			
☐	RS veth-app-caddy	VETH	1500		0 bps	0 bps	0	0	0 bps			
☐	RS veth-app-gitea	VETH	1500		0 bps	0 bps	0	0	0 bps			
☐	RS veth-app-grafana	VETH	1500		0 bps	0 bps	0	0	0 bps			
☐	RS veth-app-mosquitto	VETH	1500		6.1 kbps	8.4 kbps	8	8	0 bps			
☐	RS veth-app-nodered	VETH	1500		0 bps	0 bps	0	0	0 bps			
☐	RS veth-app-ntfy	VETH	1500		0 bps	0 bps	0	0	0 bps			
☐	RS veth-app-vaultwarden	VETH	1500		0 bps	0 bps	0	0	0 bps			

Рис. Б.1. Вкладка Interfaces в WinBox

Addresses				
New Enable Disable Remove				
	Address	Network	Interface	VRF
app network				
☐	D 172.18.0.1/24	172.18.0.0	internal	main
☐	D 192.168.0.102/24	192.168.0.0	ether1	main
Office 1 Users				
☐	192.168.1.100/25	192.168.1.0	vian11-office1-users	main
Office 1 IoT				
☐	192.168.1.200/25	192.168.1.128	vian12-office1-iot	main
Office 2 Users				
☐	192.168.2.100/25	192.168.2.0	vian21-office2-users	main
Office 2 IoT				
☐	192.168.2.200/25	192.168.2.128	vian22-office2-iot	main
Office 3 Users				
☐	192.168.3.100/25	192.168.3.0	vian31-office3-users	main
Office 3 IoT				
☐	192.168.3.200/25	192.168.3.128	vian32-office3-iot	main

Рис. Б.2. Вкладка IP > Addresses в WinBox

		Зубрицький П.А.			9ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

v4 DHCP Server							
DHCP Networks Leases Options Option Sets Option Matcher Alerts							
New Enable Disable Remove							
	Name	Interface	Relay	Lease Time	Address Pool	Add AR...	
☐	dhcp1	vlan11-office1-users		00:30:00	pool-off1-users	no	
☐	dhcp2	vlan12-office1-iot		00:30:00	pool-off1-iot	no	
☐	dhcp3	vlan21-office2-users		00:30:00	pool-off2-users	no	
☐	dhcp4	vlan22-office2-iot		00:30:00	pool-off2-iot	no	
☐	dhcp5	vlan31-office3-users		00:30:00	pool-off3-users	no	
☐	dhcp6	vlan32-office3-iot		00:30:00	pool-off3-iot	no	

v4 Pool Pools Used Addresses							
New Remove							
	Name	Addresses	Next Pool	Total	Used	Available	
☐	pool-off1-iot	192.168.1.129-192.168.1.199, 192.168.1.201-192.168.1.254		125	1	124	
☐	pool-off1-users	192.168.1.1-192.168.1.99, 192.168.1.101-192.168.1.126		125	3	122	
☐	pool-off2-iot	192.168.2.129-192.168.2.199, 192.168.2.201-192.168.2.254		125	1	124	
☐	pool-off2-users	192.168.2.1-192.168.2.99, 192.168.2.101-192.168.2.126		125	3	122	
☐	pool-off3-iot	192.168.3.129-192.168.3.199, 192.168.3.201-192.168.3.254		125	1	124	
☐	pool-off3-users	192.168.3.1-192.168.3.99, 192.168.3.101-192.168.3.126		125	3	122	

Рис. Б.3. Вкладки IP > DHCP-Server та IP > pool (Pools) WinBox

v4 Pool Pools Used Addresses				
	Pool	Address	Owner	Info
☐	pool-off1-iot	192.168.1.199	DHCP	08:00:27:2C:91...
☐	pool-off1-users	192.168.1.3	DHCP	00:50:79:66:68:...
☐	pool-off1-users	192.168.1.2	DHCP	00:50:79:66:68:...
☐	pool-off1-users	192.168.1.1	DHCP	00:50:79:66:68:...
☐	pool-off2-iot	192.168.2.199	DHCP	08:00:27:EF:F2:...
☐	pool-off2-users	192.168.2.3	DHCP	00:50:79:66:68:...
☐	pool-off2-users	192.168.2.2	DHCP	00:50:79:66:68:...
☐	pool-off2-users	192.168.2.1	DHCP	00:50:79:66:68:...
☐	pool-off3-iot	192.168.3.199	DHCP	08:00:27:2C:91...
☐	pool-off3-users	192.168.3.3	DHCP	00:50:79:66:68:...
☐	pool-off3-users	192.168.3.2	DHCP	00:50:79:66:68:...
☐	pool-off3-users	192.168.3.1	DHCP	00:50:79:66:68:...

Рис. Б.4. Вкладка IP > pool (Used Addresses)

		Зубрицький П.А.			92ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

Firewall

Filter Rules

NAT

Mangle

Raw

Service Ports

Connections

Address Lists

Layer7 Protocols

New

Enable

Disable

Remove

Find

Filter

all

☐	#	^	P	Action	Chain	Src. Address	Dst. Address	Sr...	Ds...	Proto...	Sr...	Dst. Port	In. Inter...	Out. Int...	In. ... Ou...	Bytes	Pack...
Caddy HTTP																	
☐	#	0		➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		80				0 B	0
Caddy HTTPS																	
☐	#	1		➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		443				1746 B	34
Hairpin NAT for Caddy																	
☐	#	2		➡ masquerade	srcnat	192.168.0.0/24	172.18.0.20			6 (tcp)		80,443				1226 B	24
Internet Access for all subnets																	
☐	#	3		➡ masquerade	srcnat									ether1		8.7 KiB	123
☐	#	4		➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		443				0 B	0
Caddy HTTP																	
☐	#	5		➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		80				0 B	0
Caddy HTTPS																	
☐	#	6		➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		443				0 B	0
app internal network masquerade rule to allow outgoing traffic be routed back																	
☐	#	7	D	➡ masquerade	srcnat								internal			180 B	3
app caddy redirect to web																	
☐	#	8	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		80				0 B	0
app caddy redirect to																	
☐	#	9	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		443				0 B	0
app gitea redirect to web																	
☐	#	10	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		3007				4368 B	84
app gitea redirect to ssh																	
☐	#	11	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		2222				0 B	0
app mosquito redirect to mqtt																	
☐	#	12	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		1883				5.9 KiB	101
app mosquito redirect to websockets																	
☐	#	13	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		9001				0 B	0
app ntfy redirect to web																	
☐	#	14	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		3005				20.5 ...	370
app vaultwarden redirect to web																	
☐	#	15	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		3006				0 B	0
app nodered redirect to web																	
☐	#	16	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		1880				3150 B	61
app grafana redirect to web																	
☐	#	17	D	➡ dst - nat	dstnat		192.168.0.102			6 (tcp)		3000				4472 B	86

Рис. Б.5. Вкладка IP > Firewall (NAT) в Winbox

		Зубрицький П.А.			9ЖКІКБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

**Зведена таблиця розподілу IP-адресації та параметрів кінцевих пристроїв
інфраструктури**

Таблиця В.1 - Параметри мережевих налаштувань та розподілу адресного простору вузлів лабораторного стенду

Пристрій	Інтерфейс	IP-адреса	Шлюз	Підмережа
MikroTik CHR	ether1 (WAN)	192.168.0.102	192.168.0.1	192.168.0.0/24
Container Стек	veth1 (Внутрішній)	192.168.0.102	192.168.0.1	192.168.0.0/24
MikroTik CHR	vlan11-users	192.168.1.100	-	192.168.1.0/25
MikroTik CHR	vlan12-iot	192.168.1.200	-	192.168.1.128/25
PC1 (VPCS)	Access (VLAN 11)	192.168.1.1	192.168.1.100	192.168.1.0/25
PC2 (VPCS)	Access (VLAN 11)	192.168.1.2	192.168.1.100	192.168.1.0/25
PC3 (VPCS)	Access (VLAN 11)	192.168.1.3	192.168.1.100	192.168.1.0/25
Ubuntu-1 (IoT)	Access (VLAN 12)	192.168.1.199	192.168.1.200	192.168.1.128/25
MikroTik CHR	vlan21-users	192.168.2.100	-	192.168.2.0/25
MikroTik CHR	vlan22-iot	192.168.2.200	-	192.168.2.128/25
PC4 (VPCS)	Access (VLAN 21)	192.168.2.1	192.168.2.100	192.168.2.0/25
PC5 (VPCS)	Access (VLAN 21)	192.168.2.2	192.168.2.100	192.168.2.0/25
PC6 (VPCS)	Access (VLAN 21)	192.168.2.3	192.168.2.100	192.168.2.0/25
Win_10 (IoT)	Access	192.168.2.199	192.168.2.200	192.168.2.128/25

		Зубрицький П.А.			94 ККІБ.КР.Б - 123 - 06 - ПЗ	Арк.
		Матвеев К.І.				
Змн.	Арк.	№ докум.	Підпис	Дата		94

	(VLAN 22)			
MikroTik CHR	vlan31-users	192.168.3.100	-	192.168.3.0/25
MikroTik CHR	vlan32-iot	192.168.3.200	-	192.168.3.128/25
PC7 (VPCS)	Access (VLAN 31)	192.168.3.1	192.168.3.100	192.168.3.0/25
PC8 (VPCS)	Access (VLAN 31)	192.168.3.2	192.168.3.100	192.168.3.0/25
PC9 (VPCS)	Access (VLAN 31)	192.168.3.3	192.168.3.100	192.168.3.0/25
Ubuntu-2 (IoT)	Access (VLAN 32)	192.168.3.199	192.168.3.200	192.168.3.128/25

```

PC1 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC1       192.168.1.1/25      192.168.1.100      00:50:79:66:68:00  5059   127.0.0.
1:5060
fe80::250:79ff:fe66:6800/64

```

Рис. В.1. Перевірка отриманої IP через DHCP на PC1

```

PC2 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC2       192.168.1.2/25      192.168.1.100      00:50:79:66:68:01  5061   127.0.0.
1:5062
fe80::250:79ff:fe66:6801/64

```

Рис. В.2. Перевірка отриманої IP через DHCP на PC2

```

PC3 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC3       192.168.1.3/25      192.168.1.100      00:50:79:66:68:02  5063   127.0.0.
1:5064
fe80::250:79ff:fe66:6802/64

```

Рис. В.3. Перевірка отриманої IP через DHCP на PC3

```

PC4 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC4       192.168.2.1/25    192.168.2.100    00:50:79:66:68:03    5042    127.0.0.
1:5043
          fe80::250:79ff:fe66:6803/64
  
```

Рис. В.4. Перевірка отриманої IP через DHCP на PC4

```

PC5 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC5       192.168.2.2/25    192.168.2.100    00:50:79:66:68:04    5040    127.0.0.
1:5041
          fe80::250:79ff:fe66:6804/64
  
```

Рис. В.5. Перевірка отриманої IP через DHCP на PC5

```

PC6 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC6       192.168.2.3/25    192.168.2.100    00:50:79:66:68:05    5044    127.0.0.
1:5045
          fe80::250:79ff:fe66:6805/64
  
```

Рис. В.5. Перевірка отриманої IP через DHCP на PC6

```

PC7 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC7       192.168.3.1/25    192.168.3.100    00:50:79:66:68:06    5053    127.0.0.
1:5054
          fe80::250:79ff:fe66:6806/64
  
```

Рис. В.7. Перевірка отриманої IP через DHCP на PC7

```

PC8 - PuTTY
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC8       192.168.3.2/25    192.168.3.100    00:50:79:66:68:07    5051    127.0.0.
1:5052
          fe80::250:79ff:fe66:6807/64
  
```

Рис. В.8. Перевірка отриманої IP через DHCP на PC8

```
show
NAME      IP/MASK      GATEWAY      MAC      LPORT  RHOST:PO
RT
PC9       192.168.3.3/25      192.168.3.100      00:50:79:66:68:08      5048      127.0.0.
1:5049
fe80::250:79ff:fe66:6808/64
```

Рис. В.9. Перевірка отриманої IP через DHCP на PC9

```
student@student-VirtualBox:~/Diplom$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gro
up default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether 08:00:27:2c:91:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.199/25 brd 192.168.1.255 scope global dynamic nopref
ixroute enp0s3
```

Рис. В.10. Налаштування IP на Ubuntu-1 за допомоги DHCP

```
C:\Users\Student\Desktop>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : 
    IPv4 Address. . . . . : 192.168.2.199
    Subnet Mask . . . . . : 255.255.255.128
    Default Gateway . . . . . : 192.168.2.200
```

Рис. В.11. Налаштування IP на Win-10 за допомоги DHCP

```
student@student-VirtualBox:~/Dyplom$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN gro
up default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel st
ate UP group default qlen 1000
    link/ether 08:00:27:2c:91:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.3.199/25 brd 192.168.3.255 scope global dynamic nopref
ixroute enp0s3
```

Рис. В.12. Налаштування IP на Ubuntu-2 за допомоги DHCP