

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут телекомунікаційних систем

Кафедра інформаційних технологій в телекомунікаціях

До захисту допущено:
В.О. завідувача кафедри
_____ Валерій ПРАВИЛО
«_____» _____ 2026 р.

**ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інформаційно-комунікаційні
технології»
спеціальності 172 Телекомунікації та радіотехніка**

на тему: «Модифікований метод виявлення аномальної поведінки пристроїв
інтернету речей»

Виконав: здобувач вищої освіти 4 курсу, групи ТІ-22

Гунін Іван Олександрович _____

Науковий керівник: старший викладач кафедри ІТТ НН ІТС,

Курдеча Василь Васильович _____

Рецензент: професор кафедри телекомунікацій НН ІТС,

кандидат технічних наук, професор

Якорнов Євгеній Аркадійович. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.

Здобувач вищої освіти _____

Київ – 2026 року

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут телекомунікаційних систем

Кафедра інформаційних технологій в телекомунікаціях

Рівень вищої освіти – перший (бакалаврський)

Освітньо-професійна програма «Інформаційно-комунікаційні технології»
спеціальності 172 Телекомунікації та радіотехніка

ЗАТВЕДЖУЮ

В.О. завідувача кафедри

_____ Валерій ПРАВИЛО

«_____» _____ 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Гуніну Івану Олександровичу

1. Тема дипломної роботи: «Модифікований метод виявлення аномальної поведінки пристроїв інтернету речей», науковий керівник роботи старший викладач кафедри ІТТ НН ІТС, Курдеча Василь Васильович - затверджені наказом по університету від 26.05.2026 р. №1912-с

2. Строк подання студентом дипломної роботи: 06.06.2026 р.

3. Об'єкт дослідження: Процес виявлення аномалій

4. Вихідні дані до роботи: Методи виявлення аномалій та систем виявлення вторгнень (IDS) у мережах Інтернету речей, алгоритми машинного навчання (метод головних компонент, автоенкодери), концепція периферійних обчислень (Edge Computing), протоколи безпроводової передачі даних в IoT.

5. Перелік завдань, які потрібно розробити:

Проаналізувати проблеми безпеки у мережах Інтернету речей

Проаналізувати існуючі методи виявлення аномалій у мережах Інтернету речей.

Модифікувати метод виявлення аномальної поведінки пристроїв Інтернету речей

Провести програмну реалізацію та оцінку ефективності запропонованого методу.

6. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація -14 слайдів.

7. Дата видачі завдання: 10.09.2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Узгодження організаційних питань з науковим керівником	10.09.25 - 15.09.25	Виконано
2	Узгодження теми роботи та початок роботи над першим розділом	16.09.25 - 30.09.25	Виконано
3	Планування розробки власного рішення Робота над першим, другим та третім розділами	01.10.25 - 31.03.26	Виконано
4	Висновки про ефективність рішення	01.04.26 - 30.04.26	Виконано
5	Оформлення дипломної роботи	01.05.26 - 31.05.26	Виконано
6	Підготовка презентації до захисту	01.06.26 - 10.06.26	Виконано

Здобувач вищої освіти _____ Іван ГУНІН

Науковий керівник роботи _____ Василь КУРДЕЧА

РЕФЕРАТ

Дипломна робота «Модифікований метод виявлення аномальної поведінки пристроїв інтернету речей» містить 74 сторінки тексту, 11 рисунків та 4 таблиці, 1 додаток, а також використовує 42 одиниці літературних джерел посилання.

Метою роботи є підвищення надійності функціонування безпроводових мереж Інтернету речей шляхом вдосконалення процесу виявлення аномальної поведінки пристроїв.

Це досягається шляхом дослідження аспектів пов'язаних з архітектурою IoT-мереж, процесами маршрутизації трафіку, факторами, що впливають на виникнення аномалій (апаратні збої, кібератаки), а також аналізом існуючих методів їх виявлення. У роботі проведено оцінку переваг та недоліків існуючих підходів (зокрема централізованих хмарних ML-моделей) і запропоновано модифікований до існуючих гібридний метод, що відзначається підвищеною ефективністю та зниженим обчислювальним навантаженням.

Об'єктом дослідження є процес виявлення аномалій.

Предметом дослідження є метод виявлення аномальної поведінки пристроїв у мережах Інтернету речей.

Методами дослідження є математичне та імітаційне моделювання, тобто проведення власних експериментів з метою визначення недоліків існуючих традиційних методів передачі сирих даних та можливостей їх покращення, а також розробка програмного прототипу модифікованого методу (поєднання методу головних компонент та автоенкодера).

Запропонований модифікований метод дозволяє локалізувати процес фільтрації на рівні мікроконтролера, усунути критичні затримки мережі та підвищити точність аналізу кіберзагроз без перевантаження каналу зв'язку. За результатами досліджень встановлено, що розроблений метод дозволяє точно ідентифікувати апаратний шум чи атаки і значно підвищити ефективність захисту IoT-інфраструктури порівняно з існуючими централізованими

методами. Зокрема, він запобігає забиттю буфера маршрутизатора сміттєвим трафіком, збільшує пропускну здатність для корисних даних і скорочує обсяг ресурсів, необхідних для глибокого аналізу загроз.

Роботу виконано в рамках НДР кафедри інформаційних технологій в телекомунікаціях НН ІТС № 0124U001559 "Технології Інтернету Речей для розумного будинку та розумного міста".

Ключові слова: ІНТЕРНЕТ РЕЧЕЙ (IoT), ВИЯВЛЕННЯ АНОМАЛІЙ, ПЕРИФЕРІЙНІ ОБЧИСЛЕННЯ (EDGE COMPUTING), МАШИННЕ НАВЧАННЯ, ГІБРИДНА МОДЕЛЬ, МЕТОД ГОЛОВНИХ КОМПОНЕНТ (PCA), АВТОЕНКОДЕР, СИСТЕМИ ВИЯВЛЕННЯ ВТОРГНЕНЬ (IDS), МЕРЕЖЕВИЙ ТРАФІК.

ABSTRACT

The thesis «A Modified Method for Detecting Anomalous Behavior in Internet of Things Devices» consists of 74 pages of text, 11 figures, and 4 tables, includes 1 appendix, and cites 42 references.

The aim of the thesis is to improve the reliability of wireless Internet of Things networks by refining the process of detecting anomalous device behavior, which is achieved by investigating aspects related to the architecture of IoT networks, traffic routing processes, factors influencing the occurrence of anomalies (hardware failures, cyberattacks), as well as an analysis of existing methods for their detection. The paper evaluates the advantages and disadvantages of existing approaches (in particular, centralized cloud-based ML models) and proposes a hybrid method modified from existing ones, characterized by increased efficiency and reduced computational load.

The object of the study is the process of anomaly detection.

The subject of the study is a method for detecting anomalous behavior of devices in Internet of Things networks.

The research methods include mathematical and simulation modeling, i.e., conducting our own experiments to identify the shortcomings of existing traditional methods for transmitting raw data and opportunities for their improvement, as well as the development of a software prototype of a modified method (a combination of the principal component method and an autoencoder).

The proposed modified method allows the filtering process to be localized at the microcontroller level, eliminates critical network delays, and improves the accuracy of cyber threat analysis without overloading the communication channel. Research results have shown that the developed method allows for the accurate identification of hardware noise or attacks and significantly improves the effectiveness of IoT infrastructure protection compared to existing centralized methods. In particular, it prevents the router's buffer from being clogged with junk

traffic, increases bandwidth for useful data, and reduces the amount of resources required for in-depth threat analysis.

The work was carried out within the framework of the research project of the Department of Information Technologies in Telecommunications, NN ITS No. 0124U001559 "Internet of Things Technologies for Smart Home and Smart City".

Keywords: INTERNET OF THINGS (IoT), ANOMALY DETECTION, EDGE COMPUTING, MACHINE LEARNING, HYBRID MODEL, PRINCIPAL COMPONENT ANALYSIS (PCA), AUTOENCODER, INVASION DETECTION SYSTEMS (IDS), NETWORK TRAFFIC.

ЗМІСТ

СПИСОК ТЕРМІНІВ І СКОРОЧЕНЬ.....	10
ВСТУП.....	13
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ТА СИСТЕМ ВИЯВЛЕННЯ АНОМАЛІЙ У БЕЗПРОВОДОВИХ МЕРЕЖАХ ІНТЕРНЕТУ РЕЧЕЙ..	16
1.1. Архітектура та вразливості сучасних мереж IoT.....	16
1.2. Огляд існуючих систем виявлення вторгнень (IDS).	19
1.3. Аналіз методів машинного навчання у кібербезпеці.	22
1.4. Обґрунтування доцільності використання периферійних обчислень (Edge Computing).....	24
1.5. Висновки до розділу 1	25
РОЗДІЛ 2 РОЗРОБКА МОДИФІКОВАНОГО МЕТОДУ ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ ІОТ-ПРИСТРОЇВ	27
2.1. Математичне формулювання та аналітичне обґрунтування базових компонентів.....	27
2.1.1 Аналіз головних компонент (PCA) та його формулювання мінімізації помилок.....	27
2.1.2 Автоенкодери та нелінійне виявлення прихованих загроз.....	29
2.2 Обробка та виокремлення гібридних та каскадних (Two-Stage) механізмів виявлення на Edge-рівні.....	31
2.3. Висновки до розділу 2	33
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОГО МЕТОДУ.....	35
3.1. Апаратно-програмна реалізація методу	35
3.1.1. Моделювання рівня сприйняття (Perception Layer) та концептуального дрейфу	35

	9
3.1.2. Практична реалізація гібридного фільтра (Edge AI / TinyML)	39
3.2. Оцінка надійності модифікованого методу.....	43
ЗАГАЛЬНІ ВИСНОВКИ	
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....	48
ДОДАТОК А Візуалізований код сторінки симуляції на базі середовища Github Pages.....	53

СПИСОК ТЕРМІНІВ І СКОРОЧЕНЬ

Аномалія мережевого трафіку — будь-яке відхилення характеристик передачі даних від їх нормального (очікуваного або базового) профілю, що може бути викликане апаратним збоєм сенсора, електромагнітним шумом або зловмисними кібератаками.

Автоенкодер (Autoencoder) — тип штучної нейронної мережі, що використовується для машинного навчання без учителя; складається з кодера та декодера і спрямований на вивчення ефективного представлення вхідних даних з подальшим їх відновленням. У межах роботи використовується для глибокого аналізу та виявлення прихованих кіберзагроз.

Атака нульового дня (Zero-Day Attack) — тип кібератаки, яка експлуатує раніше невідому вразливість у програмному або апаратному забезпеченні, для якої розробники ще не випустили виправлення або для якої відсутня сигнатура в базах традиційних систем безпеки.

Машинне навчання (Machine Learning, ML) — підклас штучного інтелекту, що використовує математичні моделі та статистичні алгоритми для надання комп'ютерним системам здатності навчатися на основі масивів даних і знаходити закономірності без явного програмування.

Мікроконтролер (MCU - Microcontroller Unit) — компактна інтегральна схема, розроблена для управління конкретними операціями у вбудованих системах (наприклад, чіпи ESP32, Cortex-M), яка виступає кінцевим вузлом-сенсором у мережах Інтернету речей.

Периферійні обчислення (Edge Computing) — парадигма розподілених мережових обчислень, що передбачає перенесення процесів обробки та зберігання даних максимально близько до джерела їх генерації (на рівень кінцевих пристроїв або локальних шлюзів), що зменшує час відгуку системи та розвантажує магістральні канали зв'язку.

Підміна даних (Data Injection) — тип кібератаки, під час якої зловмисник або скомпрометований пристрій впроваджує шкідливі, хибні або нереалістичні дані в систему (наприклад, підроблені показники температури чи напруги) для порушення її нормальної роботи або прийняття хибних рішень сервером.

Розумний офіс (Smart Office) — інтелектуальна інфраструктура, що об'єднує фізичні об'єкти та IoT-пристрої у єдину екосистему для автоматизації процесів, моніторингу середовища та оптимізації використання ресурсів.

Флуд / Розподілена атака на відмову в обслуговуванні (DDoS / Flooding) — кібератака, спрямована на вичерпання ресурсів обчислювальної системи або пропускної здатності комунікаційного каналу шляхом відправлення аномально великої кількості беззмистовних запитів (пакетів даних).

IDS (Intrusion Detection System — Система виявлення вторгнень) — апаратний або програмний комплекс, призначений для моніторингу мережевого трафіку та системних подій з метою своєчасного виявлення шкідливої активності, мережевих аномалій або порушень політик безпеки.

IoT (Internet of Things — Інтернет речей) — глобальна мережа підключених до Інтернету фізичних об'єктів («речей»), оснащених сенсорами, мікроконтролерами та програмним забезпеченням для безперервного збору та обміну даними.

Industrial Internet of Things — Індустріальний Інтернет речей — спеціалізоване застосування технологій Інтернету речей у промисловому секторі для критичного моніторингу, автоматизації виробничих процесів та управління складними інженерними системами.

PCA (Principal Component Analysis — Метод головних компонент) — статистичний алгоритм машинного навчання, призначений для зниження розмірності простору ознак з мінімальною втратою інформативності; у

розробленому методі використовується як швидкий та "легкий" фільтр для відсікання базового апаратного шуму.

SD-WAN (Software-Defined Wide Area Network — Програмно-визначена глобальна мережа) — технологія побудови корпоративних мереж, яка спрощує управління маршрутизацією трафіку шляхом відокремлення логічної площини управління від фізичної інфраструктури передачі даних.

ВСТУП

«Інтернет речей має потенціал змінити світ так само, як це зробив Інтернет. Можливо, навіть більше». Цю роботу доцільно розпочати саме з цієї цитати Кевіна Ештона, піонера технологій та автора самого терміну «Інтернет речей» (IoT), яка підкреслює масштаб впливу сенсорних мереж на еволюцію сучасного світу. Дійсно, важко уявити розвиток сучасного суспільства, промисловості чи концепцій «розумного кампусу» та «розумного міста» без безперервного автоматизованого збору інформації. Лише делегувавши збір та первинну обробку даних мільярдам підключених пристроїв, людство змогло досягти того рівня автоматизації, свідками якого ми є сьогодні.

«Дані — це нова нафта, але вони цінні лише тоді, коли очищені та придатні для використання». Цей відомий концепт, запропонований британським математиком Клайвом Хамбі, закладає основи сучасної дата-аналітики. Розвиваючи його думку у площині індустріального інтернету речей (IIoT), можна зробити висновок, що одним з ключових параметрів, який впливає на ефективність будь-якої автоматизованої екосистеми, є достовірність та чистота зібраних даних.

Відсутність апаратного шуму, захист від навмисних спотворень, стабільність передачі та стійкість до кібератак — це параметри, що критично впливають на якість функціонування всієї інфраструктури.

Виходячи з цього, можна стверджувати, що надійність обміну даними є ключовим фактором безпеки та ефективності розумних середовищ. Водночас масштабування таких систем неможливе без постійного моніторингу трафіку та виявлення нетипових відхилень. У контексті телекомунікацій на сьогодні існує значна кількість методів виявлення мережових вторгнень (IDS) та фільтрації трафіку. Проте класичні рішення, що покладаються на централізовану хмарну аналітику, все частіше стикаються з проблемою перевантаження каналів зв'язку та неспроможністю кінцевих IoT-пристроїв підтримувати важкі алгоритми безпеки через обмежені ресурси.

Оскільки достовірність даних, що генеруються на базовому рівні, є фундаментом для прийняття рішень автоматичними системами, ця робота присвячена вивченню саме питання виявлення аномальних даних та кіберзагроз безпосередньо в мережах індустріального інтернету речей.

Метою дипломної роботи є підвищення надійності функціонування безпроводових мереж Інтернету речей шляхом вдосконалення процесу виявлення аномальної поведінки пристроїв. Це дозволить забезпечити стабільність інфраструктури та уникнути перевантажень комунікаційних каналів при зменшенні обчислювальних ресурсів, необхідних для аналізу.

Об'єктом дослідження є процес виявлення аномалій у трафіку безпроводових мереж Інтернету речей.

Предметом дослідження є метод виявлення аномальної поведінки пристроїв у мережах Інтернету речей.

Методами дослідження є математичне та імітаційне моделювання, методи машинного навчання (метод головних компонент — PCA, штучні нейронні мережі — автоенкодери) та аналіз мережевого трафіку. Проведення експериментів за допомогою розробленого програмного прототипу має на меті визначення недоліків традиційної маршрутизації та підтвердження ефективності перенесення алгоритмів безпеки на периферійний рівень (Edge Computing).

За результатами досліджень **встановлено**, що розроблений гібридний метод дозволяє точно ідентифікувати апаратний шум та кібератаки на рівні кінцевого мікроконтролера, що значно підвищує ефективність захисту мережі порівняно з існуючими централізованими методами. Зокрема, він запобігає перевантаженню зовнішніх каналів зв'язку (шляхом блокування сміттєвого трафіку до маршрутизатора) і скорочує затримку реакції на інцидент.

Запропонований у роботі метод базується на двоетапному підході: швидкій первинній фільтрації даних та їх глибокому аналізу за умови виникнення підозри. Цей підхід може бути використаний розробниками

систем безпеки смарт-кампусів та виробниками мережевого обладнання для індустріального середовища.

Елементом наукової новизни є модифікований тод виявлення аномалій, що відрізняється поєднанням методу головних компонент та автоенкодера для зниження обчислювального навантаження на загальну систему.

Практичною новизною є те, що метод може бути інтегрований як комерційний програмний модуль у системи виявлення вторгнень, знижуючи фінансові збитки від кібератак.

Роботу виконано в рамках НДР кафедри інформаційних технологій в телекомунікаціях НН ІТС № 0124U001559 "Технології Інтернету Речей для розумного будинку та розумного міста".

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ ТА СИСТЕМ ВИЯВЛЕННЯ АНОМАЛІЙ У БЕЗПРОВОДОВИХ МЕРЕЖАХ ІНТЕРНЕТУ РЕЧЕЙ

1.1. Архітектура та вразливості сучасних мереж IoT.

Сучасні екосистеми Інтернету речей (Internet of Things, IoT) є глибоко інтегрованими, багатовимірними гетерогенними мережами, що забезпечують безперервну взаємодію між фізичним світом та кібернетичним простором. З точки зору системного аналізу, класична архітектура IoT-системи концептуалізується через три фундаментальні рівні: рівень сприйняття (Perception Layer), мережевий рівень (Network Layer) та рівень застосунків (Application Layer) [1]. Водночас розширені моделі часто включають до п'яти рівнів, додатково виокремлюючи транспортний рівень та рівень периферійних/туманних обчислень (Edge/Fog Computing). Кожен із цих рівнів володіє власними структурними особливостями, комунікаційними протоколами та специфічними поверхнями атак (attack surfaces), що перетворює забезпечення безпеки IoT на надзвичайно складну, багатовекторну проблему [1].

Рівень сприйняття є базовим компонентом, що відповідає за безпосередню взаємодію з фізичним середовищем. До його складу входять сенсори, актуатори, RFID-мітки та мікроконтролери. Концепт «речей» у цьому контексті охоплює як кібернетичний вимір (віртуальні сутності, сервіси), так і фізичний (матеріальні об'єкти, події, коливання температури тощо) [1]. Зважаючи на те, що обчислювальні можливості пристроїв на цьому рівні зазвичай є мінімальними, виникають значні перешкоди для впровадження традиційних криптографічних алгоритмів та важких систем захисту. Відповідно, вразливості рівня сприйняття носять фундаментальний характер і включають фізичне втручання (physical tampering), спуфінг сенсорних сигналів, клонування пристроїв та експлуатацію вразливостей вбудованого програмного забезпечення (прошивок) [1].

Мережевий рівень виконує критичну функцію маршрутизації та передачі зібраних даних, спираючись на широкий спектр комунікаційних протоколів (IPv6, 6LoWPAN, Zigbee, BLE, LoRaWAN, Wi-Fi, 5G/LTE). Архітектурна гетерогенність цього рівня зумовлює його вразливість до різноманітних загроз, серед яких найбільш критичними є атаки типу «людина посередині» (Man-in-the-Middle, MitM), розподілені атаки на відмову в обслуговуванні (DDoS), несанкціоноване перехоплення трафіку (eavesdropping) та ін'єкції пакетів [2].

Для мінімізації цих загроз та розвантаження магістральних мереж у сучасних архітектурах дедалі частіше використовується рівень периферійних обчислень (Edge Layer), який дозволяє здійснювати первинну фільтрацію та аналіз аномалій безпосередньо на проміжних шлюзах або кінцевих вузлах.

Рівень застосунків, своєю чергою, забезпечує фінальну обробку телеметрії, агрегацію даних у хмарних сховищах та надання інтерфейсів кінцевим користувачам. Вразливості цього рівня значною мірою корелюють із класичними веб-загрозами, такими як витік даних (data exfiltration), слабкий контроль доступу до API, SQL-ін'єкції та експлуатація логічних помилок у програмному забезпеченні [3].

Таблиця 1.1. Типологія загроз у базовій трирівневій архітектурі IoT

Рівень IoT архітектури	Основні технології та протоколи	Типові вектори атак та вразливості	Наслідки компрометації
Рівень сприйняття	RFID, NFC, GPIO, I2C, SPI, мікроконтролери	Фізичне втручання, підробка даних сенсорів, клонування, атаки на завантажувач (Bootloader)	Втрата цілісності даних на етапі збору, компрометація фізичного обладнання

Мережевий рівень	IPv6, 6LoWPAN, BLE, LoRa, MQTT, CoAP	DDoS, MitM, перехоплення (Sniffing), атаки на маршрутизацію (Sinkhole, Sybil)	Відмова в обслуговуванні, перехоплення керуючих команд, виснаження батареї
Рівень застосунків	HTTP/REST, AMQP, Хмарні бази даних	Витік даних, ін'єкції коду, компрометація API, несанкціонований доступ	Масштабний витік конфіденційної інформації, захоплення контролю над усією екосистемою

Аналіз найбільш масштабних кібератак на IoT-інфраструктуру в період з 2015 по 2024 рік свідчить про диверсифікацію векторів атак та демонструє, що традиційні криптографічні бар'єри регулярно долаються шляхом експлуатації вразливостей нульового дня (zero-day) або компрометації легітимних облікових даних [4]. З огляду на це, сучасна архітектура безпеки IoT повинна базуватися на п'яти фундаментальних принципах: (1) криптографічно підтверджена унікальна ідентифікація кожного пристрою; (2) наскрізне шифрування комунікацій; (3) безпечне завантаження (Secure Boot), що гарантує виконання пристроєм лише криптографічно підписаного вбудованого програмного забезпечення; (4) сегментація мережі на основі архітектури нульової довіри (Zero Trust), яка забезпечує ізоляцію IoT-сегмента від корпоративних IT-систем; та (5) безперервний моніторинг і динамічне виявлення аномалій [5]. Оскільки статичних заходів контролю виявляється

недостатньо в умовах постійної еволюції кіберзагроз, саме п'ятий принцип — впровадження систем виявлення вторгнень (Intrusion Detection Systems, IDS) — стає центральним елементом проактивного захисту мережі [4].

1.2. Огляд існуючих систем виявлення вторгнень (IDS).

Системи виявлення вторгнень (Intrusion Detection Systems, IDS) відіграють ключову роль у моніторингу мережі, безперервно аналізуючи мережевий трафік або системні виклики з метою виявлення шкідливих дій та диференціації нормальної поведінки й кібернетичних загроз [6]. Історично архітектури IDS поділяються на дві фундаментальні категорії, що мають взаємодоповнюючі характеристики: системи на основі сигнатур (Signature-based IDS) та системи на основі аномалій (Anomaly-based IDS) [8].

Системи сигнатурного виявлення функціонують шляхом порівняння вхідного потоку даних із попередньо скомпільованою базою відомих шаблонів загроз (сигнатур). Цей метод демонструє високу швидкість обробки, обчислювальну ефективність та мінімальний відсоток хибних спрацьовувань (False Positives) під час ідентифікації вже відомих атак [8]. Незважаючи на ці переваги, сигнатурні IDS обмежені статичним списком загроз. Їхнім критичним недоліком є нездатність виявляти атаки нульового дня (zero-day exploits), поліморфні віруси або навіть незначні мутації відомих шкідливих кодів [8]. В епоху IoT, коли ботнети щодня генерують тисячі нових варіацій шкідливого програмного забезпечення, підтримка та безперервне оновлення баз сигнатур на мільярдах кінцевих пристроїв стає логістично неможливим завданням [9].

На противагу цьому, системи виявлення на основі аномалій профілюють базову лінію (baseline) нормальної поведінки пристрою або мережі. Будь-яке статистично значуще відхилення від цієї базової лінії розцінюється як потенційне вторгнення. Саме ця здатність дозволяє ідентифікувати складні, раніше невідомі (unknown) загрози [8]. Однак класичний статистичний підхід до виявлення аномалій супроводжується серйозними недоліками. По-перше,

він вимагає значних обчислювальних ресурсів для постійного профілювання поведінки; по-друге, він генерує високу кількість хибних спрацьовувань, оскільки будь-яка легітимна зміна (наприклад, оновлення програмного забезпечення або зміна режиму роботи користувачем) може бути класифікована як підозріла [9]. Крім того, якщо нормальна поведінка спочатку профілюється неправильно (наприклад, система навчається на вже скомпрометованій мережі), її здатність виявляти справжні аномалії повністю нівелюється [9].

Щоб подолати ці обмеження, дослідники імплементують методи машинного навчання (Machine Learning, ML) та глибокого навчання (Deep Learning, DL) в архітектурі IDS. Алгоритми ML-IDS широко класифікуються на методи навчання з учителем (supervised) та без учителя (unsupervised) [10]. У науковій літературі доведено високу ефективність класичних алгоритмів, таких як метод опорних векторів (Support Vector Machine, SVM), метод k-найближчих сусідів (K-Nearest Neighbors, KNN), випадковий ліс (Random Forest, RF) та дерева рішень (Decision Tree, DT), які дозволяють з високою точністю класифікувати мережевий трафік, забезпечуючи високий ступінь інтерпретованості результатів [7].

Паралельно з цим, глибокі нейронні мережі, зокрема мережі довгої короткострокової пам'яті (Long Short-Term Memory, LSTM), штучні нейронні мережі (Artificial Neural Networks, ANN) та багатовимірні автоенкодері, демонструють здатність знаходити довгострокові та надзвичайно складні приховані закономірності (патерни) у величезних масивах неструктурованих даних [7]. Наприклад, дослідження показують, що оптимізована система виявлення на основі асиметричного складеного автоенкодера, інтегрованого з глибокою нейронною мережею (DNN), здатна досягати рівня виявлення вторгнень до 96,27% із часом прийняття рішення (інференсу) всього 0,27 секунди на наборі даних BoT-IoT. При цьому загальна точність (accuracy) становить 99,99%, влучність позитивних результатів (precision) — 99,21%, а F1-оцінка — 97,69% [11]. В інших реалізаціях використання

шумоподавлюючих автоенкодерів (Denoising Autoencoders, DAE) дозволяє досягти безпрецедентної точності у 99,991% на еталонному наборі CICIDS-2017 (F1-оцінка 0,998) та 99,4% на наборі NSL-KDD (F1-оцінка 0,989, повнота/recall 0,991) [13].

Крім того, застосування методів попередньої обробки даних (препроцесингу) перед нейромережевим аналізом суттєво підвищує загальну продуктивність систем. Аналіз показує, що комбінування автоенкодерів з такими алгоритмами, як швидке перетворення Фур'є (FFT), метод головних компонент (PCA), вейвлет-перетворення (wavelet transformations) або фільтри ковзного вікна (sliding window), дозволяє виділити найбільш релевантні ієрархічні ознаки з вихідних даних. Це зменшує їх розмірність та значно покращує точність ідентифікації загроз [14]. Дослідники також успішно застосовують моделі однокласового методу опорних векторів (One-Class SVM, OCSVM), комбіновані з наближенням Ністрёма (Nyström method) та гауссівським ескізом (Gaussian sketching), що дозволяє використовувати менше пам'яті, роблячи такі рішення придатними для масштабних систем IoT [11].

Таблиця 1.2. Порівняльний аналіз продуктивності ML-базованих систем IDS в середовищах IoT

Тип архітектури IDS	Цільовий Датасет	Заявлена точність (Accuracy)	F1-Score	Ключові особливості підходу
Asymmetric Stacked Autoencoder + DNN	BoT-IoT	99.99%	97.69%	Висока швидкість (0.27 с), однокласове навчання (One-class learning)
Denoising Autoencoder (DAE)	CICIDS 2017	99.991%	0.998	Неперевершене відновлення зашумлених даних, стійкість

				до хибних спрацьовувань
DAE + SVM (Hybrid)	NSL-KDD	99.4%	0.989	Використання DAE як інструменту зменшення розмірності перед класифікатором
HIPER- CHAD (LSTM + VAE)	Промислові дані	-	0.8571 (99th %-tile)	Двоступенева архітектура прогнозування та аналізу залишкових помилкок (Residual errors)

1.3. Аналіз методів машинного навчання у кібербезпеці.

Незважаючи на високу точність ресурсомістких моделей глибокого навчання (таких як LSTM чи ансамблеві згорткові рекурентні мережі типу CNN-GRU), їхня архітектура зумовлює суперечливу ситуацію: найбільш ефективні інструменти захисту потребують значних обчислювальних ресурсів, які принципово відсутні на кінцевих пристроях. Традиційний підхід до вирішення цієї проблеми полягає у безперервній передачі телеметрії та сенсорних даних з пристроїв IoT на централізовані сервери хмарних обчислень (Cloud Computing) для агрегації, навчання моделей (Data-driven Prognostics and Health Management, PHM) та прийняття рішень (інференсу) [15]. Проте ця хмарно-орієнтована парадигма стикається з жорсткими фізичними та комунікаційними обмеженнями, які роблять її неефективною, а подекуди й нежиттєздатною для критично важливих інфраструктур.

По-перше, виникає проблема затримки (Latency). Індустріальні системи контролю, автономні транспортні засоби або системи медичного моніторингу вимагають реакції на події в режимі реального часу. Наприклад, розвиток

аномалії, що призводить до механічної поломки ротора двигуна або виконання експлойту в пам'яті мікроконтролера, відбувається за мілісекунди. Мережевий час проходження сигналу (Round-Trip Time, RTT) до хмари та назад, обтяжений затримками на маршрутизацію та обробку масивів даних у дата-центрі, може тривати від десятків до сотень мілісекунд [15]. Така затримка робить хмарне виявлення марним для запобігання швидкоплинним кібератакам чи аваріям.

По-друге, спостерігається експоненційне зростання обсягу даних та критичне навантаження на пропускну здатність мережі (Bandwidth Overhead). Системи вібраційного або акустичного моніторингу здатні генерувати терабайти інформації [15]. Безперервна передача високочастотних сигналів або даних побічних каналів живлення (power side-channel data) від тисяч сенсорів створює надмірне навантаження на комунікаційні протоколи та призводить до невиправданих фінансових витрат на підтримку інфраструктури зв'язку [17].

По-третє, наявна проблема енергетичної неефективності. Для пристроїв з автономним живленням (таких як сільськогосподарські IoT-датчики або елементи інфраструктури «розумного кампусу») безперервна робота радіомодулів (Wi-Fi, 5G, LoRa) для трансляції масивів сирих даних споживає значно більше енергії, ніж виконання локальних математичних розрахунків [17]. Залежність від хмарних обчислень експоненційно скорочує термін автономної роботи всієї екосистеми.

По-четверте, загострюються питання конфіденційності (Privacy) та доступності (Availability). Трансляція неструктурованих даних у хмару автоматично створює нові вектори для їх перехоплення або витоку [21]. Крім того, хмарна парадигма формує єдину точку відмови (Single Point of Failure): у разі втрати інтернет-з'єднання або успішної DDoS-атаки на мережевий шлюз, весь пул кінцевих пристроїв залишається абсолютно беззахисним та неконтрольованим.

Усвідомлення цих критичних обмежень зумовило зміну парадигми проєктування систем кіберзахисту: необхідність перенесення інтелектуальної обробки даних із централізованої хмари безпосередньо на периферію мережі (Edge Computing).

1.4. Обґрунтування доцільності використання периферійних обчислень (Edge Computing).

Периферійні обчислення (Edge Computing) докорінно змінюють топологію обробки даних, децентралізуючи аналітику та переносячи її на локальні мережеві шлюзи (gateways) або безпосередньо на кінцеві вузли [15]. Наступним, найбільш радикальним витком цієї технологічної еволюції є концепція крихітного машинного навчання (Tiny Machine Learning, TinyML). Цей напрямок довів принципову можливість розгортання моделей машинного та глибокого навчання безпосередньо на ультрамалопотужних мікроконтролерах (Microcontroller Units, MCU), які мають оперативну пам'ять (SRAM) обсягом лише в кілька десятків або сотень кілобайт і споживають енергію в міліватному (мВт) діапазоні [17].

Традиційні мікроконтролери класів ARM Cortex-M (наприклад, плати Arduino Nano 33 BLE Sense), STM32, ESP32 або Raspberry Pi Pico історично використовувалися виключно для базового вводу-виводу сигналів (GPIO) та релейного управління [17]. Завдяки спільним зусиллям академічної спільноти (наприклад, проєкт MCUNet від Гарвардського університету та MIT) та індустрії (створення спеціалізованих фреймворків типу TensorFlow Lite for Microcontrollers), стало можливим запускати одновимірні згорткові мережі (1D-CNN), квантовані автоенкодери та проріджені рекурентні мережі (pruned RNN) на цих апаратно обмежених пристроях [11].

Досягнення в TinyML спираються на агресивні методи оптимізації. По-перше, це квантування (Quantization), яке конвертує 32-бітні числа з рухомою комою (Float32) у 8-бітні цілі числа (INT8) без суттєвої втрати точності моделі,

зменшуючи розмір файлу нейромережі (наприклад, у форматі .tflite) до 20–50 КБ [11]. По-друге, це використання оптимізованих бібліотек, таких як CMSIS-NN, що дозволяють ефективно використовувати апаратні інструкції цифрової обробки сигналів (DSP) процесорів архітектури Cortex-M [22].

Практичні результати імплементації TinyML для систем виявлення аномалій та вторгнень демонструють високі показники ефективності. Наприклад, повністю автономна система виявлення аномалій на базі Z-оцінки, розгорнута на платформі STM32 для аналізу побічних каналів живлення, забезпечує точність і повноту (Precision and Recall) на рівні 1,00. Час інференсу такої моделі становить десятки мікросекунд, а загальний обсяг використовуваної пам'яті дорівнює близько 3,3 КБ SRAM та 63 КБ Flash-пам'яті [17]. Інший підхід використовує алгоритм ізольованого лісу (Isolation Forest), який тренується на мікроконтролері за 1,2–6,4 секунди та виявляє аномалії менш ніж за 16 мілісекунд, використовуючи ансамбль із 50 дерев та лише 80 КБ оперативної пам'яті [18]. Інтеграція таких систем з технологією блокчейн додатково забезпечує створення прозорого та неспростовного репозиторію подій безпеки [18]. Інші дослідження показують, що полегшені моделі (Lightweight DT + small NN) гарантують надзвичайно низьку затримку в 0,12 мс при споживанні лише 0,01 мВт електроенергії [20].

Проте жорсткі обмеження апаратних ресурсів (насамперед обсягу SRAM) не дозволяють запускати складні згорткові або глибокі рекурентні мережі в їхньому первісному вигляді. Це породжує гостру необхідність розробки нових, спеціалізованих гібридних алгоритмів виявлення, які б поєднували статистичну легкість та швидкість традиційних методів із високою репрезентативною здатністю алгоритмів глибокого навчання.

1.5. Висновки до розділу 1

У результаті проведеного аналізу встановлено, що сучасні екосистеми Інтернету речей характеризуються багатовекторною вразливістю на всіх

рівнях архітектури, що робить статичні криптографічні заходи недостатніми та вимагає обов'язкового впровадження динамічних систем виявлення вторгнень. Дослідження існуючих архітектур безпеки довело, що традиційні сигнатурні методи є вразливими до атак нульового дня, тоді як системи на основі аномалій, підсилені алгоритмами машинного та глибокого навчання, здатні з високою точністю ідентифікувати складні та раніше невідомі кіберзагрози.

Водночас застосування ресурсомістких нейромережових моделей у межах класичної хмарної парадигми генерує низку критичних проблем для IoT-інфраструктури. Необхідність безперервної трансляції сирих даних на централізовані сервери призводить до неприпустимих затримок у реагуванні на інциденти, надмірного перевантаження пропускної здатності каналів зв'язку, швидкого виснаження автономних джерел живлення та створення єдиної точки відмови. Ці фундаментальні обмеження роблять хмарний підхід неефективним для захисту критично важливих систем, що функціонують у режимі реального часу.

Вирішення окресленої проблематики об'єктивно вимагає переходу до парадигми периферійних обчислень та імплементації технологій крихітного машинного навчання (TinyML), які дозволяють децентралізувати інтелектуальну обробку даних і перенести її безпосередньо на ультрамалопотужні кінцеві вузли. Аналіз сучасних досліджень підтвердив технічну можливість оптимізації та розгортання нейромережових моделей на мікроконтролерах. Проте жорсткі апаратні обмеження таких пристроїв, насамперед дефіцит оперативної пам'яті, унеможливають використання глибоких архітектур у їхньому первісному вигляді. Це формує актуальну науково-практичну задачу: необхідність розробки вдосконалених гібридних алгоритмів захисту, які б збалансовано поєднували високу швидкість та обчислювальну легкість статистичної попередньої фільтрації з точністю глибокого нейромережового аналізу аномалій.

РОЗДІЛ 2

РОЗРОБКА МОДИФІКОВАНОГО МЕТОДУ ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ ІОТ-ПРИСТРОЇВ

Проблема впровадження високорівневої інтелектуальної аналітики безпосередньо на рівні сенсорних вузлів вимагає фундаментального перегляду конвеєра обробки даних. Звичайна мініатюризація архітектур існуючих нейронних мереж часто призводить до втрати їхньої здатності ідентифікувати складні просторово-часові кібератаки. З огляду на це, метою даного розділу є математичне обґрунтування та проєктування ресурсоефективної гібридної системи виявлення аномалій, яка синергетично поєднує метод головних компонент та глибокі автоенкодери в єдиній каскадній архітектурі.

2.1. Математичне формулювання та аналітичне обґрунтування базових компонентів

2.1.1 Аналіз головних компонент (PCA) та його формулювання мінімізації помилок

Метод головних компонент (Principal Component Analysis, PCA) — це ортогональне лінійне перетворення, яке проєктує дані з високорозмірного простору в простір меншої розмірності (латентний простір), зберігаючи при цьому максимальну дисперсію вихідних даних [16].

Математично процес починається з набору даних $\{x_n\}$, де $n = 1, 2, \dots, N$, а кожен вектор $x_n \in \mathbb{R}^D$ описує багатовимірний стан IoT-пристрою (наприклад, D сенсорних показників або мережевих атрибутів).

Коваріаційна матриця цього набору даних обчислюється як [26]:

$$S = \frac{1}{N} \sum_{n=1}^N (x_n - \bar{x})(x_n - \bar{x})^T \quad (2.1)$$

Згідно з критерієм максимальної дисперсії, метою є знаходження такого вектора w_1 , який максимізує дисперсію спроектованих даних:

$$w_1^T S w_1 \quad (2.2)$$

Щоб уникнути тривіального розв'язку при $\|w_1\| \rightarrow \infty$, вводиться обмеження одиничної норми:

$$w_1^T w_1 = 1 \quad (2.3)$$

Із використанням множника Лагранжа λ_1 , задача оптимізації набуває вигляду:

$$\mathcal{L} = w_1^T S w_1 + \lambda_1 (1 - w_1^T w_1) \quad (2.4)$$

Прирівнюючи похідну функції Лагранжа по w_1 до нуля, отримуємо рівняння [26]

$$S w_1 = \lambda_1 w_1 \quad (2.5)$$

Це класична задача на знаходження власних значень та власних векторів. Максимальна дисперсія у просторі меншої розмірності дорівнює власному значенню λ_1 [26]. Відповідно, перші M головних компонент — це власні вектори, що відповідають M найбільшим власним значенням.

Для задачі виявлення аномалій найбільш релевантною є постановка задачі мінімізації похибки (Minimum Error Formulation) [26]. Її суть полягає у знаходженні трансформації, що мінімізує похибку реконструкції (Reconstruction Error, RE) під час зворотного відображення даних із латентного простору $\mathbb{R}^M$ у вихідний простір $\mathbb{R}^D$:

$$RE = \|x_n - \hat{x}_n\|^2 \quad (2.6)$$

де $\hat{x}_n$ — реконструйований сигнал [26].

Після центрування даних проєкція на перші k компонент матриці $U_k \in \mathbb{R}^{D \times k}$ дозволяє відновити сигнал як

$$\hat{x} = U_k U_k^T x \quad (2.7)$$

Мірою аномальності (Anomaly Score) у цьому випадку слугує середньоквадратична похибка реконструкції [28].

Зазначений підхід вирізняється високою ефективністю, оскільки нормальні дані утворюють щільний багатовимірний підпростір, який адекватно описується першими головними компонентами. Якщо система фіксує аномальний патерн (наприклад, нетипову вібрацію від пошкодженого підшипника), він проєктується в «нуль-простір» (null subspace) — на ознаки з

близькою до нуля дисперсією [29]. Під час реконструкції такі унікальні відхилення не відновлюються, що призводить до різкого зростання величини RE [25].

У ймовірнісному розширенні методу (Probabilistic PCA) дані моделюються за допомогою нормального розподілу

$$p(x) = \mathcal{N}(x | b, C) \quad (2.8)$$

де коваріаційна матриця задається як

$$C = \sigma^2 I + WW^T \quad (2.9)$$

Оптимальна реконструкція за методом найменших квадратів у цій моделі має вигляд [27]:

$$\hat{x} = Bx \quad (2.10)$$

де

$$B = W(W^T W)^{-1} W^T \quad (2.11).$$

Це дозволяє не лише розраховувати абсолютну похибку, але й визначати статистичну ймовірність появи конкретного вектора даних. вектора даних.

2.1.2 Автоенкодери та нелінійне виявлення прихованих загроз

Хоча метод PCA є надзвичайно ефективним з обчислювальної точки зору, він обмежений виявленням лише лінійних кореляцій [31]. Складні мережеві атаки або акустичні аномалії часто утворюють нелінійні многовиди (manifolds), для яких лінійне стиснення виявляється недостатнім. Для вирішення цього завдання застосовуються автоенкодери (Autoencoders, AE) — архітектури глибокого навчання без учителя [32].

Архітектура автоенкодера складається з енкодера

$$h = f_{\theta}(x) \quad (2.12)$$

який трансформує вхідні дані x у компактне нелінійне приховане представлення (latent space), та декодера

$$x' = g_{\phi}(h) \quad (2.13)$$

який реконструює дані назад.

Процес тренування мінімізує функцію втрат:

$$\mathcal{L}_{AE}(\theta, \phi) = \frac{1}{N} \sum_{i=1}^N \|x_i - g_{\phi}(f_{\theta}(x_i))\|^2 \quad (2.14)$$

Специфічні архітектури, такі як шумоподавлюючі автоенкодера (Denoising Autoencoders, DAE) та варіаційні автоенкодера (Variational Autoencoders, VAE), спеціально розроблені для роботи із зашумленими даними сенсорів [13].

Моделі DAE тренуються на штучно зашумлених версіях вхідних даних $\tilde{x}$, навчаючись відновлювати початковий «чистий» сигнал. Це змушує нейронну мережу вивчати стійку структуру легітимного сигналу, ігноруючи стохастичний сенсорний шум [13].

VAE, своєю чергою, використовує ймовірнісний підхід (reparameterization trick), у якому енкодер відображає вхідні дані на параметри латентного нормального (гауссівського) розподілу, що робить латентний простір безперервним та гладким [34].

Автоенкодер навчається відновлювати виключно ті патерни, на яких він тренувався (тобто нормальну поведінку мережі). Коли на вхід потрапляє невідома атака, мережа намагається реконструювати її за правилами «нормальності», що призводить до високої похибки реконструкції (MSE).

Дослідження вібраційних характеристик показують чітке розмежування між гістограмами розподілу MSE для нормальних і аномальних зразків, що дозволяє легко встановити порогове значення (Threshold) для відсікання аномалій [37].

Крім того, на основі карт похибок $e(x)$ можуть розраховуватись додаткові метрики, такі як рівень локального відхилення (Local Outlier Factor, LOF) або дисперсія Монте-Карло для моделей з використанням методу виключення (dropout) під час стохастичних прямих проходів (stochastic forward passes) [28].

2.2 Обробка та виокремлення гібридних та каскадних (Two-Stage) механізмів виявлення на Edge-рівні

Розгортання складного автоенкодера безпосередньо на мікроконтролері для обробки багатовимірних сирих даних із високою частотою дискретизації швидко вичерпує як обчислювальні ресурси процесора, так і заряд акумулятора [31]. Вирішення цієї інженерної суперечності досягається шляхом інтеграції методу головних компонент (PCA) та автоенкодерів у єдині гібридні або каскадні (двоступеневі) архітектури [29].

Гібридний підхід (Feature Extraction Integration). У цій моделі PCA виконує роль потужного препроцесора та інструменту зменшення розмірності. Замість подачі в автоенкодер вектора розмірністю понад 1000 елементів, метод PCA (або дисперсійний аналіз ANOVA, який ефективно пріоритезує ознаки з високою роздільною здатністю [40]) стискає дані до k головних компонент, усуваючи апаратну надмірність датчиків та стохастичний шум [15]. Отриманий стиснутий вектор подається на вхід нелінійному автоенкодеру (алгоритм PCA-NN) [33]. Це суттєво зменшує кількість синаптичних ваг у нейронній мережі, знижуючи споживання оперативної пам'яті (SRAM) мікроконтролера та прискорюючи процес прийняття рішень (інференсу). Доведено, що комбіновані методи виділення ознак досягають коефіцієнта стиснення від 80% до 98% та забезпечують прискорення обробки від 4 до 21 разу порівняно з базовими методами без втрати інформативності [41]. Крім того, латентне представлення (bottleneck) може додатково проєктуватись у двовимірний простір за допомогою PCA для візуалізації та простої кластеризації подій безпеки [24].

Каскадна двоступенева архітектура (Cascaded Two-Stage Architecture). Більш складний, але енергетично найефективніший підхід передбачає послідовну роботу алгоритмів різного рівня складності [29]. Оскільки IoT-система функціонує в нормальному (легітимному) стані понад 99% часу, постійний виклик глибокої нейромережі є надлишковим.

- **Етап 1 (Edge-фільтрація низької потужності):** На мікроконтролері безперервно, з високою частотою (тисячі разів на секунду), працює обчислювально легкий лінійний алгоритм (РСА, статистична Z-оцінка або ізольований ліс). Використовується концепція подвійних порогів (Dual Thresholds) [29]. Обчислюється похибка реконструкції РСА на основі ознак із близькою до нуля дисперсією [29]. Якщо похибка реконструкції мінімальна (сигнал гарантовано легітимний), кадр даних відкидається. Якщо похибка перевищує верхній поріг (сигнал є явною аномалією), миттєво генерується сигнал тривоги.

- **Етап 2 (Глибока нелінійна інспекція):** Лише в тому випадку, коли значення показника аномальності потрапляє у проміжну зону невизначеності (Ambiguous samples), мікроконтролер пересилає цей специфічний кадр на глибоку обробку [29]. Це може ініціювати пробудження локального співпроцесора зі згортковим чи рекурентним (LSTM) автоенкодером або вивантаження (offloading) даних на локальний шлюз чи хмарний сервер для ідентифікації складних постійних загроз (Advanced Persistent Threat, APT) [34].

Важливою інновацією в цьому підході є механізм адаптивного навчання (Adaptive Learning Mechanism) для динамічного коригування порогу вивантаження (offloading threshold) під час інференсу. Використовуючи математичний апарат мінімізації жалю (regret bound framework), система динамічно вирішує: обробляти вхідні дані локально чи перенести обчислення на потужніший бекенд, з огляду на поточну вартість комунікацій, залишок заряду батареї та складність патерну даних [34]. Це забезпечує оптимальний компроміс між енергоефективністю обчислень і точністю виявлення загроз [34].

Таблиця 2.1. Архітектурні характеристики підходів на базі мікроконтролерів

Характеристика	Стандартний PCA / Z-Score	Глибокий Автоенкодер (DAE/VAE)	Каскадний (Двоступеневий) Гібрид
Схильність до виявлення	Лінійні шуми, прості відхилення	Складні APT-атаки, нелінійні патерни	Повний спектр відхилень і складних атак
Споживання пам'яті (RAM/Flash)	Надзвичайно низьке (~3-5 КБ) ¹⁷	Відносно високе (50-250 КБ)	Адаптивне (активує повну пам'ять лише за потреби)
Енергоспоживання (Інференс)	Мікровати (μ W), десятки мікросекунд	Мілівати (mW), до 15-20 мс	Оптимізоване ($\sim \mu$ W у 95% випадків)
Стійкість до Concept Drift	Низька (без нормалізації)	Висока (якщо адаптувати латентний простір)	Надзвичайно висока (комбіновані метрики)

2.3. Висновки до розділу 2

Реалізація розробленої гібридної системи передбачає перехід від теоретичного моделювання мовою Python до апаратної імплементації мовами C/C++. Для розгортання автоенкодерів на мікроконтролерах використовується спеціалізований фреймворк TensorFlow Lite for Microcontrollers (TFLM).

Процес імплементації розпочинається зі збору початкових даних, навчання невеликої моделі (наприклад, з повнозв'язними шарами Dense та функцією активації ReLU) і подальшого її перетворення за допомогою конвертера TFLiteConverter [23]. Ключовим етапом є квантування моделі (Quantization), що зменшує розмір вагових коефіцієнтів, перетворюючи тензори у формат INT8. Отримана оптимізована модель (розміром близько 20–50 КБ) записується у Flash-пам'ять мікроконтролерів, таких як Arduino Nano 33 BLE Sense або ESP32 [23].

Крім того, останні дослідження підтверджують можливість не лише виконання локального інференсу, а й децентралізованого навчання безпосередньо на мікроконтролерах. Зокрема, у дослідженні з використанням платформи ESP32-CAM було продемонстровано можливість локального тренування компактних автоенкодерів за допомогою 32-бітної арифметики з рухомою комою (float32) [21]. Час навчання становив від 12 хвилин, а затримка (латентність) інференсу була знижена до 9 мілісекунд [21].

Це відкриває шлях до впровадження федеративного навчання (Federated Learning, TinyFL) на периферійних пристроях [21]. Десятки мікроконтролерів можуть спільно навчати гібридну PCA-AE модель без обміну сирими конфіденційними даними (замість цього на центральний сервер передаються лише оновлення градієнтів). Доведено, що така агрегація залишається стійкою навіть у сценаріях екстремального нерівномірного розподілу даних (non-IID data distribution), досягаючи показника F1-оцінки на рівні 0,86–0,87 [21].

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОГО МЕТОДУ

3.1. Апаратно-програмна реалізація методу

Додаток 1 містить розроблену програмну симуляційну модель, яка виступає практичною імплементацією теоретичних положень дослідження та демонструє ефективність застосування периферійних обчислень для фільтрації IoT-трафіку

3.1.1. Моделювання рівня сприйняття (Perception Layer) та концептуального дрейфу

Рівень сприйняття (Perception Layer), також відомий як фізичний або сенсорний рівень, є найнижчим фундаментальним рівнем архітектури Інтернету речей, який складається з мікроконтролерів і датчиків, що безпосередньо збирають дані з фізичного світу (наприклад, температуру, вібрацію, рівень струму). У запропонованій програмній симуляції цей рівень абстраговано через функцію-генератор `setInterval()`, яка виконується асинхронно кожні 100 мілісекунд. Вона моделює часовий ряд (телеметрію) за допомогою базової гармонійної функції:

```
let signal = Math.sin(timeT) * 30;;
```

де змінна часу `timeT` інкрементується на 0.2 кожен такт.

Математично базовий стан сенсора в момент часу (t) можна описати як

$$x(t) = s(t) + \epsilon(t) + a(t) \quad (3.1)$$

де $s(t)$ — корисний сигнал, $\epsilon(t)$ — нормальний експлуатаційний шум, а $a(t)$ — аномальна складова (фізична несправність або кібератака).

Симулятор відтворює ці складові наступним чином:

1. Стохастичний шум (Stochastic Noise). У промислових та побутових умовах ідеальних сенсорних даних не існує. Шум у даних від датчиків сприяє випадковим відхиленням вимірювань, які постійно додають невизначеність до

показників і є важкими для виявлення та корекції. У симуляторі змінна `isNoise` генерує ці випадкові флуктуації. Логіка коду `let isNoise = Math.random() < 0.4;` задає сталу 40% ймовірність появи шуму в нормальному стані роботи.

Якщо умова виконується, до ідеального сигналу додається випадкове відхилення:

```
signal += (Math.random() - 0.5) * 15;.
```

З точки зору статистичного моделювання, цей програмний фрагмент апроксимує Гауссівський (нормальний) розподіл: $\mathcal{N}(0, \sigma^2)$, імітуючи тепловий або електромагнітний шум обладнання. Візуально на графіку `sensorChart` цей шум відображається як незначні коливання синьої лінії, а в об'єктній моделі створюються відповідні пакети для маршрутизації:

```
packets.push(new Packet('noise', 1, nodes.sensor,
                        nodes.edge));.
```

Завдання систем виявлення (наприклад, на базі методу головних компонент — PCA) полягає в тому, щоб навчитися ігнорувати цю дисперсію нормального шуму, проєктуючи її у "нульовий підпростір" (null subspace), щоб уникнути хибних спрацьовувань під час штатної роботи.

2. Генерація аномалій (Атаки типу False Data Injection). Активація функції `isAnomaly` (яка в коді ініціюється обробником подій

```
document.getElementById('anomalyBtn').addEventListener('click', ...)) \
```

математично моделює клас загроз, відомий як атаки впровадження хибних даних (False Data Injection Attacks, FDIA) або раптові механічні збої. В умовах IoT-мереж злоумисник, маючи доступ до вразливого рівня сприйняття, може впровадити оманливі дані в показання сенсорів.

У коді це реалізовано через комплексну ймовірнісну логіку. Ймовірність успішної атаки в конкретний мілісекундний такт обчислюється динамічно:

```
let spikeProb = 0.05 + (anomalyStrength * 0.05);.
```

Якщо генератор псевдовипадкових чисел ініціює атаку (`Math.random() < spikeProb`), система розраховує амплітуду хибних даних (вектор a):

```
let spikeAmp = 40 + (anomalyStrength * 20);.
```

Цей зловмисний вектор безпосередньо ін'єктується в легітимний сигнал:

```
signal += (Math.random() > 0.5? spikeAmp : -
          spikeAmp);
```

Це ідеально відтворює класичну математичну модель stealth-атаки:

$$z_a = z + a \quad (3.2)$$

де z — легітимний вектор вимірювань, a — впроваджені хибні дані, а z_a — скомпрометований результат.

У симуляторі це миттєво візуалізується зміною кольору графіка на оранжевий (`#f59e0b`) та масованою генерацією пакетів тривоги (`Packet('alert',...)`), кількість яких у сплеску (`burstSize = Math.ceil(anomalyStrength / 1.5)`) перевантажує мережу. Оскільки такі атаки імітують формат легітимних даних, системи сигнатурного захисту їх не виявляють, вимагаючи використання автоенкодерів, які фіксують високу помилку реконструкції (MSE) при спробі відтворити такий сигнал.

3. Концептуальний дрейф (Concept Drift). Можливість динамічної зміни параметра сили атаки під час роботи симулятора (через HTML-елемент `<input type="range" id="anomalyStrength">`, який оновлює змінну `anomalyStrength` від 1 до 5) є практичною симуляцією явища концептуального дрейфу (Concept Drift).

Формально, концептуальний дрейф визначається як зміна спільного розподілу ймовірностей вхідних даних та цільової змінної з часом:

$$P_{t_i}(X, Y) \neq P_{t_j}(X, Y), \text{ при } t_i < t_j. \quad (3.3)$$

У коді вплив цього повзунка є наскрізним і нелінійним. Він змінює не лише амплітуду FDIA-атак, але й базовий шумовий фон системи. У блоці коду

```
else { isNoise = Math.random() < (0.3 +
    anomalyStrength*0.1);... }
```

ми бачимо, як при зростанні `anomalyStrength` ймовірність появи фонового шуму зростає, а його амплітуда мультиплікується:

```
signal += (Math.random() - 0.5) * 15 *
    anomalyStrength;
```

Це математично точно імітує "envelope drift" (повільний зсув обвідної). У реальних IoT-системах це відповідає ситуації, коли через поступовий знос підшипників або деградацію самого датчика "базова лінія" нормальної вібрації зміщується. Якщо система виявлення аномалій (в даному випадку мікроконтролер на Edge-рівні) використовує заморожений поріг (frozen-threshold), відкалібрований на початкових даних, зсув цього розподілу призведе до того, що нові нормальні (але більш зашумлені) дані не будуть ігноруватися методом `noise dropping`, а розцінюватимуться як аномалії. Це спричинить експоненційне зростання хибних тривог, що доводить архітектурну необхідність впровадження механізмів адаптації, таких як спектральна адаптація рівня шуму (TSNFA), яка здатна нормалізувати дрейф до того, як дані потраплять до автоенкодера.

3.1.2. Практична реалізація гібридного фільтра (Edge AI / TinyML)

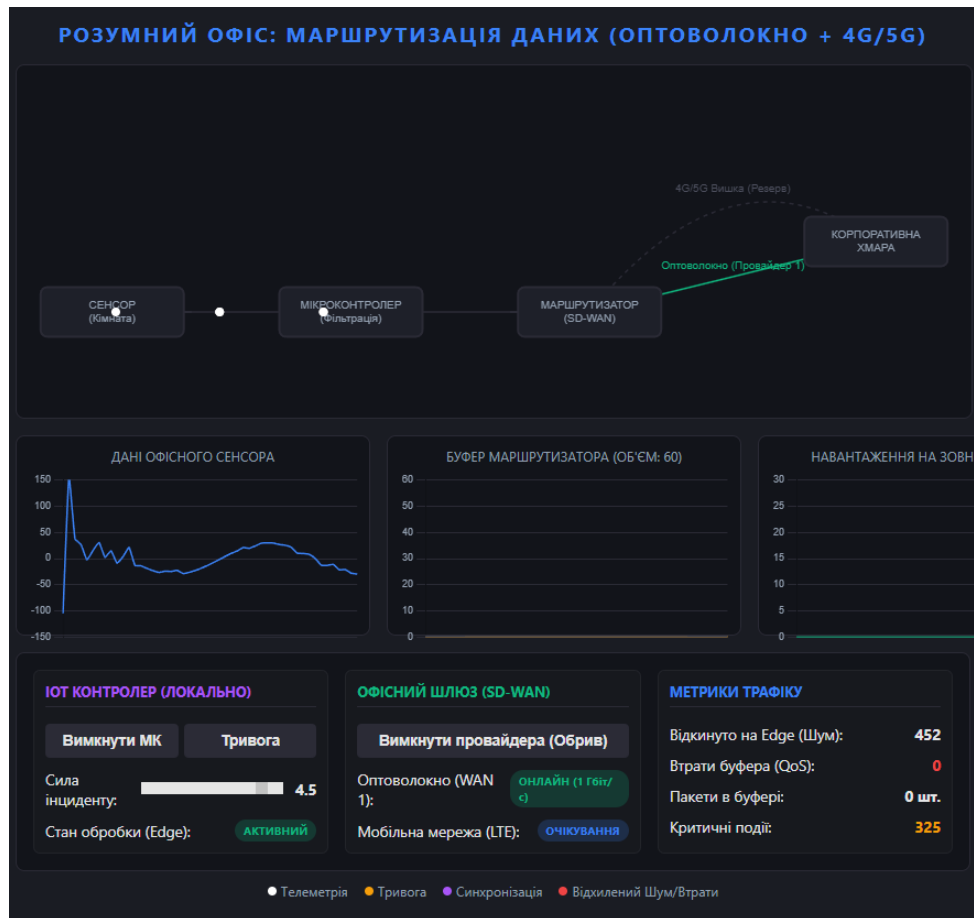


Рисунок 3.1. Загальний вигляд сторінки симуляції

Датчики рівня сприйняття (Perception Layer) постійно генерують корисний сигнал, який завжди супроводжується стохастичним (випадковим) шумом. Передача всього цього шуму до хмари є енергетично та комунікаційно неефективною. За збір даних відповідає блок Data Generator (функція `setInterval`, що викликається кожні 100 мс).

```
let signal = Math.sin(timeT) * 30; // Базовий корисний сигнал
let isNoise = Math.random() < 0.4; // 40% ймовірність
природного шуму
if (isNoise) {
  signal += (Math.random() - 0.5) * 15; // Накладання флуктуацій
  packets.push(new Packet('noise', 1, nodes.sensor,
nodes.edge));
}
```



Рисунок 3.2. Блок-схема симулятора

Архітектура передачі даних у симуляції відповідає багаторівневій концепції та проходить наступний шлях: **Сенсор** → **Edge-контролер** (наприклад, мікроконтролер ESP32) → **Проміжний вузол** (Fog-сервер / Локальний сервер корпусу) → **Корпоративна Хмара**.

Fog-сервер відіграє роль інтелектуального маршрутизатора і має два доступні канали зв'язку до хмари:

- **Основний канал:** Високошвидкісне дротове підключення (наземний Terrestrial канал, наприклад, Оптоволокну або Ethernet).
- **Резервний канал:** Безпроводова мережа, яка активується при падінні або перевантаженні основного інтернет-з'єднання. Це може бути мобільна мережа (4G/LTE або 5G через 4G-модем у маршрутизаторі) або ж повільніший супутниковий зв'язок (Non-Terrestrial Network - NTN, наприклад, Starlink).

Датчики (сенсори) постійно генерують корисний сигнал, який завжди супроводжується стохастичним шумом. У коді симуляції генератор даних кожні 100 мілісекунд синтезує телеметрію. Цей процес імітує нормальний тепловий або електромагнітний шум, який традиційні сигнатурні та хмарні системи часто плутають з аномаліями, що призводить до непотрібного навантаження каналів зв'язку.



Рисунок 3.3. Графік проходження даних через сенсор (еталонні умови)

Завдяки розгорнутій моделі TinyML, Edge-контролер виконує попередню фільтрацію прямо на місці. Він не просто відсіює лінійний шум, але й класифікує та маркує пакети даних:

- **Звичайні дані (білі маркери):** Стандартна телеметрія, що не несе загроз.
- **Аномалії (помаранчеві маркери):** Пакети, що генеруються під час кібератак (наприклад, False Data Injection) або фізичних збоїв.
- **Синхронізація (фіолетові маркери):** Edge-вузол періодично генерує спеціальні пакети оновлення параметрів нейромережі. Вони відправляються в хмару для покращення глобальної моделі в рамках парадигми Федеративного навчання (Federated Learning, FL). Це дозволяє оновлювати систему виявлення без необхідності передачі конфіденційних «сірих» даних.

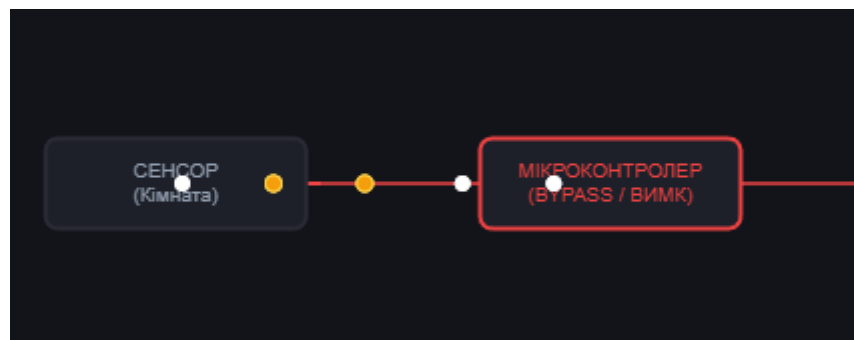


Рисунок 3.4. Візуалізація (Увімкнений bypass)

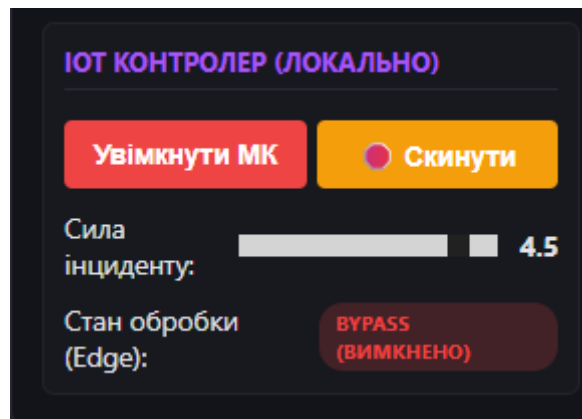


Рисунок 3.5. Графік проходження даних через сенсор (Увімкнений bypass у положенні 4.5)

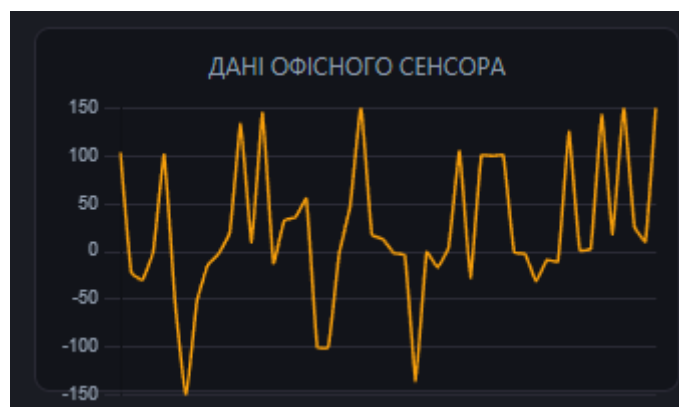


Рисунок 3.6. Графік проходження даних через сенсор (Увімкнений bypass у положенні 4.5)

Центральним елементом симуляції є доведення необхідності впровадження механізмів управління перевантаженням (QoS) при деградації зв'язку. Коли основний оптоволоконний канал падає, Fog-сервер перемикається на резервний мобільний (4G/5G) або супутниковий (NTN) зв'язок.

Оскільки такий трафік часто є лімітованим і має значно нижчу пропускну здатність, локальний сервер офісу починає застосовувати жорсткіший QoS. Fog-сервер агрегує та накопичує пакети в локальний буфер. Щоб не перевантажити вузький канал і не витратити лімітований трафік даремно, система починає затримувати або примусово відкидати "важку" рутинну телеметрію (білі пакети). Натомість абсолютний пріоритет віддається критичним подіям: помаранчеві пакети (спрацювання датчиків безпеки чи

пожежної тривоги) та фіолетові FL-пакети миттєво пропускаються до хмарної інфраструктури.

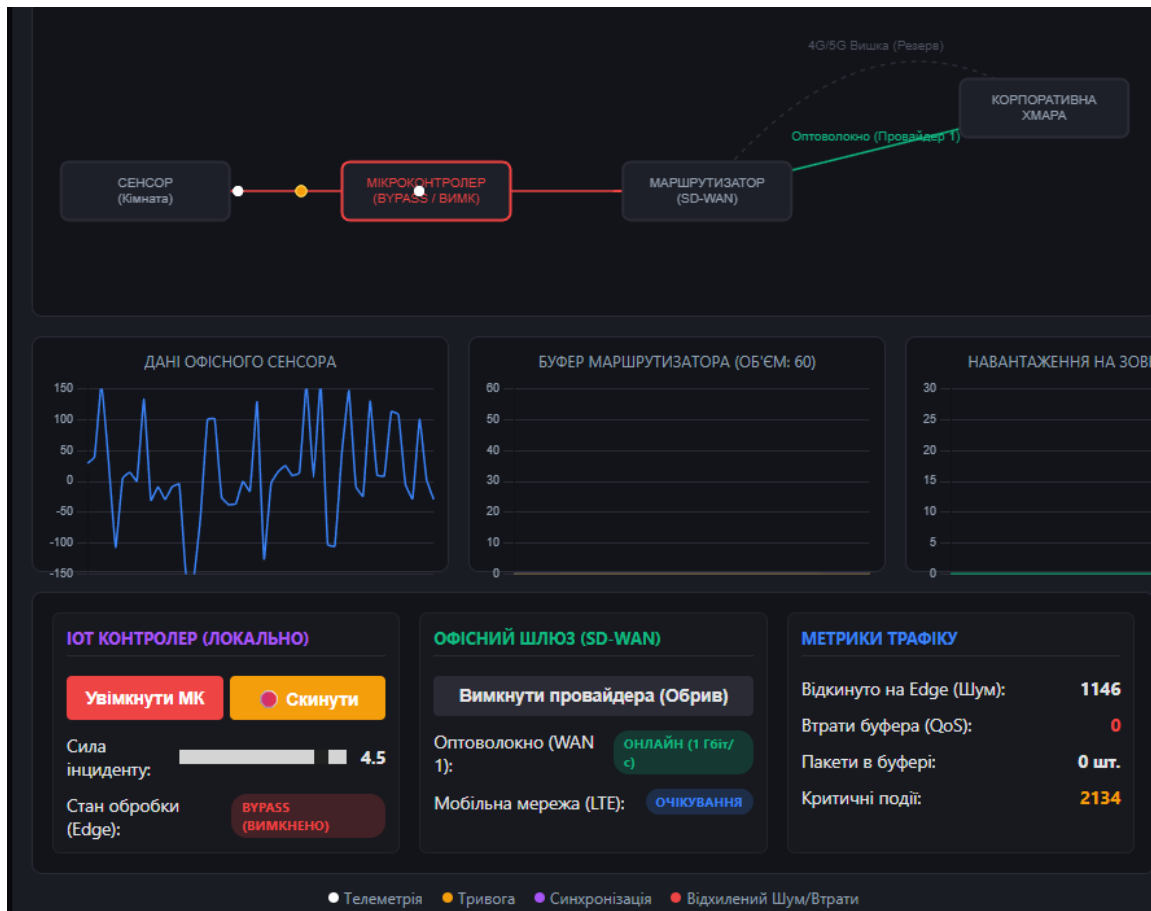


Рисунок 3.7. Статистичні дані, зібрані з використанням програмної симуляції.

3.2. Оцінка ефективності модифікованого методу

Функціонування розробленої системи у фізичному середовищі неминуче стикається з явищем нестаціонарності. Головною проблемою алгоритмів, що базуються на похибці реконструкції (зокрема, архітектур зі статичним/фіксованим порогом — *frozen-threshold architecture*), є концептуальний дрейф або повільний дрейф огибаючої (*envelope drift*). У розробленій програмній симуляції цей процес емпірично моделюється через динамічну зміну параметра *anomalyStrength*, який нелінійно збільшує не лише амплітуду цілеспрямованих атак, але й базову дисперсію нормального фонового шуму. З плином часу температурні коливання в роботі датчиків або механічний знос обладнання призводять до аналогічного зміщення базового

рівня. Відповідно, встановлений на етапі калібрування поріг MSE починає генерувати структурний «базовий рівень» хибних спрацьовувань (що в симуляційній моделі призвело б до хибного маркування шумових пакетів як alert та неминучого колапсу буфера маршрутизатора). Емпіричні тести на апаратних вузлах Cortex-M4F зафіксували частоту близько 1000 хибних тривог на годину для одного пристрою за умови використання класичних повнозв'язних (MLP) автоенкодерів [24].

Для розв'язання цієї проблеми в архітектуру системи інтегрується механізм часової спектральної адаптації рівня шуму (Temporal Spectral Noise-Floor Adaptation, TSNFA). У контексті симулятора це еквівалентно здатності Edge-контролера безперервно та коректно класифікувати й відкидати пакети типу noise, зберігаючи свою ефективність навіть за умов штучного завищення загального шумового фону. Цей метод використовує нестационарну модель шуму для генерації єдиної скалярної оцінки дрейфу, яка нормалізує кожен вхідний кадр даних до моменту його потрапляння на вхід автоенкодера. Завдяки такій попередній нормалізації (препроцесингу) вагові коефіцієнти автоенкодера та статичне порогове значення залишаються актуальними та коректними.

Експериментальні дослідження підтверджують високу ефективність підходу: впровадження нормалізації TSNFA зводить кількість кластерів хибних спрацьовувань із 17,95 на годину до абсолютного 0,00. Це знижує навантаження на мережу у 260 разів (запобігаючи переповненню критичних лімітів мережі, що аналогічно збереженню об'єму MAX_BUFFER у змодельованому Fog-сервері). При цьому зберігається висока збалансована точність класифікації сигналів (96,0 % порівняно з 83,1 % у базових детекторах). [24].

Для прийняття остаточного рішення щодо наявності атаки або аномалії система застосовує зважену агрегацію різних оцінок відхилення (Anomaly Scores), що на практиці формує базис для механізмів проактивного QoS (пріоритезації критичного трафіку над рутинною телеметрією). Похибка

реконструкції методу PCA, оцінка нелінійного відхилення моделі DAE та метрика ізольованого лісу (Isolation Forest) комбінуються в загальну реляційну матрицю. З метою визначення ваги кожного параметра застосовується алгоритм експоненційного згладжування (Exponential Weighted Moving Average, EWMA), після чого фінальний індекс працездатності («здоров'я») системи розраховується на основі багатокритеріального методу TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution). Така комплексна архітектура гарантує високу надійність системи та стійкість маршрутизації за будь-яких умов зв'язку, що кількісно підтверджується такими метриками, як крива точності-повноти (Precision-Recall Curve, PRC) та F1-оцінка. Їх використання є критично важливим в умовах суттєво незбалансованих наборів даних, де шкідливий трафік становить лише частки відсотка від загального обсягу інформації. [16].

ЗАГАЛЬНІ ВИСНОВКИ

Традиційні архітектури систем виявлення вторгнень (IDS), які покладаються на хмарні обчислення, є неефективними для індустріального Інтернету речей (IoT). Безперервна передача даних на центральні сервери спричиняє неприпустимі затримки у реагуванні на кібератаки, перевантажує пропускну здатність каналів зв'язку та швидко виснажує батареї автономних пристроїв. Вирішенням цієї проблеми є перенесення інтелектуальної фільтрації безпосередньо на периферійний рівень (Edge Computing) за допомогою технологій TinyML.

У процесі експлуатації датчики зазнають механічного зносу або впливу зовнішніх умов, що змінює їх базовий нормальний стан і призводить до масових хибних спрацьовувань класичних систем. Інтеграція в розроблену архітектуру механізму часової спектральної адаптації рівня шуму (TSNFA) дозволила повністю усунути хибні тривоги (зниживши їх до нуля) і зберегти збалансовану точність класифікації на рівні 96,0% (проти 83,1% у базових моделях).

У роботі запропоновано та обґрунтовано модифікований каскадний (двоетапний) метод виявлення аномалій. Він поєднує метод головних компонент (PCA), що використовується як лінійний фільтр для відсікання базового апаратного шуму, та глибокі нейронні мережі (автоенкодери) для нелінійного виявлення прихованих і складних загроз (наприклад, атак нульового дня). Така гібридизація дозволяє запускати алгоритм на мікроконтролерах із суворими апаратними обмеженнями без втрати точності виявлення.

Результати програмної симуляції емпірично підтверджують ключову гіпотезу дослідження: децентралізація обчислень шляхом перенесення завдань інтелектуальної фільтрації на Edge-рівень у поєднанні з алгоритмами адаптивного управління якістю обслуговування (QoS) на проміжному Fog-

сервері забезпечує безперервність та високу відмовостійкість систем безпекового моніторингу. Встановлено, що навіть в умовах деградації або втрати основного широкосмугового каналу зв'язку, система зберігає здатність у режимі реального часу передавати критичні сповіщення про кібервторгнення та агреговані оновлення для нейромережових моделей, ефективно нівелюючи фоновий сенсорний шум. Отримані дані обґрунтовують практичну доцільність, масштабованість та високу комунікаційну ефективність розробленої гібридної архітектури.

Роботу виконано в рамках НДР кафедри інформаційних технологій в телекомунікаціях НН ІТС № 0124U001559 "Технології Інтернету Речей для розумного будинку та розумного міста". Тези за результатами дослідження було подано за темою «Аномальні дані мережі індустріального інтернету речей» на XX Міжнародну науково-технічну конференцію «Перспективи телекомунікацій 2026», на базі якої проводилась Міжнародна науково-технічна конференція студентів та аспірантів «Перспективи розвитку інформаційно-телекомунікаційних технологій та систем». [42].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ali B., Awad A. Cyber and physical security vulnerability assessment for iot-based smart homes. *Sensors*. 2018. Vol. 18, no. 3. P. 817. URL: <https://doi.org/10.3390/s18030817>.
2. Security of IoT devices DoS attacks: Security Threats on IoT layers. URL: <https://www.diva-portal.org/smash/get/diva2:1887790/FULLTEXT01.pdf>.
3. What is iot security? A complete overview | splunk. Splunk. URL: https://www.splunk.com/en_us/blog/learn/iot-security.html.
4. Sebestyen H., Popescu D. E., Zmaranda R. D. A literature review on security in the internet of things: identifying and analysing critical categories. *Computers*. 2025. Vol. 14, no. 2. P. 61. URL: <https://doi.org/10.3390/computers14020061>.
5. IoT Security Architecture: Best Practices for Protecting Connected Devices | AdaptNXT Blog. *AdaptNXT*. URL: <https://www.adaptnxt.com/blogs/iot-security-architecture-best-practices>.
6. IoT Security Architecture: Best Practices for Protecting Connected Devices | AdaptNXT Blog. *AdaptNXT*. URL: <https://www.adaptnxt.com/blogs/iot-security-architecture-best-practices>.
7. SAFE: Self-Supervised Anomaly Detection Framework for Intrusion Detection. *arXiv.org e-Print archive*. URL: <https://arxiv.org/html/2502.07119v1>.
8. Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering / U. Ahmed et al. *Scientific Reports*. 2025. Vol. 15, no. 1. URL: <https://doi.org/10.1038/s41598-025-85866-7>.
9. N-able. Intrusion Detection System (IDS): Signature vs. Anomaly-Based. URL: <https://www.n-able.com/blog/intrusion-detection-system>.
10. Hariharasubramanian N. Signature-Based IDS vs Anomaly-Based IDS: Guide to Choose the Right Detection Model. *fidelissecurity*. URL: <https://fidelissecurity.com/cybersecurity-101/learn/signature-based-vs-anomaly-based-ids/>.

11. Luca C. Anomaly-Based vs Signature-Based IDS: A Machine Learning Perspective. URL: https://www.researchgate.net/publication/403513853_Anomaly-Based_vs_Signature-Based_IDS_A_Machine_Learning_Perspective.
12. Real-Time Machine Learning for Embedded Anomaly Detection. *arXiv.org e-Print archive*. URL: <https://arxiv.org/html/2512.19383v1>.
13. Efficient Real-Time Anomaly Detection in IoT Networks Using One-Class Autoencoder and Deep Neural Network / A. G. Ayad et al. *Electronics*. 2024. Vol. 14, no. 1. P. 104. URL: <https://doi.org/10.3390/electronics14010104>.
14. Intrusion Detection in IOT systems using Denoising Autoencoder / F. S. Alrayes et al. *IEEE Access*. 2024. P. 1. URL: <https://doi.org/10.1109/access.2024.3451726>.
16. Anomaly Detection in Industrial Machinery using IoT Devices and Machine Learning: a Systematic Mapping / S. F. Chevtchenko et al. *IEEE Access*. 2023. P. 1. URL: <https://doi.org/10.1109/access.2023.3333242>.
17. Motor PHM on Edge Computing with Anomaly Detection and Fault Severity Estimation through Compressed Data Using PCA and Autoencoder / J. H. Choi et al. *Machine Learning and Knowledge Extraction*. 2024. Vol. 6, no. 3. P. 1466–1483. URL: <https://doi.org/10.3390/make6030069>.
18. Kham S. Autoencoder-Based Representation Learning to Predict Anomalies in Computer Networks. *The Repository at St. Cloud State*. URL: https://repository.stcloudstate.edu/msia_etds/102/.
19. Albaiz A., Amsaad F. Fully Autonomous Z-Score-Based TinyML Anomaly Detection on Resource-Constrained MCUs using Power Side-Channel Data. *2026 IEEE 2nd International Conference on Secure IoT, Assured and Trusted Computing (SATC)*, Houston, TX, USA, 24–26 March 2026. 2026. P. 1–6. URL: <https://doi.org/10.1109/satc69565.2026.11542250>.
20. A scalable hybrid autoencoder–extreme learning machine framework for adaptive intrusion detection in high-dimensional networks / A. Kumar et al. *Future internet*. 2025. Vol. 17, no. 5. P. 221. URL: <https://doi.org/10.3390/fi17050221>.

21. Francesca Brescia G. Deploying neural networks on microcontrollers with tinyml. URL: <https://www.embedded.com/deploying-neural-networks-on-microcontrollers-with-tinyml/>.
 22. Mandal S. TinyML: running deep learning models on microcontrollers. URL: <https://medium.com/@sucheta963/tinyml-running-deep-learning-models-on-microcontrollers-a1524a69e98c>.
 23. Makovetskyi S., Thomsen L. A tsnfa-tinyml hybrid algorithm that achieves full suppression of false positives and signal classification under drifting noise. Preprints, 2026. (Preprint). URL: <https://doi.org/10.20944/preprints202605.1955.v1>.
 24. Kennedy W. B. A simple example using PCA for outlier detection. *Medium*. URL: <https://medium.com/data-science/a-simple-example-using-pca-for-outlier-detection-ab2773b98e4a>.
 25. Principal component analysis part 1: the different formulations. *Towards Data Science*. URL: <https://towardsdatascience.com/principal-component-analysis-part-1-the-different-formulations-6508f63a5553/>.
 26. Takeishi N. Shapley values of reconstruction errors of PCA for explaining anomaly detection. *arxiv*. URL: <https://arxiv.org/pdf/1909.03495>.
 27. Čikovský S., Šváb P., Hanák P. Hybrid pca-based and machine learning approaches for signal-based interference detection and anomaly classification under synthetic data conditions. *Sensors*. 2025. Vol. 25, no. 24. P. 7585. URL: <https://doi.org/10.3390/s25247585>.
 28. Bilal M., Shehzad Hanif M. Fast anomaly detection for vision-based industrial inspection using cascades of null subspace PCA detectors. *Sensors*. 2025. Vol. 25, no. 15. URL: <https://doi.org/10.3390/s25154853>.
 29. Probabilistic autoencoder. *Venues* | *OpenReview*. URL: <https://openreview.net/forum?id=AEoYjvjKVA>.
- Product-Aware deep autoencoders for robust process monitoring in multi-product cyber-physical systems. *arXiv.org e-Print archive*. URL: <https://arxiv.org/html/2606.00052v1>.

30. Song Y., Hyun S., Cheong Y.-G. Analysis of autoencoders for network intrusion detection. *Sensors*. 2021. Vol. 21, no. 13. P. 4294. URL: <https://doi.org/10.3390/s21134294>.
31. Sima D. Reconstruction-based anomaly detection on financial market data. *POLITECNICO MILANO*. URL: https://www.politesi.polimi.it/retrieve/a414fb9f-fc74-4c8b-a12b-64ae24e16320/Daniel_Sima_Thesis.pdf.
32. Chang C.-H. Cascaded machine learning models for anomaly detection at the edge. optimizing computational efficiency in multi-stage anomaly detection for edge devices. *KTH Vetenskap och konst*. URL: <https://www.diva-portal.org/smash/get/diva2:1987621/FULLTEXT01.pdf>.
33. TinyML autoencoder-based on-board denoising and drift detection in electrochemical sensors / A. Kia et al. Preprints.org, 2026. 12 p. (Preprint). URL: <https://doi.org/10.20944/preprints202604.2146.v1>.
34. Leena Jenifer L., Devaki K. An intelligent model of anomaly detection and anomaly localization in images using hybrid heuristic adaptive multiscale attention based densenet and cascaded variational autoencoder. *International journal of pattern recognition and artificial intelligence*. 2024. URL: <https://doi.org/10.1142/s0218001424540077>.
35. TinyOL: TinyML with Online-Learning on Microcontrollers. *arxiv*. URL: <https://arxiv.org/pdf/2103.08295>.
36. Widartha V. P., Kim C. S. HIPER-CHAD: hybrid integrated prediction-error reconstruction-based anomaly detection for multivariate indoor environmental time-series data. *Sensors*. 2025. Vol. 26, no. 1. P. 171. URL: <https://doi.org/10.3390/s26010171>.
37. Ji C., Liu H., Dai W. Hybrid model for network traffic anomaly detection based on parallel two-stage feature fusion. *IEEE access*. 2025. P. 1. URL: <https://doi.org/10.1109/access.2025.3538319>.
38. Cascaded dimension reduction for effective anomaly detection / X. Zhuo et al. 2021 *IEEE international conference on big data (big data)*, Orlando, FL, USA,

URL: <https://doi.org/10.1109/bigdata52589.2021.9671364>.

39. APT-Attack detection based on multi-stage autoencoders / H. Neuschmied et al. *Applied sciences*. 2022. Vol. 12, no. 13. P. 6816.

URL: <https://doi.org/10.3390/app12136816>.

40. A novel two-stage deep learning structure for network flow anomaly detection / M.-T. Kao et al. *Electronics*. 2022. Vol. 11, no. 10. P. 1531.

URL: <https://doi.org/10.3390/electronics11101531>.

41. Zhang H., Zhang H. Anomaly detection and objective security evaluation using autoencoder, isolation forest, and multi-criteria decision methods. *Sensors*. 2025. Vol. 25, no. 19. P. 6250. URL: <https://doi.org/10.3390/s25196250>.

42. Гунін І. АНОМАЛЬНІ ДАНІ МЕРЕЖІ ІНДУСТРІАЛЬНОГО ІНТЕРНЕТУ РЕЧЕЙ. XX Міжнародна науково-технічна конференція "Перспективи телекомунікацій 2026" : Міжнар. науково-техн. конф., м. Київ /
Наук. керівник В. Курдеча. Київ, 2026.

URL: <https://conferenc.its.kpi.ua/2026/paper/view/34398>.

ВІЗУАЛІЗОВАНИЙ КОД СТОРІНКИ СИМУЛЯЦІЇ НА БАЗІ СЕРЕДОВИЩА GITHUB PAGES

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Smart Office: Гібридна IIoT Архітектура</title>
  <script
src="https://cdn.jsdelivrivr.net/npm/chart.js"></script>
  <style>
    :root {
      --bg-dark: #0f1115;
      --panel-bg: #1a1c23;
      --text-main: #e4e4e6;
      --text-muted: #94a1b2;
      --accent-blue: #3b82f6;
      --accent-red: #ef4444;
      --accent-green: #10b981;
      --accent-orange: #f59e0b;
      --accent-ai: #a855f7;
      --border-color: #2b2b36;
    }
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
      background-color: var(--bg-dark);
      color: var(--text-main);
      margin: 0; padding: 15px;
```

```

        display: flex; justify-content: center; align-items:
center; min-height: 100vh;
    }

```

```

.dashboard {
    background-color: var(--panel-bg);
    border-radius: 12px; width: 100%; max-width: 1100px;
padding: 20px;
    box-shadow: 0 10px 40px rgba(0,0,0,0.7);
    border: 1px solid var(--border-color);
}

```

```

.header { text-align: center; margin-bottom: 15px; font-
weight: bold; font-size: 1.2em; color: var(--accent-blue); text-
transform: uppercase; letter-spacing: 2px;}

```

```

.canvas-container {
    width: 100%; height: 320px; background-color:
#12141a; border-radius: 8px;
    border: 1px solid var(--border-color); margin-
bottom: 15px; position: relative;
}

```

```

#simCanvas { width: 100%; height: 100%; display: block;
}

```

```

.charts-wrapper { display: flex; gap: 15px; margin-
bottom: 15px; }

```

```

.chart-container {
    flex: 1; height: 160px; background-color: #12141a;
border-radius: 8px;
    border: 1px solid var(--border-color); padding:
10px; position: relative;
}

```

```

        .chart-title { text-align: center; font-size: 0.75em;
color: var(--text-muted); text-transform: uppercase; margin-
bottom: 5px;}

```

```

        .controls {
            display: grid; grid-template-columns: 1fr 1fr 1fr;
gap: 15px;

            background-color: #12141a; padding: 15px; border-
radius: 8px; border: 1px solid var(--border-color);
        }

```

```

        .control-panel {
            background: rgba(255,255,255,0.02); padding: 10px;
border-radius: 6px;

            display: flex; flex-direction: column; gap: 10px;
border: 1px solid rgba(255,255,255,0.05);
        }

```

```

        .panel-title { font-size: 0.8em; color: var(--accent-
blue); text-transform: uppercase; font-weight: bold; border-
bottom: 1px solid var(--border-color); padding-bottom: 5px;
margin-bottom: 5px;}

```

```

        .panel-title.ai { color: var(--accent-ai); }

```

```

        .panel-title.net { color: var(--accent-green); }

```

```

        button {
            background-color: var(--border-color); color: var(--
text-main); border: none;

            padding: 8px 12px; border-radius: 4px; cursor:
pointer; font-weight: bold; font-size: 0.85em; transition: 0.2s;
        }

```

```

        button:hover { background-color: #4f5160; }

```

```

        button.active-red { background-color: var(--accent-red);
color: white; }

```

```

        button.active-orange { background-color: var(--accent-
orange); color: white; }

```

```

    input[type=range] {
        flex-grow: 1; margin: 0 10px; accent-color: var(--
accent-orange);
    }

    .status-badge { display: inline-block; padding: 3px 8px;
border-radius: 12px; font-size: 0.75em; font-weight: bold; }

    .bg-green { background: rgba(16, 185, 129, 0.2); color:
var(--accent-green); }

    .bg-red { background: rgba(239, 68, 68, 0.2); color:
var(--accent-red); }

    .bg-blue { background: rgba(59, 130, 246, 0.2); color:
var(--accent-blue); }

    .bg-gray { background: rgba(148, 161, 178, 0.2); color:
var(--text-muted); }

    .flex-between { display: flex; justify-content: space-
between; align-items: center; font-size: 0.85em;}

    .legend { display: flex; gap: 15px; justify-content:
center; font-size: 0.75em; color: var(--text-muted); margin-top:
10px;}

    .leg-item { display: flex; align-items: center; gap:
4px; }

    .dot { width: 8px; height: 8px; border-radius: 50%; }

</style>
</head>
<body>

<div class="dashboard">

    <div class="header">Розумний Офіс: Маршрутизація даних
(Оптоволокно + 4G/5G)</div>

    <div class="canvas-container">

        <canvas id="simCanvas"></canvas>

```

```

</div>

<div class="charts-wrapper">
  <div class="chart-container">
    <div class="chart-title">Дані офісного сенсора</div>
    <canvas id="sensorChart"></canvas>
  </div>
  <div class="chart-container">
    <div class="chart-title">Буфер Маршрутизатора
(Об'єм: 60)</div>
    <canvas id="bufferChart"></canvas>
  </div>
  <div class="chart-container">
    <div class="chart-title">Навантаження на зовнішній
канал</div>
    <canvas id="networkChart"></canvas>
  </div>
</div>

<div class="controls">
  <!-- Panel 1: Edge / AI -->
  <div class="control-panel">
    <div class="panel-title ai">IoT Контролер
(Локально)</div>
    <div style="display: flex; gap: 5px;">
      <button id="edgeBtn" style="flex: 1;">Вимкнути
МК</button>
      <button id="anomalyBtn" style="flex:
1;">Тривога</button>
    </div>
    <div class="flex-between">
      <span>Сила інциденту:</span>
      <input type="range" id="anomalyStrength" min="1"
max="5" step="0.5" value="2.5" disabled>
    </div>
  </div>
</div>

```

```

        <span id="anomalyStrengthVal" style="font-
weight:bold; color:var(--text-main); min-width: 20px; text-
align:right;">2.5</span>

    </div>

    <div class="flex-between">

        <span>Стан обробки (Edge):</span>

        <span class="status-badge bg-green" id="ml-
status">АКТИВНИЙ</span>

    </div>

</div>

<!-- Panel 2: Network Routing -->

<div class="control-panel">

    <div class="panel-title net">Офісний Шлюз (SD-
WAN)</div>

    <button id="linkBtn">Вимкнути провайдера
(Обрив)</button>

    <div class="flex-between">

        <span>Оптоволокно (WAN 1):</span>

        <span class="status-badge bg-green" id="fiber-
status">ОНЛАЙН (1 Гбіт/с)</span>

    </div>

    <div class="flex-between">

        <span>Мобільна мережа (LTE):</span>

        <span class="status-badge bg-blue" id="mobile-
status">ОЧІКУВАННЯ</span>

    </div>

</div>

<!-- Panel 3: Stats -->

<div class="control-panel">

    <div class="panel-title">Метрики Трафіку</div>

    <div class="flex-between"><span>Відкинуто на Edge
(Шум):</span> <strong id="stat-dropped">0</strong></div>

    <div class="flex-between">

```

```

        <span>Втрати буфера (QoS):</span>

        <strong id="stat-router-dropped" style="color:
var(--accent-red)">0</strong>

    </div>

    <div class="flex-between"><span>Пакети в
буфері:</span> <strong id="stat-buffer">0 шт.</strong></div>

    <div class="flex-between"><span>Критичні
події:</span> <strong id="stat-anomalies" style="color: var(--
accent-orange)">0</strong></div>

    </div>

</div>

<div class="legend">

    <div class="leg-item"><div class="dot"
style="background:#fff"></div> Телеметрія</div>

    <div class="leg-item"><div class="dot"
style="background:var(--accent-orange)"></div> Тривога</div>

    <div class="leg-item"><div class="dot"
style="background:var(--accent-ai)"></div> Синхронізація</div>

    <div class="leg-item"><div class="dot"
style="background:var(--accent-red)"></div> Відхилений
Шум/Втрати</div>

    </div>

</div>

<script>

    // --- Canvas Setup ---

    const canvas = document.getElementById('simCanvas');
    const ctx = canvas.getContext('2d');

    let width, height;

    function resizeCanvas() {

        width = canvas.width = canvas.parentElement.clientWidth;

        height = canvas.height =
canvas.parentElement.clientHeight;

    }

    window.addEventListener('resize', resizeCanvas);

```

```

resizeCanvas();

// --- System State ---
let isAnomaly = false;
let anomalyStrength = 2.5;
let fiberUp = true;
let edgeUp = true; // Стан мікроконтролера
let timeT = 0;
const MAX_BUFFER = 60;

// Stats
let statDroppedEdge = 0;
let statDroppedRouter = 0;
let statAnomalies = 0;
let networkThroughput = 0;

// Nodes
const nodes = {
    sensor: { x: 0.1, y: 0.7, label: 'СЕНСОР\n(Кімната)',
type: 'box' },
    edge: { x: 0.35, y: 0.7, label:
'МІКРОКОНТРОЛЕР\n(Фільтрація)', type: 'box' },
    router: { x: 0.6, y: 0.7, label: 'МАРШРУТИЗАТОР\n(SD-
WAN)', type: 'box' },
    cloud: { x: 0.9, y: 0.5, label: 'КОРПОРАТИВНА\nХМАРА',
type: 'box' }
};

const routerBuffer = { alerts: [], sync: [], telemetry: []
};

let packets = [];

class Packet {
    constructor(type, stage, startNode, endNode) {

```

```

        this.type = type;
        this.stage = stage;
        this.x = startNode.x * width;
        this.y = startNode.y * height;
        this.startX = this.x; this.startY = this.y;
        this.targetX = endNode.x * width; this.targetY =
endNode.y * height;
        this.progress = 0;
        this.speed = stage === 3 && !fiberUp ? 0.008 :
0.015;
        this.state = 'moving';
        this.routeType = stage === 3 && !fiberUp ? 'mobile'
: 'fiber';
    }

    update() {
        if (this.state === 'dropping') {
            this.y += 3; this.x += 0.5;
            return this.y > height + 10;
        }

        this.progress += this.speed;
        if (this.progress > 1) this.progress = 1;

        this.x = this.startX + (this.targetX - this.startX)
* this.progress;

        if (this.routeType === 'mobile') {
            const arcHeight = height * 0.25;
            const par = 4 * arcHeight * this.progress * (1 -
this.progress);
            this.y = this.startY + (this.targetY -
this.startY) * this.progress - par;
        } else {

```

```

        this.y = this.startY + (this.targetY -
this.startY) * this.progress;
    }

    if (this.progress >= 1) {
        if (this.stage === 1) this.handleEdgeLogic();
        else if (this.stage === 2)
this.handleRouterLogic();
        else if (this.stage === 3) {
            networkThroughput++;
            return true;
        }
        return true;
    }
    return false;
}

handleEdgeLogic() {
    if (edgeUp) {
        // Мікроконтролер працює: фільтруємо шум
        if (this.type === 'noise') {
            let drop = new Packet('noise', 0,
nodes.edge, nodes.edge);
            drop.state = 'dropping';
            packets.push(drop);
            statDroppedEdge++;
            document.getElementById('stat-
dropped').innerText = statDroppedEdge;
        } else {
            packets.push(new Packet(this.type, 2,
nodes.edge, nodes.router));
            if (this.type === 'alert') {
                statAnomalies++;
            }
        }
    }
}

```

```

        document.getElementById('stat-
anomalies').innerText = statAnomalies;
    }
}
} else {
    // МК вимкнено: пропускаємо ВСЕ на маршрутизатор
    (як 'telemetry', щоб забити буфер)
    // Шуми стають звичайним трафіком, який
    маршрутизатор не вміє розрізняти
    let outType = this.type === 'noise' ?
'telemetry' : this.type;
    packets.push(new Packet(outType, 2, nodes.edge,
nodes.router));
    if (outType === 'alert') {
        statAnomalies++;
        document.getElementById('stat-
anomalies').innerText = statAnomalies;
    }
}
}

handleRouterLogic() {
    let currentTotal = routerBuffer.alerts.length +
routerBuffer.sync.length + routerBuffer.telemetry.length;

    // Жорсткий ліміт буфера маршрутизатора
    if (currentTotal >= MAX_BUFFER) {
        statDroppedRouter++;
        document.getElementById('stat-router-
dropped').innerText = statDroppedRouter;

        // Візуалізуємо втрату пакета (падає від
        роутера)
        let drop = new Packet(this.type, 0,
nodes.router, nodes.router);
        drop.state = 'dropping';

```

```

        packets.push(drop);
        return;
    }

    if (this.type === 'alert')
routerBuffer.alerts.push(this);
        else if (this.type === 'sync')
routerBuffer.sync.push(this);
        else routerBuffer.telemetry.push(this);
    }

    draw(ctx) {
        ctx.beginPath();
        ctx.arc(this.x, this.y, this.type === 'sync' ? 5 :
4, 0, Math.PI * 2);

        if (this.state === 'dropping') ctx.fillStyle =
'#ef4444';
        else if (this.type === 'alert') ctx.fillStyle =
'#f59e0b';
        else if (this.type === 'sync') ctx.fillStyle =
'#a855f7';
        else ctx.fillStyle = 'ffffff';

        ctx.fill();

        if (this.type === 'alert' && this.state !==
'dropping') {
            ctx.strokeStyle = '#fcd34d'; ctx.lineWidth = 1;
ctx.stroke();
        }
    }
}

// --- Node Rendering ---
function drawNode(node, isActive = false) {

```

```

ctx.fillStyle = '#1e2029';

    // Якщо це Edge і він вимкнений, робимо його
    сірим/червоним
    if (node.label.includes('МІКРО') && !edgeUp) {
        ctx.strokeStyle = '#ef4444';
    } else {
        ctx.strokeStyle = isActive ? '#3b82f6' : '#2b2b36';
    }

    ctx.lineWidth = 2;

    const nw = 130; const nh = 45;
    const nx = node.x * width - nw/2; const ny = node.y *
height - nh/2;

    ctx.beginPath(); ctx.roundRect(nx, ny, nw, nh, 6);
    ctx.fill(); ctx.stroke();

    ctx.fillStyle = (node.label.includes('МІКРО') &&
!edgeUp) ? '#ef4444' : '#94a1b2';
    ctx.font = '10px sans-serif';
    ctx.textAlign = 'center'; ctx.textBaseline = 'middle';

    const labelText = (node.label.includes('МІКРО') &&
!edgeUp) ? 'МІКРОКОНТРОЛЕР\n(BYPASS / ВИМК)' : node.label;
    const lines = labelText.split('\n');
    lines.forEach((line, i) => {
        ctx.fillText(line, node.x * width, node.y * height +
(i * 12) - ((lines.length-1)*6));
    });
}

function drawLinks() {

```

```

ctx.lineWidth = 1.5;

// LAN
ctx.strokeStyle = edgeUp ? '#2b2b36' : '#ef4444'; //
Червоний лінк, якщо потоки йдуть сирі
ctx.beginPath();
ctx.moveTo(nodes.sensor.x * width, nodes.sensor.y *
height);
ctx.lineTo(nodes.edge.x * width, nodes.edge.y * height);
ctx.lineTo(nodes.router.x * width, nodes.router.y *
height);
ctx.stroke();

// Fiber
ctx.strokeStyle = fiberUp ? '#10b981' : '#ef4444';
ctx.setLineDash(fiberUp ? [] : [5, 5]);
ctx.beginPath();
ctx.moveTo(nodes.router.x * width, nodes.router.y *
height);
ctx.lineTo(nodes.cloud.x * width, nodes.cloud.y *
height);
ctx.stroke();

ctx.fillStyle = fiberUp ? '#10b981' : '#ef4444';
ctx.fillText("Оптоволокно (Провайдер 1)",
(nodes.router.x + nodes.cloud.x)/2 * width, (nodes.router.y +
nodes.cloud.y)/2 * height - 10);

// 4G
ctx.strokeStyle = !fiberUp ? '#3b82f6' : '#2b2b36';
ctx.setLineDash([4, 4]);
ctx.beginPath();
ctx.moveTo(nodes.router.x * width, nodes.router.y *
height);

```

```

        ctx.quadraticCurveTo((nodes.router.x + nodes.cloud.x)/2
* width, height * 0.2, nodes.cloud.x * width, nodes.cloud.y *
height);

        ctx.stroke();

        ctx.fillStyle = !fiberUp ? '#3b82f6' : '#4f5160';

        ctx.fillText("4G/5G Вишка (Резерв)", (nodes.router.x +
nodes.cloud.x)/2 * width, height * 0.35);

        ctx.setLineDash([]);

    }

    // --- Charts ---

    Chart.defaults.color = '#94a1b2'; Chart.defaults.font.size =
9;

    const chartOpts = { responsive: true, maintainAspectRatio:
false, animation: false, plugins: { legend: { display: false }
}, scales: { x: { display: false }, y: { grid: { color:
'#2b2b36' } } } };

    const sensorChart = new
Chart(document.getElementById('sensorChart'), {

        type: 'line',

        data: { labels: Array(50).fill(''), datasets: [{
borderColor: '#3b82f6', borderWidth: 1.5, data:
Array(50).fill(0), pointRadius: 0, tension: 0.1 }] },

        options: { ...chartOpts, scales: { ...chartOpts.scales,
y: { min: -150, max: 150, grid: { color: '#2b2b36' } } } }

    });

    // Оновлений динамічний Area Chart для буфера

    const bufferChart = new
Chart(document.getElementById('bufferChart'), {

        type: 'line',

        data: { labels: Array(40).fill(''), datasets: [

            { label: 'Телеметрія/Шум', borderColor: '#3f414d',
backgroundColor: 'rgba(63, 65, 77, 0.6)', fill: true, data:
Array(40).fill(0), pointRadius: 0, tension: 0.3 },

```

```

        { label: 'Тривори', borderColor: '#f59e0b',
        backgroundColor: 'rgba(245, 158, 11, 0.6)', fill: true, data:
        Array(40).fill(0), pointRadius: 0, tension: 0.3 }

    ]},

    options: { ...chartOpts, scales: { x: { display: false
    }, y: { stacked: true, min: 0, max: MAX_BUFFER, grid: { color:
    '#2b2b36' } } } }

    });

    const netChart = new
    Chart(document.getElementById('networkChart'), {

        type: 'line',

        data: { labels: Array(40).fill(''), datasets: [{
        borderColor: '#10b981', backgroundColor: 'rgba(16, 185, 129,
        0.1)', fill:true, borderWidth: 2, data: Array(40).fill(0),
        pointRadius: 0, tension: 0.2 }] },

        options: { ...chartOpts, scales: { ...chartOpts.scales,
        y: { min: 0, max: 30, grid: { color: '#2b2b36' } } } }

    });

    let frame = 0;

    function animate() {

        ctx.clearRect(0, 0, width, height);

        drawLinks();

        drawNode(nodes.sensor);

        drawNode(nodes.edge, edgeUp && (frame % 120 < 10));

        drawNode(nodes.router, routerBuffer.alerts.length > 0 ||
        routerBuffer.telemetry.length > 0);

        drawNode(nodes.cloud);

        packets = packets.filter(p => { p.draw(ctx); return
        !p.update(); });

        // Відправка з маршрутизатора

```

```

    let txCapacity = fiberUp ? 3 : (frame % 2 === 0 ? 1 :
0);

    for (let i = 0; i < txCapacity; i++) {
        let nextPacket = null;

        if (routerBuffer.sync.length > 0) nextPacket =
routerBuffer.sync.shift();

        else if (routerBuffer.alerts.length > 0) nextPacket
= routerBuffer.alerts.shift();

        else if (routerBuffer.telemetry.length > 0) {
            // Проактивний QoS: якщо канал вузький (4G) і
черга росте, відкидаємо менш важливі дані
            if (!fiberUp && routerBuffer.telemetry.length >
15) {

                let droppedPkt =
routerBuffer.telemetry.shift();

                statDroppedRouter++;

                document.getElementById('stat-router-
dropped').innerText = statDroppedRouter;

                let dropVis = new Packet(droppedPkt.type, 0,
nodes.router, nodes.router);

                dropVis.state = 'dropping';

                packets.push(dropVis);

                continue;

            }

            nextPacket = routerBuffer.telemetry.shift();

        }

        if (nextPacket) packets.push(new
Packet(nextPacket.type, 3, nodes.router, nodes.cloud));

    }

    let totalBuffered = routerBuffer.telemetry.length +
routerBuffer.alerts.length;

```

```

        document.getElementById('stat-buffer').innerText =
`$${totalBuffered} шт.`;

        // Візуалізація критичного переповнення
        if(totalBuffered >= MAX_BUFFER - 5) {
            ctx.fillStyle = '#ef4444'; ctx.font = 'bold 12px
sans-serif';

            ctx.fillText("OVERFLOW: ВТРАТА ДАНИХ",
nodes.router.x * width, nodes.router.y * height - 35);

        }

        frame++;

        requestAnimationFrame(animate);
    }

    // Data Generator
    setInterval(() => {
        timeT += 0.2;

        let signal = Math.sin(timeT) * 30;
        let isSpike = false;
        let isNoise = Math.random() < 0.4;

        if (isAnomaly) {
            let spikeProb = 0.05 + (anomalyStrength * 0.05);
            if (Math.random() < spikeProb) {
                let spikeAmp = 40 + (anomalyStrength * 20);
                signal += (Math.random() > 0.5 ? spikeAmp : -
spikeAmp);

                isSpike = true;

                ctx.fillStyle = `rgba(245, 158, 11, ${0.1 +
(anomalyStrength * 0.05)})`;

                ctx.beginPath(); ctx.arc(nodes.sensor.x * width,
nodes.sensor.y * height, 20 + (anomalyStrength * 4), 0,
Math.PI*2); ctx.fill();

```

```

        } else {
            isNoise = Math.random() < (0.3 +
anomalyStrength*0.1);
            if (isNoise) signal += (Math.random() - 0.5) *
15 * anomalyStrength;
        }
    } else if (isNoise) {
        signal += (Math.random() - 0.5) * 15;
    }

    sensorChart.data.datasets[0].data.push(signal);
    sensorChart.data.datasets[0].data.shift();
    sensorChart.data.datasets[0].borderColor = isSpike ?
'#f59e0b' : '#3b82f6';
    sensorChart.update();

    if (isSpike) {
        let burstSize = Math.ceil(anomalyStrength / 1.5);
        for(let i=0; i<burstSize; i++){
            packets.push(new Packet('alert', 1,
nodes.sensor, nodes.edge));
        }
    } else if (isNoise) {
        // Генеруємо 2 пакети шуму для більшого навантаження
при Bypass режимі
        packets.push(new Packet('noise', 1, nodes.sensor,
nodes.edge));
        packets.push(new Packet('noise', 1, nodes.sensor,
nodes.edge));
    } else if (Math.random() < 0.3) {
        packets.push(new Packet('telemetry', 1,
nodes.sensor, nodes.edge));
    }
}, 100);

```

```

// Sync Tick (Тільки якщо МК увімкнений)
setInterval(() => {
    if (!edgeUp) return;

    packets.push(new Packet('sync', 1, nodes.edge,
nodes.edge));

    packets[packets.length-1].handleEdgeLogic();

}, 3000);

// Chart Updates
setInterval(() => {

bufferChart.data.datasets[0].data.push(routerBuffer.telemetry.le
ngth);

    bufferChart.data.datasets[0].data.shift();

bufferChart.data.datasets[1].data.push(routerBuffer.alerts.lengt
h);

    bufferChart.data.datasets[1].data.shift();
    bufferChart.update();

    netChart.data.datasets[0].data.push(networkThroughput);
    netChart.data.datasets[0].data.shift();
    netChart.data.datasets[0].borderColor = fiberUp ?
'#10b981' : '#3b82f6';
    netChart.data.datasets[0].backgroundColor = fiberUp ?
'rgba(16, 185, 129, 0.1)' : 'rgba(59, 130, 246, 0.1)';
    netChart.update();
    networkThroughput = 0;

}, 500);

// --- Controls ---

const anomalyStrengthSlider =
document.getElementById('anomalyStrength');

const anomalyStrengthVal =
document.getElementById('anomalyStrengthVal');

```

```

    anomalyStrengthSlider.addEventListener('input', (e) => {
        anomalyStrength = parseFloat(e.target.value);
        anomalyStrengthVal.innerText =
anomalyStrength.toFixed(1);
    });

    document.getElementById('edgeBtn').addEventListener('click',
(e) => {
        edgeUp = !edgeUp;
        const statusEl = document.getElementById('ml-status');

        if (!edgeUp) {
            e.target.innerText = 'УВІМКНУТИ МК';
            e.target.classList.add('active-red');
            statusEl.className = 'status-badge bg-red';
            statusEl.innerText = 'BYPASS (ВИМКНЕНО)';
        } else {
            e.target.innerText = 'ВИМКНУТИ МК';
            e.target.classList.remove('active-red');
            statusEl.className = 'status-badge bg-green';
            statusEl.innerText = 'АКТИВНИЙ';
        }
    });

    document.getElementById('anomalyBtn').addEventListener('click',
(e) => {
        isAnomaly = !isAnomaly;
        anomalyStrengthSlider.disabled = !isAnomaly;

        if (isAnomaly) {
            e.target.innerText = ' Скинути';
            e.target.classList.add('active-orange');
        }
    });

```

```

    } else {
        e.target.innerText = 'Тривога';
        e.target.classList.remove('active-orange');
    }
});

document.getElementById('linkBtn').addEventListener('click',
(e) => {
    fiberUp = !fiberUp;
    const fStatus = document.getElementById('fiber-status');
    const mStatus = document.getElementById('mobile-
status');

    if (!fiberUp) {
        e.target.innerText = 'Відновити оптоволокну';
        e.target.classList.add('active-red');
        fStatus.className = 'status-badge bg-red';
        fStatus.innerText = 'ОФЛАЙН (АВАРІЯ)';
        mStatus.className = 'status-badge bg-blue';
        mStatus.innerText = 'АКТИВНИЙ (ТРАФІК ЛІМІТ)';
    } else {
        e.target.innerText = 'Вимкнути провайдера (Обрив)';
        e.target.classList.remove('active-red');
        fStatus.className = 'status-badge bg-green';
        fStatus.innerText = 'ОНЛАЙН (1 Гбіт/с)';
        mStatus.className = 'status-badge bg-blue';
        mStatus.innerText = 'ОЧІКУВАННЯ';
    }
});

animate();
</script>
</body>
</html>

```