

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Міністерство освіти і науки України

Кваліфікаційна наукова
праця на правах рукопису

ГАЙДАЙ АНАТОЛІЙ РУСЛАНОВИЧ

УДК 004.75 (004.62)

ДИСЕРТАЦІЯ
МЕТОД ТА ЗАСОБИ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ
В МЕРЕЖАХ ІНТЕРНЕТ РЕЧЕЙ
123 Комп'ютерна інженерія
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ А. Р. Гайдай

Науковий керівник: Клименко Ірина Анатоліївна, д. т. н., професор

Київ – 2026

АНОТАЦІЯ

Гайдай А. Р. Метод та засоби інтелектуальної обробки даних в мережах Інтернет речей. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 – Комп’ютерна інженерія з галузі знань 12 – Інформаційні технології. – Національний Технічний Університет України «Київський Політехнічний Інститут імені Ігоря Сікорського», Київ, 2026.

Дисертаційна робота присвячена розв’язанню актуальної науково-практичної задачі підвищення ефективності обробки потоків даних та функціонування інфраструктури на граничному рівні розподілених систем Інтернету речей (IoT), що функціонують в режимі реального часу та за умов надходження великої кількості різнорідних сенсорних даних, багатопоточності обробки та масштабування інфраструктури. У роботі обґрунтовано системний підхід до адаптивної інтелектуальної обробки даних на граничному рівні розподілених систем IoT, що ґрунтується на формуванні адаптивно-інтелектуально керованої гібридної інфраструктури IoT, у межах якої процеси обробки потоків даних, моніторингу даних та стану ресурсів, інтелектуальної аналітики та забезпечення ефективності інфраструктури інтегровані в єдиний керуючий контур.

Актуальність дослідження зумовлена активним розвитком промислових високотехнологічних та критичних систем, де широко застосовуються системи IoT. У таких системах функціонують тисячі сенсорів і вимірювальних пристроїв, які безперервно генерують значні обсяги даних, формуючи надвеликі потоки інформації, що потребують ефективної адаптивної обробки та інтелектуальної інтеграції ресурсів.

Збільшення обсягів сенсорних даних у сучасних системах IoT робить традиційне передавання необроблених даних до централізованих хмарних систем все менш ефективним, оскільки це призводить до зростання навантаження на мережеву інфраструктуру, збільшення затримок обробки та обмеження масштабованості. Перехід до граничних обчислень (*edge computing*) частково

вирішує ці проблеми. Проте актуальності набуває проблема обробки цих даних у режимі реального часу, оскільки значна кількість алгоритмів обробки та керування виконується на пристроях з обмеженими апаратними ресурсами, таких як мікроконтролери, одноплатні мінікомп'ютери та інші спеціалізовані обчислювальні платформи. Додатковою складністю є те, що сенсорні дані часто характеризуються низькою якістю та залежать від динаміки стану оточуючого середовища, включаючи, наприклад, сезонні, технологічні та експлуатаційні коливання, що ускладнює їх аналіз і потребує застосування ефективних методів попередньої обробки та корекції.

Традиційні методи обробки потоків даних базуються на централізованій архітектурі та підходах до організації системи керування інфраструктурою і орієнтовані на обробку наперед визначених подій. Така організація обмежує оперативність прийняття рішень, знижує масштабованість і не дозволяє ефективно реагувати на динамічні зміни стану інфраструктури. В умовах зростання обсягів сенсорних даних та обмежених ресурсів граничних пристроїв виникає необхідність переходу до гібридної інфраструктури з розподіленим та централізованим керуванням і забезпечує автоматизацію, адаптивність та інтелектуальність процесів. Такий підхід дозволяє інтегрувати ключові функції обробки, моніторингу та аналітики у єдиний керуючий контур, підвищуючи ефективність, безпеку та гнучкість систем IoT у режимі реального часу.

Метою роботи є підвищення ефективності обробки потоків даних та функціонування інфраструктури на граничному рівні розподілених систем IoT, що функціонують в режимі реального часу та за умов надходження великої кількості різномірних сенсорних вимірювань, багатопоточності обробки та масштабування інфраструктури. Для досягнення поставленої мети були обґрунтовані принципи системного підходу до адаптивної інтелектуальної обробки даних на основі формування адаптивно-інтелектуальної керованої гібридної інфраструктури розподіленої системи IoT; формалізовані критерії ефективності обробки даних та функціонування інфраструктури IoT; розроблено новий комплексний метод адаптивної інтелектуальної обробки даних у розподілених системах IoT, який

базується на інтеграції взаємопов'язаних часткових методів та засобів для підвищення достовірності сенсорних вимірювань, ефективного керування потоками даних, балансування обчислювального навантаження та забезпечення безпеки функціонування інфраструктури IoT; сформована методика практичної реалізації запропонованого методу; розроблені арифметичні розширювачі на гетерогенній реконфігуровній платформі SoC/ПЛІС, зокрема, для реалізації багаторівневого конвеєра фільтрації сенсорних даних та нейромережевого модуля прийняття рішень; реалізовані механізми взаємодії між вбудованим процесором ARM і програмно-конфігурованою логікою ПЛІС, що забезпечують динамічну реконфігурацію ПЛІС; проведені експериментальні дослідження для оцінки ефективності запропонованих рішень для забезпечення обробки потоків даних на граничному рівні інфраструктури IoT.

У межах дослідження розроблено формальні моделі обробки потоків сенсорних даних, на основі яких проведено аналіз ефективності запропонованого методу, що разом із результатами експериментальних досліджень на реальному устаткуванні підтвердив доцільність його застосування та практичну ефективність у умовах адаптивної інтелектуальної обробки даних на граничному рівні IoT.

Наукова новизна отриманих результатів полягає у розробленні комплексного методу адаптивної інтелектуальної обробки даних у розподілених системах IoT, який відрізняється від відомих поєднанням сукупності взаємопов'язаних часткових методів і засобів у межах єдиного керуючого контуру для забезпечення динамічного керування потоками даних та реконфігурації обчислювальних вузлів на ПЛІС з ціллю забезпечення оперативних керуючих впливів, підвищення ефективності використання обчислювальних ресурсів і безпеки функціонування інфраструктури.

Удосконалено спосіб адаптивного налаштування порогів контролю у моніторингових системах, який на відміну від відомих підходів із фіксованими пороговими значеннями поєднує механізми вирівнювання сигналу, виділення тренду та адаптивного самоналаштування порогів, що дозволяє зменшити кількість хибних спрацювань і підвищити достовірність виявлення аномалій.

Запропоновано спосіб динамічної реконфігурації обчислювальних вузлів на гетерогенних платформах SoC/ПЛІС, який забезпечує ефективну інтеграцію реконфігурованої логіки з вбудованим процесором ARM та дозволяє динамічно перерозподіляти обчислювальні задачі між апаратною і програмною складовими системи залежно від навантаження та класу вирішуваних задач. Така інтеграція створює умови для реалізації динамічних алгоритмів обробки сенсорних потоків у реальному часі на периферії системи.

Отримав подальшого розвитку метод автоматичного прийняття рішень у безпекових сценаріях розподілених систем IoT, який відрізняється від відомих інтеграцією трьох компонентів: безперервного моніторингу стану системи з виконанням формалізованих алгоритмів аналізу аномальних подій та автоматизованим виконання керуючих впливів.

Запропоновано спосіб інтелектуальної нейромережевої обробки сенсорних даних на ПЛІС, який поєднує апаратну стабілізацію вхідних даних із нейромережевим прийняттям рішень у межах єдиного обчислювального контуру, що дозволяє підвищити ефективність обробки даних завдяки апаратній реалізації нейромережі для прийняття рішень у реальному часі на периферії IoT.

Практичне значення отриманих результатів полягає у створенні програмно-апаратної платформи адаптивної інтелектуальної обробки сенсорних даних, що включає три взаємопов'язані компоненти: апаратний багаторівневий конвеєр на ПЛІС для фільтрації даних та отримання трендів, необхідних для реалізації адаптивних порогів; інтелектуальний нейромережевий модуль для прийняття рішень на основі оброблених сенсорних сигналів; автоматизоване адаптивне керування інфраструктурою IoT, що забезпечує динамічне перерозподілення обчислювальних ресурсів і оптимізацію потоків обробки даних».

Запропонований метод досліджено з використанням гетерогенної платформи SoC/ПЛІС Cyclone V. Експерименти проводились із використанням реальних потоків сенсорних даних (температура, вологість, освітленість та концентрація CO₂) і підтвердили, що застосування апаратних фільтрів з адаптивними порогоми контролю підвищує якість сенсорних вимірювань, зменшує затримки обробки та

формування керуючих впливів та кількість хибних реакцій, підвищує енергоефективність порівняно з виключно програмною реалізацією на мінікомп'ютерах.

Запропоновані метод та засоби можуть бути використані в промислових і високотехнологічних системах, включаючи підприємства, виробництва, критичні технологічні об'єкти, а також у системи промислового IoT та розумних містах, для підвищення якості моніторингу даних та ресурсів, безпеки, адаптивного балансування навантаження та інтеграції нейронних і реконфігурованих обчислень на ПЛС.

Основні положення та результати дисертаційного дослідження пройшли апробацію на міжнародних науково-технічних конференціях, а також обговорювалися на наукових семінарах профільної кафедри. За результатами дослідження опубліковано наукові праці, які повною мірою відображають зміст дисертації, зокрема статті у фахових виданнях України та тези доповідей. Дисертаційна робота оформлена відповідно до державних стандартів і складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел та додатків.

Ключові слова: розподілені системи, Інтернет речей, граничні обчислення, локальні мережі, балансування навантаження, безпека, системи-на-кристалі, ПЛС, реконфігурація ПЛС, інтелектуальна обробка даних, якість сенсорних даних, фільтрація сенсорних даних, адаптивні пороги контролю, нейронний обчислювач, моніторингові системи.

ABSTRACT

Haidai A.R. Method and Tools for Intelligent Data Processing in Internet of Things Networks. – Qualifying scientific work on manuscript rights.

Dissertation for obtaining the scientific degree of Doctor of Philosophy in specialty 123 – Computer engineering from the field of knowledge 12 – Information technologies. – National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Kyiv, 2026.

The dissertation is devoted to solving a scientific and practical problem of improving the efficiency of data stream processing and the functioning of infrastructure at the edge level of distributed Internet of Things (IoT) systems operating in real time under conditions of large volumes of heterogeneous sensor data, multithreaded processing, and infrastructure scalability.

The study substantiates a systemic approach to adaptive intelligent data processing at the edge level of distributed IoT systems, based on the formation of an adaptively intelligent managed hybrid IoT infrastructure. Within this infrastructure, data stream processing, data and resource monitoring, intelligent analytics, and infrastructure efficiency assurance are integrated into a unified control loop.

The relevance of the research is driven by the rapid development of industrial, high-tech, and critical systems where IoT technologies are widely applied. Such systems involve thousands of sensors and measurement devices continuously generating large volumes of data, forming massive information flows that require efficient adaptive processing and intelligent resource integration.

The growth in sensor data volumes in modern IoT systems makes traditional transmission of raw data to centralized cloud systems increasingly inefficient, as it leads to increased network load, higher processing latency, and limited scalability. The transition to edge computing partially addresses these issues. However, the problem of real-time data processing remains actual, since many processing and control algorithms are executed on devices with limited hardware resources, such as microcontrollers, single-board computers, and other specialized computing platforms. An additional

challenge is that sensor data are often of low quality and depend on environmental dynamics, including seasonal, technological, and operational variations, which complicates analysis and necessitates effective preprocessing and correction methods.

Traditional data stream processing methods are based on centralized architectures and predefined event-driven approaches to infrastructure management. Such organization limits decision-making responsiveness, reduces scalability, and does not allow effective reaction to dynamic infrastructure changes. Under conditions of increasing data volumes and limited edge resources, there is a need to transition to a hybrid infrastructure combining distributed and centralized control, enabling automation, adaptability, and intelligence of processes. This approach allows integrating key functions of processing, monitoring, and analytics into a unified control loop, increasing efficiency, security, and flexibility of IoT systems in real time.

The aim of the work is to improve the efficiency of data stream processing and infrastructure functioning at the edge level of distributed IoT systems operating in real time under conditions of large volumes of heterogeneous sensor measurements, multithreaded processing, and infrastructure scalability. To achieve this goal, the following were accomplished: substantiation of principles of a systemic approach to adaptive intelligent data processing based on the formation of an adaptively intelligent managed hybrid IoT infrastructure; formalization of efficiency criteria for data processing and IoT infrastructure functioning; development of a new comprehensive method for adaptive intelligent data processing in distributed IoT systems based on the integration of interrelated partial methods and tools to improve sensor data reliability, efficient data flow management, load balancing, and infrastructure security; development of a methodology for practical implementation of the proposed method; design of arithmetic accelerators on a heterogeneous reconfigurable SoC/FPGA platform, including implementation of a multi-level sensor data filtering pipeline and a neural network decision-making module; implementation of interaction mechanisms between the embedded ARM processor and FPGA programmable logic enabling dynamic FPGA reconfiguration; and conducting experimental studies to evaluate the effectiveness of the proposed solutions for edge-level data processing in IoT infrastructure.

Within the study, formal models of sensor data stream processing were developed, based on which the efficiency of the proposed method was analyzed. Together with experimental results obtained on real hardware, this confirmed its feasibility and practical effectiveness under conditions of adaptive intelligent data processing at the IoT edge level.

The scientific novelty of the obtained results lies in the development of a comprehensive method for adaptive intelligent data processing in distributed IoT systems, which differs from existing approaches by combining a set of interrelated partial methods and tools within a unified control loop to ensure dynamic data flow management and reconfiguration of FPGA-based computing nodes, enabling operational control actions, improved resource utilization efficiency, and enhanced infrastructure security.

The method of adaptive adjustment of control thresholds in monitoring systems has been improved. Unlike known approaches with fixed thresholds, it combines signal smoothing, trend extraction, and adaptive self-tuning of thresholds, which reduces false alarms and improves anomaly detection accuracy.

A method for dynamic reconfiguration of computing nodes on heterogeneous SoC/FPGA platforms has been proposed. It ensures effective integration of reconfigurable logic with an embedded ARM processor and enables dynamic redistribution of computational tasks between hardware and software components depending on load and task type. This integration creates conditions for implementing dynamic real-time sensor data processing algorithms at the system edge.

The method of automated decision-making in security scenarios of distributed IoT systems has been further developed. It differs from existing approaches by integrating three components: continuous system monitoring, formalized anomaly detection algorithms, and automated execution of control actions.

A method for intelligent neural network processing of sensor data on FPGA has been proposed, combining hardware stabilization of input data with neural network decision-making within a unified computational loop. This enables increased efficiency of data processing through hardware implementation of neural inference in real time at the IoT edge.

The practical significance of the results lies in the development of a hardware-software platform for adaptive intelligent processing of sensor data, which includes three interconnected components: a hardware multi-level FPGA pipeline for data filtering and trend extraction required for adaptive thresholds; an intelligent neural network module for decision-making based on processed sensor signals; and an automated adaptive IoT infrastructure management system providing dynamic redistribution of computational resources and optimization of data processing flows.

The proposed method was investigated using a heterogeneous SoC/FPGA platform (Cyclone V). Experiments were conducted using real sensor data streams (temperature, humidity, illumination, and CO₂ concentration) and confirmed that the use of hardware filters with adaptive thresholds improves sensor data quality, reduces processing and response latency, decreases false reactions, and increases energy efficiency compared to purely software-based implementations on mini-computers.

The proposed methods and tools can be applied in industrial and high-tech systems, including enterprises, manufacturing, critical technological facilities, as well as industrial IoT and smart city systems, to improve data and resource monitoring quality, security, adaptive load balancing, and integration of neural and reconfigurable FPGA-based computing.

The main provisions and results of the dissertation were validated at international scientific and technical conferences and discussed at departmental scientific seminars. Based on the research results, scientific publications were produced, fully reflecting the dissertation content, including articles in specialized Ukrainian journals and conference proceedings.

The dissertation is structured in accordance with national standards and consists of an introduction, four chapters, general conclusions, a list of references, and appendices.

Keywords: distributed systems, Internet of Things, edge computing, local area networks, load balancing, security, systems-on-chip, FPGA, FPGA reconfiguration, intelligent data processing, sensor data quality, sensor data filtering, adaptive control thresholds, neural networks, monitoring systems.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Haidai A., Klymenko I. “Automatic network reconfiguration method with dynamic ip address management.” Information, Computing and Intelligent systems 6, 217–229, 2025 DOI: <https://doi.org/10.20535/2786-8729.6.2025.334139> (Фахове наукове видання України категорії Б) *Здобувач провів комплексний аналіз сучасних викликів інформаційної безпеки та спроможностей штучного інтелекту в протидії кіберзагрозам, розробив та практично реалізував метод автоматичної реконфігурації мережі шляхом інтеграції системи моніторингу Zabbix із мережесевим шлюзом pfSense, налаштував механізми оперативного реагування, включаючи автоматичну зміну IP-адрес, ізоляцію підозрілих вузлів та адаптацію правил фаєрволу, що дозволило автоматизувати управління інцидентами безпеки. Клименко І. здійснила загальне керівництво, обґрунтувала необхідність переходу до динамічних систем захисту, розробила концепцію керуючого інформаційного середовища та визначила ключові політики безпеки для детектування аномалій у поведінці мережесевих вузлів.*
2. Haidai A., Klymenko I. “Methodology of adaptive data processing in IoT monitoring systems with multilevel sensor data filtering and self-tuning.” Information, Computing and Intelligent systems 7, 110-126, 2025 DOI: <https://doi.org/10.20535/2786-8729.7.2025.341409> (Фахове наукове видання України категорії Б) *Здобувач розробив алгоритм адаптивного коригування порогових значень на основі середнього значення та стандартного відхилення сигналу. Реалізував інтеграцію адаптивних тригерів у систему Zabbix та провів серію експериментів, які підтвердили підвищення точності вимірювань (зокрема, для датчиків DHT11) та суттєве зменшення кількості хибних спрацювань тригерів. Клименко І. здійснила загальне керівництво, обґрунтувала та розробила трирівневу методику фільтрації потокових даних, що комбінує медіанне, ковзне середнє та експоненційне*

згладжування, забезпечила баланс між придушенням шумів та чутливістю системи, мінімізувавши затримки обробки для роботи в режимі реального часу.

3. Гайдай А., Клименко І. “Високопродуктивна апаратно-програмна система для граничної обробки сенсорних потоків у системах IoT на платформі FPGA-SOC” Технічні науки та технології 4(42), 195-207, 2025 DOI: [https://doi.org/10.25140/2411-5363-2025-4\(42\)-195-207](https://doi.org/10.25140/2411-5363-2025-4(42)-195-207) (Фахове наукове видання України категорії Б) Здобувач розробив концепцію та архітектуру трирівневого фільтраційного модуля на базі FPGA. Реалізував обчислювальний конвеєр, що поєднує медіанний фільтр, ковзне середнє та експоненційне згладжування. Провів комплексні експериментальні дослідження, що підтвердили зменшення затримок на 26% порівняно з програмними аналогами, а також довів ефективність системи в усуненні імпульсних шумів та стабілізації тренду сенсорних сигналів. Клименко І. Розробила програмно-апаратний механізм взаємодії між процесором ARM та логікою FPGA через інтерфейс Avalon-MM. Обґрунтувала використання фіксованого формату обчислень для мінімізації витрат логічних ресурсів та забезпечення паралельної обробки даних у режимі реального часу.
4. Haidai A., Klymenko I. “Method of load balancing in distributed three-layer IoT architecture.” Information, Computing and Intelligent systems 4, 60-68, 2024 DOI: <https://doi.org/10.20535/2786-8729.4.2024.311595> (Фахове наукове видання України категорії Б) Здобувач розробив математичну модель управління даними та граничними обчислювальними вузлами. Обґрунтував метод динамічного розподілу навантаження залежно від типу даних та поточних показників продуктивності системи, що дозволило оптимізувати використання системних ресурсів і дешевших каналів зв'язку. Клименко І. спроектувала багаторівневу структуру системи та створила її діючий прототип. Провела серію експериментів для перевірки стійкості системи до відмов і мережеских збоїв, підтвердивши ефективність механізмів тимчасового зберігання та кешування результатів на проміжних вузлах.

5. Каплунов А., Гайдай А., Гер В., Нікольський С. “Засоби моніторингу мережі в іот інфраструктурі з гібридною архітектурою.” Телекомунікаційні та інформаційні технології № 2, с. 22-32, 2023 DOI: <https://doi.org/10.31673/2412-4338.2022.022332> (Фахове наукове видання України категорії Б) *Здобувач розробив архітектуру системи моніторингу на базі технологій граничних обчислень (Edge Computing). Запропонував концепцію гібридної IoT-інфраструктури, що дозволяє поєднувати локальний моніторинг із хмарними сервісами для застосунків, чутливих до затримок часу. Гер В. розробив методика інтеграції сучасних інструментів моніторингу в мережеву інфраструктуру. Нікольський С: розробив спосіб аналізу продуктивності комп'ютерної мережі на основі зваженого показника. Впровадив використання вагових коефіцієнтів для параметрів мережі, що дозволило підвищити точність оцінки загального стану інфраструктури. Каплунов А. реалізував механізми запобігання критичним станам мережі через балансування навантаження. Довів ефективність запропонованих засобів для моніторингу потоків даних та планування ресурсів у складних IoT-системах.*
6. Клименко І., Гайдай А., Нікольський С., Ткаченко В. “Архітектурна концепція моніторингової системи на основі нейронного модуля IoT аналітики даних.” Адаптивні системи автоматичного управління Том 2 № 41, с. 111-123, 2022 DOI: <https://doi.org/10.20535/1560-8956.41.2022.271355> *Здобувач розробив прототип системи, інтеграцію нейронної мережі для аналітики даних IoT. Забезпечив поєднання доступної переносної електроніки з недорогим апаратним забезпеченням. Нікольський С. розробив теоретичну концепцію системи на базі парадигми периферійних обчислень. Обґрунтував перенесення критичних за часом обчислень із хмарних сервісів на рівень локальної мережі користувача для підвищення швидкості прийняття рішень. Клименко І. провела дослідження швидкості відгуку системи. Виявила та проаналізував виклики, пов'язані з безпекою даних у розподілених мережах та обмеженістю ресурсів пристроїв. Ткаченко В. Організувала тестування*

прототипу в контексті «розумного будинку» та медичної діагностики. Оцінила вплив системи на енергоефективність та масштабованість мережі, а також сформулювала напрямки для подальших досліджень.

7. Гайдай А., Клименко І. “Прогресивна архітектура моніторингу сенсорних даних на граничному рівні у розподілених системах IoT“ IX International Scientific and Practical Conference «SoftTech-2025» Kyiv, November 26-28, 2025 Здобувач презентував результати алгоритму адаптивного коригування порогових значень на основі середнього значення та стандартного відхилення сигналу. Клименко І. здійснила загальне керівництво, обґрунтувала та розробила трирівневу методику фільтрації поточкових даних, що комбінує медіанне, ковзне середнє та експоненційне згладжування, забезпечила баланс між придушенням шумів та чутливістю системи, мінімізувавши затримки обробки для роботи в режимі реального часу.
8. Haidai A., Klymenko I. “The method of the functional parameters estimation of the sleep monitoring system based on neural network“ The International Conference on Security, Fault Tolerance, Intelligence, Kyiv, 2022 Здобувач презентував прототип системи, інтеграцію нейронної мережі для аналітики даних IoT. Забезпечив поєднання доступної переносної електроніки з недорогим апаратним забезпеченням. Klymenko I. провела дослідження швидкості відгуку системи. Виявила та проаналізував виклики, пов'язані з безпекою даних у розподілених мережах та обмеженістю ресурсів пристроїв.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	19
ВСТУП.....	20
РОЗДІЛ 1 МЕТОДИ ТА ЗАСОБИ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ В МЕРЕЖАХ ІНТЕРНЕТУ РЕЧЕЙ	29
1.1. Архітектура систем Інтернету речей та проблеми ефективності обробки даних....	29
1.2. Особливості комплексного моніторингу інфраструктури систем IoT	32
1.3. Методи балансування навантаження в розподілених системах IoT	35
1.4. Методи інтелектуальної обробки даних з використанням нейромережевих обчислювачів в системах IoT	38
1.5. Підходи до організації обробки даних та інформаційної безпеки в розподілених системах IoT.....	41
1.6. Методи забезпечення достовірності, стабільності та якості сенсорних та технологічних даних	43
1.7. Огляд підходів до використання арифметичних прискорювачів на ПЛІС на граничному рівні інфраструктури IoT	47
1.8. Узагальнення проблематики, аналіз обмежень існуючих підходів та постановка задач дисертаційного дослідження	50
ВИСНОВКИ ДО РОЗДІЛУ 1	54
РОЗДІЛ 2 МЕТОД АДАПТИВНОЇ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ	56
2.1. Обґрунтування системного підходу до адаптивної інтелектуальної обробки даних	56
2.1.1. Архітектура та принципи функціонування адаптивно керованої інфраструктури IoT	56
2.1.2. Комплексний метод адаптивної інтелектуальної обробки даних у розподілених системах IoT	63

2.2. Удосконалений спосіб адаптивного налаштування порогів контролю у моніторингових IoT системах	65
2.3. Формування критеріїв ефективності використання ресурсів обчислювальних вузлів на граничному рівні системи IoT	71
2.3.1. Оцінка обчислювальної ефективності обробки сенсорних даних на граничному рівні IoT.....	71
2.3.2. Комплексний критерій ефективності обчислювальних ресурсів.....	74
2.3.3. Інтегральний показник завантаженості вузла для прийняття рішення щодо реконфігурації	76
2.4. Метод автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT.....	80
2.5. Спосіб динамічної реконфігурації граничних обчислювальних вузлів	85
2.6. Спосіб інтелектуальної нейромережевої обробки сенсорних даних на ПЛІС	87
2.6.1. Концепція нейромережевого обчислювача та обґрунтування формування первинного простору ознак	87
2.6.2. Опис способу інтелектуальної нейромережевої обробки сенсорних даних ..	89
2.6.3. Структура нейромережевого обчислювача.....	93
ВИСНОВОК ДО РОЗДІЛУ 2	97
РОЗДІЛ 3 РЕАЛІЗАЦІЯ АДАПТИВНОЇ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ НА ГРАНИЧНОМУ РІВНІ ІНФРАСТРУКТУРИ IoT	100
3.1. Методика реалізації адаптивно керованої інфраструктури IoT з використанням моніторингової системи Zabbix	100
3.2. Організація адаптивної обробки даних на граничному рівні інфраструктури IoT	103
3.2.1 Збір даних з датчиків	103
3.2.2 Збір даних з операційної системи вузлів інфраструктури IoT	104
3.3. Апаратний багаторівневий конвеєр на ПЛІС для фільтрації даних	107

3.4. Розроблення підсистеми балансування навантаження на граничному рівні.	111
3.5. Розроблення методу автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT.....	114
3.5.1 Призначення та роль модуля автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT.....	114
3.5.2 Логічні компоненти архітектури модуля.....	115
3.5.3 Механізм списків дозволених IP адрес: модель та правила використання ...	117
3.5.4 Механізм білого списку ПЗ: модель та правила використання.....	118
3.5.5 Типові сценарії перебудови мережі.....	118
3.5.6 Взаємодія з Zabbix і pfSense: логіка керування.....	120
3.5.7 Механізми захисту від помилок і «флапінгу» правил.....	120
3.6. Модуль інтелектуальної нейромережевої обробки для прийняття рішень на основі оброблених сенсорних сигналів	121
ВИСНОВОК ДО РОЗДІЛУ 3	126
РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА МЕТОДУ ТА ЗАСОБІВ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ.....	130
4.1. Організація та методика проведення експериментів.....	130
4.2. Дослідження способу адаптивного налаштування порогів контролю у моніторингових IoT системах	132
4.2.1. Загальна структура експериментів	132
4.2.2. Перевірка ефективності очищення сенсорних даних.....	135
4.2.3. Оцінка ефективності використання обчислювальних ресурсів.....	148
4.3. Експериментальна перевірка механізму балансування навантаження	151
4.4. Перевірка методу автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT.....	156
4.5. Експериментальне дослідження апаратної реалізації нейромережевого обчислювача на ПЛІС	165

ВИСНОВОК ДО РОЗДІЛУ 4	172
ВИСНОВКИ.....	176
ЛІТЕРАТУРА.....	180
ДОДАТОК А.....	196
ДОДАТОК Б	210

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ARM (Advanced RISC Machine) – архітектура процесорів зі зменшеним набором команд, широко використовується у вбудованих та SoC-системах.

API (Application Programming Interface) – програмний інтерфейс взаємодії між компонентами програмного забезпечення або між системами.

CPU (Central Processing Unit) – центральний процесор, універсальний обчислювальний блок системи.

DSP (Digital Signal Processing) – цифрова обробка сигналів, сукупність методів аналізу та перетворення цифрових сигналів.

DRAM (Dynamic Random Access Memory) – це тип енергозалежної напівпровідникової оперативної пам'яті, що використовується в комп'ютерах для тимчасового зберігання даних.

HDL (Hardware Description Language) – мова опису апаратури (VHDL, Verilog), що використовується для програмування ПЛІС.

HPS (Hard Processor System) – апаратний процесорний підсистемний блок у ПЛІС-SoC.

IoT (Internet of Things) – Інтернет речей, мережа фізичних пристроїв, оснащених сенсорами, виконавчими механізмами та засобами зв'язку.

Latency – затримка обробки або передавання даних у системі.

Linux – багатозадачна операційна система з відкритим кодом.

MCU (Microcontroller Unit) – мікроконтролер, компактний обчислювальний пристрій для керування периферією.

PID (Process Identifier) – унікальний ідентифікатор процесу в ОС.

RTOS (Real-Time Operating System) – операційна система реального часу з детермінованими затримками.

SoC (System on Chip) – система на кристалі, що поєднує процесор, пам'ять та периферію на одному чипі.

Zabbix – система моніторингу стану серверів, мереж і сервісів із підтримкою тригерів та автоматизації.

ВСТУП

Стрімкий розвиток інформаційних технологій і цифровізація більшості сфер діяльності сприяли широкому впровадженню розподілених систем збору та аналізу даних із застосуванням Інтернету речей (IoT). Такі системи активно застосовуються у промислових, високотехнологічних та критичних системах, включаючи енергетику, медицину[1], транспорт, екологічний моніторинг та «розумні» міста, де необхідне безперервне отримання великої кількості сенсорних вимірювань у режимі реального часу [2, 3]. Кількість підключених пристроїв і сенсорів зростає експоненційно, що призводить до збільшення обсягів даних, складності їх обробки та підвищених вимог до надійності, швидкодії й масштабованості інфраструктури IoT [2, 3].

Передавання необроблених даних безпосередньо до центральних серверів або хмарних платформ підвищує навантаження на мережу, збільшує затримки та ускладнює масштабування системи [3, 4]. Тому, класичні централізовані підходи до обробки даних у системах IoT не відповідають вимогам сучасних розподілених систем IoT, для яких критичними є обробка надвеликих потоків даних в реальному часі, ефективне використання ресурсів і масштабованість [5].

Актуальною тенденцією розвитку систем IoT є парадигма граничних обчислень [6, 7]. Застосування граничних обчислень дозволяє виконувати очищення, агрегацію та попередню обробку даних на периферійних вузлах, що зменшує обсяг передаваного трафіку, знижує затримки та підвищує стійкість системи в умовах порушення або нестабільності мережевого з'єднання з хмарною інфраструктурою. Водночас такий підхід істотно ускладнює архітектуру системи IoT та висуває підвищені вимоги до продуктивності, енергоефективності й надійності граничних вузлів обробки даних, які мають обмежені апаратні ресурси [5]. Для реалізації граничних вузлів обробки даних в сучасних архітектурах IoT використовують мікроконтролери, промислові контролери, одноплатні мінікомп'ютери, вбудовані системи реального часу та інші спеціалізовані обчислювальні платформи [8, 9].

Характерною особливістю сенсорних даних є зашумленість, нерівномірність, імпульсні викиди, короточасні флуктуації, довготривалі дрейфи, а також сезонні, добові, експлуатаційні та техногенні коливання. Ці фактори суттєво знижують достовірність аналітики та ефективність прийняття рішень у моніторингових і керуючих системах [10, 11]. Традиційні методи контролю з фіксованими порогоми не здатні адекватно реагувати на динамічні зміни стану інфраструктури, що призводить до великої кількості хибних керуючих реакцій або пропуску аномалій та критичних ситуацій. У сучасних системах IoT із динамічними потоками даних актуальною проблемою є адаптивна обробка сенсорних даних, коли пороги контролю налаштовуються автоматично відповідно до динамічних змін стану середовища. [11, 12]. Інтеграція таких підходів у розподілені системи IoT підвищує достовірність та стабільність сенсорних вимірювань для подальшої ефективної обробки, збільшує інформативність даних, підвищує стійкість інфраструктури та забезпечує оперативне прийняття керуючих рішень.

Одним із актуальних напрямів підвищення ефективності обробки потоків даних у розподілених систем IoT є розвиток апаратних рішень. Зокрема такі рішення базуються на використанні програмованих логічних інтегральних схем (ПЛІС) [9, 13] та гетерогенних систем-на кристалі SoC/ПЛІС (SoC, System on a Chip) з програмно-конфігурованою областю на ПЛІС та вбудованими процесорами високої продуктивності (наприклад, на базі архітектури ARM Cortex-A, з підтримкою універсальних операційних систем, зокрема Linux). Функціональні розширювачі на ПЛІС забезпечують масивний паралелізм, конвеєрну обробку потоків і детерміновані затримки, що підвищує ефективність обробки великої кількості сенсорних даних у режимі реального часу [14, 15]. ПЛІС реалізує обчислювально інтенсивні операції та спеціалізовані задачеорієнтовані обчислення, тоді як вбудована операційна система забезпечує керування, мережеву взаємодію та адаптацію параметрів системи.

Для інтелектуальної обробки даних актуальними є підходи з використанням нейромережових модулів на ПЛІС, які, поряд з апаратним прискоренням, забезпечують швидку обробку у режимі реального часу [16].

Але апаратні прискорення, попри високий потенціал продуктивності, породжують проблеми оптимального розподілу функцій між апаратною та програмною складовими, а також динамічної реконфігурації для ефективного використання ресурсів граничних вузлів [17]. Це зумовлює необхідність ефективного керування ресурсами граничного вузла з максимальним використанням потенціалу гетерогенних платформ без погіршення продуктивності та енергоефективності.

З огляду на вищесказане, актуальність дослідження зумовлена активним розвитком промислових високотехнологічних та критичних систем, де широко застосовуються системи IoT. У таких системах функціонують тисячі сенсорів і вимірювальних пристроїв, які безперервно генерують значні обсяги даних, формуючи надвеликі потоки інформації, що потребують ефективної адаптивної обробки та інтелектуальної інтеграції ресурсів.

Підвищення ефективності обробки потоків даних та функціонування інфраструктури на граничному рівні розподілених систем IoT залишається складною науково-практичною задачею. Традиційні централізовані підходи, локальні методи вирішення окремих задач обробки даних та керування інфраструктурою, фіксовані параметри контролю та прийняття рішень не забезпечують належної достовірності та оперативності обробки, що обмежує можливості ефективного управління потоками даних.

У зв'язку з цим доцільним є розроблення системного підходу до адаптивної інтелектуальної обробки даних на граничному рівні розподілених систем IoT, що ґрунтується на формуванні адаптивно-інтелектуально керованої гібридної інфраструктури IoT, у межах якої процеси обробки потоків даних, моніторингу даних та стану ресурсів, інтелектуальної аналітики та забезпечення ефективності інфраструктури інтегровані в єдиний керуючий контур. Такий підхід дозволить підвищити ефективність обробки потоків даних та функціонування інфраструктури на граничному рівні розподілених систем IoT.

Зв'язок роботи з науковими програмами, планами, темами.

Дисертаційна робота входить в план наукової роботи кафедри обчислювальної техніки КПІ ім. Ігоря Сікорського і виконана в рамках наступних пошукових досліджень (ініціативних тематик): «Високопродуктивні комп'ютерні системи та мережі: теорія, методи і засоби апаратної та програмної реалізації» (факультет інформатики та обчислювальної техніки – керівник: к.т.н., доц. Волокита А.М.), Державний реєстраційний номер: Д/р № 0121U108261, дата реєстрації: 01.04.2021; “Нелінійні та багатокритеріальні математичні моделі для технологій Data Science та Embedded Systems” (факультет інформатики та обчислювальної техніки – керівник: д.т.н., проф. Писарчук О.О.) Державний реєстраційний номер № 0124U003323, дата реєстрації: 12.06.2024.

Об'єктом дослідження є процеси обробки потоків даних у розподілених системах Інтернету речей (IoT), що функціонують в режимі реального часу та за умов надходження великої кількості різнорідних сенсорних вимірювань, багатопоточності обробки та масштабування інфраструктури.

Предметом дослідження є метод та засоби адаптивної інтелектуальної обробки даних на граничному рівні розподілених систем IoT, які дозволяють підвищити ефективність обробки потоків даних та функціонування інфраструктури системи IoT.

Метою роботи Підвищення ефективності обробки потоків даних та функціонування інфраструктури на граничному рівні розподілених систем IoT, що функціонують в режимі реального часу та за умов надходження великої кількості різнорідних сенсорних вимірювань, багатопоточності обробки та масштабування інфраструктури шляхом забезпечення достовірності та стабільності сенсорних вимірювань; динамічного та адаптивного керування потоками даних та забезпечення оперативного прийняття керуючих рішень; підвищення безпеки функціонування інфраструктури та ефективності використання обчислювальних ресурсів.

Для досягнення поставленої мети у роботі розв'язуються такі завдання:

1. Обґрунтувати системний підхід до адаптивної інтелектуальної обробки даних на граничному рівні розподілених систем IoT, що передбачає побудову адаптивно-інтелектуально керованої інфраструктури IoT, у межах якої процеси обробки потоків даних, моніторингу стану ресурсів та даних, інтелектуальної аналітики та забезпечення ефективності інфраструктури інтегровані в єдиний керуючий контур.

2. Формалізувати критерії ефективності обробки даних та функціонування інфраструктури, зокрема за показниками:

- продуктивності обробки потоків даних у реальному часі;
- затримок обробки та передавання даних;
- енергоспоживання граничних обчислювальних вузлів;
- рівня балансування обчислювального навантаження;
- надійності та безпеки функціонування моніторингової інфраструктури.

3. Розробити метод та засоби адаптивної інтелектуальної обробки даних, що забезпечують ефективне керування потоками на основі показників сенсорів, стану інфраструктури та ресурсів з урахуванням сформульованих критеріїв ефективності та вимог щодо достовірності та стабільності вимірювань, оперативного прийняття рішень, безпеки інфраструктури та ефективного використання ресурсів.

4. Розробити методику реалізації методу адаптивної інтелектуальної обробки даних на основі використання моніторингової системи Zabbix, яка може бути застосована для організації надійного моніторингу середовища та стану апаратних ресурсів у промислових системах IoT.

5. Розробити модуль багаторівневої фільтрації сенсорних даних та нейромережевий модуль прийняття рішень на гетерогенній платформі SoC/ПЛІС, а також механізми ефективної взаємодії між вбудованим процесором ARM Cortex A та програмно-конфігурованою логікою ПЛІС для ефективного перерозподілу задач і динамічної реконфігурації ПЛІС.

6. Виконати експериментальні дослідження та оцінити ефективність запропонованих рішень в умовах адаптивної інтелектуальної обробки даних на граничному рівні розподіленої інфраструктури IoT.

Наукова новизна

1. Запропоновано новий комплексний метод адаптивної інтелектуальної обробки даних у розподілених системах IoT, який відрізняється від відомих тим, що інтегрує сукупність взаємопов'язаних часткових методів і засобів в єдиний керуючий контур, який координує обробку даних, обчислювальні ресурси та реконфігурацію інфраструктури в реальному часі і забезпечує оперативне формування керуючих рішень. Це забезпечує універсальність застосування у різних сценаріях функціонування розподілених IoT-систем, адаптивне керування потоками даних та обчислювальними вузлами, динамічна реконфігурація інфраструктури відповідно до змін навантаження, підвищення масштабованості. У сукупності це підвищує ефективність обробки даних та функціонування інфраструктури.

2. Удосконалено спосіб адаптивного налаштування порогів контролю у моніторингових системах, який на відміну від відомих підходів із фіксованими значеннями, поєднує механізми очищення сигналів, їх стабілізації та виділення тренду з адаптивним самоналаштуванням порогів контролю. Це забезпечує їх динамічну адаптацію до змін стану середовища та режимів функціонування системи, що в цілому дозволяє зменшити кількість хибних спрацювань, підвищити чутливість до аномалій, якість сенсорних вимірювань та ефективність обробки даних у реальному часі.

3. Запропоновано спосіб динамічної реконфігурації обчислювальних вузлів на гетерогенних платформах SoC/ПЛІС, який на відміну від відомих, забезпечує інтеграцію реконфігуровної логіки з вбудованим процесорним ядром ARM та узгоджене виконання задач між апаратною і програмною складовими системи, а також можливість динамічної реконфігурації ПЛІС, залежно від обсягу обчислювального навантаження та класу задач. Це забезпечує підвищення швидкодії обробки даних, ефективності використання апаратних ресурсів і динамічний перерозподіл навантаження між обчислювальними вузлами, що в сукупності підвищує ефективність реалізації адаптивних алгоритмів обробки сенсорних потоків у режимі реального часу.

4. Отримав подальшого розвитку метод автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT, який, на відміну від відомих, забезпечує інтеграцію трьох взаємопов'язаних компонентів: безперервного моніторингу стану системи та мережевої взаємодії, виконання формалізованих алгоритмів виявлення аномальних подій та автоматизованого застосування керуючих впливів. Запропонований метод за рахунок локалізації критичних обчислень, динамічної реконфігурації мережевої інфраструктури та зниження залежності від зовнішніх мережевих з'єднань забезпечує обмеження впливу відмов, перевантажень і аномальної активності, що в сукупності підвищує стійкість функціонування та рівень кібербезпеки мережевої інфраструктури граничного рівня систем IoT.

5. Запропоновано спосіб інтелектуальної нейромережевої обробки сенсорних даних на ПЛІС, який відрізняється від відомих поєднанням апаратної стабілізації даних із нейромережевим прийняттям рішень у межах єдиного обчислювального контуру. Це забезпечує узгодженість статистичних характеристик сигналів та підвищує стійкість нейронної моделі до шумів, та дозволяє підвищити ефективність обробки даних завдяки апаратній реалізації нейромережі для прийняття рішень у реальному часі на периферії IoT.

Практичне значення

Практичне значення отриманих результатів полягає у створенні програмно-апаратної платформи адаптивної обробки сенсорних даних, що включає три взаємопов'язані компоненти: апаратний багаторівневий конвеєр на ПЛІС для фільтрації даних та отримання трендів, необхідних для реалізації адаптивних порогів; інтелектуальна нейромережева обробка для прийняття рішень на основі оброблених сенсорних сигналів; автоматизоване адаптивне керування інфраструктурою IoT, що забезпечує динамічне перерозподілення обчислювальних ресурсів і оптимізацію потоків обробки даних.

Розроблено апаратний конвеєрний прискорювач фільтрації та аналізу сенсорних даних та нейромережевий модуль прийняття рішень для реалізації на гетерогенних платформах SoC/ПЛІС, яка може бути використана для моніторингу

стану навколишнього середовища, обробки потоків сенсорних даних у реальному часі та підтримки прийняття рішень щодо різних сценаріїв обробки даних на граничних вузлах розподіленої системи IoT.

Запропонований метод адаптивної інтелектуальної обробки даних реалізовано у вигляді методики на основі використання моніторингової системи Zabbix, яка може бути застосована для організації надійного моніторингу середовища та стану апаратних ресурсів у промислових системах IoT, зокрема на великих промислових підприємствах, технологічних виробництвах, у системах промислового Інтернету речей та розумних містах

Запропоновані метод та засоби можуть бути використані для підвищення якості моніторингу сенсорних і технологічних даних та апаратних ресурсів, адаптивного балансування навантаження та інтеграції нейронних і реконфігурованих обчислень на ПЛІС. Крім того, запропонована методика може застосовуватися у навчальному процесі для підготовки фахівців у галузі промислової інженерії, IoT та embedded-систем, дозволяючи студентам набувати практичних навичок у реалізації адаптивних інтелектуальних систем.

Апробація результатів дисертації. Основні результати роботи опубліковано та обговорено на міжнародних та всеукраїнських наукових конференціях, зокрема на: The International Conference on Security, Fault Tolerance, Intelligence (м. Київ, Україна, 2022); IX International Scientific and Practical Conference «SoftTech-2025» (м. Київ, Україна, листопад 2025).

Публікації. Результати проведених наукових досліджень відображені у 8 наукових працях, 5 статті у фахових наукових виданнях України, 1 стаття у наукових виданнях України, а також опубліковано 2 тез доповідей та матеріалів конференцій.

Особистий внесок здобувача.

Дисертаційна робота є самостійним дослідженням автора. Усі наукові результати дисертаційного дослідження отримані здобувачем особисто. В роботах, які були опубліковані в співавторстві (Додаток Б): [1] – провів комплексний аналіз сучасних викликів інформаційної безпеки та використання спроможностей

штучного інтелекту в протидії загрозам, розробив та реалізував частковий метод автоматичного прийняття рішень у безпекових сценаріях; [2] – розробив спосіб адаптивного налаштування порогів контролю у моніторингових системах, провів серію експериментів, які підтвердили підвищення точності вимірювань та суттєве зменшення кількості хибних спрацювань тригерів; [3] – розробив концепцію та архітектуру трирівневого фільтраційного модуля на базі FPGA. Реалізував обчислювальний конвеєр, що поєднує медіанний фільтр, ковзне середнє та експоненційне згладжування. Провів комплексні експериментальні дослідження, що підтвердили зменшення затримок на 26% порівняно з програмними аналогами, а також довів ефективність системи в усуненні імпульсних шумів та стабілізації тренду сенсорних сигналів; [4] – розробив математичну модель управління даними та граничними обчислювальними вузлами, обґрунтував спосіб балансування навантаження; [5] – розробив архітектуру системи моніторингу на базі технологій граничних обчислень; [6] – розробив прототип системи, інтеграцію нейронної мережі для аналітики даних IoT; [7] – презентував результати алгоритму адаптивного коригування порогових значень на основі середнього значення та стандартного відхилення сигналу; [8] – презентував прототип системи, інтеграцію нейронної мережі для аналітики даних IoT.

Структура і обсяг роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел із 122 найменувань та додатків. Загальний обсяг дисертації становить 213 сторінки, з яких 195 сторінок основного тексту, 2 додатки на 18 сторінках, та містить 24 рисунка, 45 формул, 8 таблиці.

РОЗДІЛ 1

МЕТОДИ ТА ЗАСОБИ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ В МЕРЕЖАХ ІНТЕРНЕТУ РЕЧЕЙ

1.1. Архітектура систем Інтернету речей та проблеми ефективності обробки даних

Інтернет речей (IoT) являє собою розподілену систему, що об'єднує фізичні об'єкти, сенсорні пристрої, комунікаційні мережі та програмні сервіси [18, 19] з метою автоматизованого збору, передачі, аналізу та використання даних у режимі реального часу та управлінням самої системи. Ключовою особливістю є їх масовість, гетерогенність та орієнтація на безперервні потоки даних, які формуються великою кількістю джерел з обмеженими обчислювальними та енергетичними ресурсами.

Типова архітектура систем IoT характеризується багаторівневою побудовою. Класична трирівнева модель включає сенсорний рівень, мережевий або транспортний рівень та прикладний рівень [4, 20]. Сенсорний рівень відповідає за збирання первинних даних із фізичного середовища. Мережевий рівень забезпечує передачу зібраних даних через дротові або бездротові канали зв'язку до обчислювальних вузлів, а прикладний рівень виконує аналіз, візуалізацію та підтримку прийняття рішень.

Подальший розвиток технологій Інтернету речей (IoT) зумовив перехід від початкових простих архітектур до розширених багат шарових і гібридних моделей, зокрема з включенням граничних та туманних рівнів обробки даних [2, 21–23]. У науковій літературі такі підходи широко представлені у вигляді чотирирівневих та п'ятирівневих архітектур IoT [20]. У межах таких архітектур зазвичай рівні деталізуються відповідно до виконуваних функцій. Подібні підходи до структуризації архітектури IoT докладно розглядаються в наукових роботах і стандартах [5, 6, 24–27].

Водночас аналіз відомих архітектурних рішень показує, що більшість багаторівневих моделей можуть бути узагальнені шляхом агрегування

функціонально близьких компонентів у логічні прошарки. Такий підхід дає можливість перейти від деталізованих багаторівневих моделей до узагальненої логічної тришарової структури архітектури IoT, у межах якої чітко виокремлюється функціональний рівень граничних обчислень. Такий підхід дає змогу узагальнити різні архітектурні рішення IoT та представити їх у вигляді компактної логічної моделі з чітко визначеними рівнями абстракції, кожен з яких орієнтований на розв'язання специфічних задач.

У результаті будь-яка багаторівнева архітектура IoT трансформується в логічну тришарову модель [2, 20, 28, 29], яка розглядається в дисертаційній роботі, як базова. Така модель включає:

- рівень пристроїв,
- граничний рівень,
- хмарний рівень.

У межах цієї моделі чітко виокремлюються відповідні рівні обчислювальної абстракції, зокрема рівень граничних [30] та рівень хмарних обчислень [31].

Далі розглянемо призначення та основні функції кожного з рівнів базової тришарової моделі.

Рівень пристроїв є нижнім рівнем архітектури IoT та включає сенсори, вимірювальні модулі, контролери та виконавчі пристрої. Саме на цьому рівні формується первинний потік даних, який може включати вимірювання температури, вологості, освітленості, тиску, вібрацій, фізіологічних параметрів, стану обладнання тощо.

Дані, що надходять із сенсорів, мають низку характерних особливостей:

- високу частоту надходження;
- наявність шумів та випадкових флуктуацій;
- імпульсні викиди;
- обмежену точність та стабільність сенсорів;
- залежність якості даних від умов експлуатації.

Обмежені апаратні ресурси сенсорних вузлів не дозволяють виконувати складну обробку даних безпосередньо на рівні пристроїв. У більшості випадків

сенсори передають сирі або мінімально оброблені дані, що створює значне навантаження на канали зв'язку та наступні рівні системи.

Граничний рівень відіграє ключову роль у сучасних архітектурах IoT і реалізує концепцію граничних обчислень. На цьому рівні розміщуються шлюзи, вбудовані комп'ютери, мікросервери та обчислювальні вузли, зокрема на платформах SoC/ПЛІС, які забезпечують прийом даних від сенсорів та їх попередню обробку.

Основними завданнями граничного рівня є [1]:

- агрегація та стиснення даних;
- зниження затримок реакції системи.

Перенесення частини обчислень на граничний рівень дозволяє суттєво зменшити навантаження на хмарну інфраструктуру та канали зв'язку, що є критично важливим для масштабних систем IoT. Однак саме на цьому рівні виникає низка складних задач, пов'язаних з ефективною обробкою потокових даних у реальному часі при обмежених ресурсах.

Хмарний рівень архітектури IoT забезпечує довготривале зберігання даних, виконання складної аналітики, машинного навчання, візуалізацію та інтеграцію з зовнішніми сервісами. На цьому рівні використовуються бази даних часових рядів, аналітичні платформи та системи підтримки прийняття рішень.

Незважаючи на високу обчислювальну потужність хмарних ресурсів, централізована обробка даних має наступні обмеження:

- значні затримки при передачі даних;
- залежність від стабільності мережевого з'єднання;
- високі вимоги до пропускної здатності каналів;
- підвищені ризики втрати або компрометації даних.

У зв'язку з цим сучасні системи IoT дедалі частіше використовують гібридні архітектури, у яких обробка даних розподіляється між граничними та хмарними рівнями.

1.2. Особливості комплексного моніторингу інфраструктури систем IoT

У межах парадигми Інтернету речей, зокрема для промислових підприємств, технологічних цехів, систем «розумних міст» та інших організацій, які здійснюють збір і обробку даних від великої кількості віддалених пристроїв різних типів, завдання моніторингу є невід’ємною частиною функціонування систем IoT. Моніторингові системи забезпечують контроль стану віддалених серверів і пристроїв IoT шляхом збору сукупності експлуатаційних метрик, відстеження їхнього робочого статусу та оцінювання загального стану мережевої інфраструктури IoT [32, 33].

Завдання моніторингу в системах IoT набувають комплексного характеру: вони охоплюють відстеження стану компонентів самої системи на різних рівнях архітектури, а також спостереження за параметрами зовнішнього середовища, що реєструються сенсорними пристроями.

В умовах активного впровадження хмарних технологій у складі систем IoT засоби моніторингу також виконують функції інтеграції локальної інфраструктури підприємства з хмарними сервісами. Сучасні хмарні платформи забезпечують розвинені інструменти для зберігання, обробки та аналітики великих обсягів даних [34]. Зокрема, концепція хмари речей дозволяє формалізувати задачі обробки даних IoT, зменшити залежність між хмарними сервісами та локальною інфраструктурою, а також забезпечити гнучкість масштабування [35]. Це зумовлює зростання зацікавленості компаній у використанні хмарних сервісів для контролю стану систем IoT і надання абонентської підтримки кінцевим користувачам.

Водночас, попри наявність широкого спектра хмарних засобів моніторингу, результати сучасних досліджень [36, 37] свідчать, що гетерогенність нижніх рівнів інфраструктури IoT, обмежені обчислювальні ресурси, а також специфічні апаратні й програмні характеристики пристроїв IoT істотно знижують ефективність застосування уніфікованих систем моніторингу та традиційних хмарних рішень із централізованою архітектурою. Використання хмарних сервісів загального призначення часто супроводжується надмірними експлуатаційними витратами та жорсткою монолітною прив’язкою платформ до універсальних технологічних

рішень, що призводить до виникнення як технічних, так і економічних обмежень під час побудови масштабованої інфраструктури IoT.

У зв'язку з цим для більшості компаній, діяльність яких базується на застосуванні технологій IoT, інтеграція з хмарними сервісами не може бути повною. Обґрунтованим концептуальним підходом у сучасній парадигмі IoT є розміщення частини функціональних компонентів безпосередньо на периферії мережі, тобто на рівні граничних обчислень. З огляду на активний розвиток хмарних технологій найбільш перспективним вважається гібридний підхід, який поєднує можливості хмарних і граничних обчислень у межах єдиної інфраструктури [35].

Доцільність широкого використання хмарних технологій у парадигмі Інтернету речей обґрунтовано в роботі [34], де узагальнено їх основні переваги, зокрема наявність віддалених обчислювальних ресурсів, підтримку розвинених інтерфейсів API, централізовані механізми безпеки та конфіденційності даних, ефективні засоби автентифікації й шифрування, а також низький вхідний бар'єр для хостинг-провайдерів. Сукупність цих характеристик дозволяє здійснювати централізований контроль функціонування систем IoT і забезпечувати підтримку користувачів. Разом із тим централізована та уніфікована архітектура хмарних технологій обмежує можливості їх ефективного застосування в гетерогенних середовищах.

Однією з ключових задач під час розроблення та експлуатації інфраструктури IoT залишається моніторинг мережного оточення [32], актуальність якого істотно зростає в умовах використання гібридних архітектур. На системи моніторингу покладаються функції контролю стану локальних і віддалених серверів, пристроїв IoT, а також оцінювання загального стану мережевої інфраструктури.

Незважаючи на наявність широкого спектра хмарних інструментів моніторингу [36], результати сучасних досліджень свідчать, що децентралізована обробка даних і реалізація частини сервісів на локальному рівні у межах гібридних архітектур ускладнюють або роблять неможливим використання універсальних хмарних систем моніторингу [37, 38]. Крім того, існуючі підходи до моніторингу

продуктивності мережі здебільшого зосереджені на вимірюванні базових параметрів, таких як завантаження процесорів, пропускна здатність, затримки та втрата пакетів, і не забезпечують механізмів балансування навантаження [39, 40], що є критично важливими для розподілених мікросервісних інфраструктур IoT. Це обумовлює необхідність подальших досліджень, спрямованих на розроблення засобів моніторингу, аналізу та планування потоків даних у гібридній інфраструктурі з урахуванням її гетерогенності, масштабованості та вимог реального часу.

У роботах [41] показано, що централізоване агрегування метрик у великих системах IoT призводить до збільшення затримок, зниження точності оцінювання стану мережі та виникнення єдиної точки відмови. У зв'язку з цим дедалі більшого поширення набувають ієрархічні та розподілені моделі моніторингу, в яких первинна обробка даних здійснюється на рівні граничних вузлів.

Граничні обчислення дозволяють реалізувати локальний аналіз телеметричних даних без необхідності передачі всього обсягу інформації до хмари [42]. Такий підхід знижує затримки, підвищує стійкість системи до відмов мережевого з'єднання та забезпечує більш оперативну реакцію на аномальні стани. Зростання кількості сенсорних вузлів призводить до експоненційного збільшення обсягів даних, що відповідає характеристикам великих даних описаним у [3, 42–44].

Водночас децентралізований характер гібридних систем IoT ускладнює узгодження та інтерпретацію зібраних метрик. Різні типи пристроїв використовують неоднорідні протоколи зв'язку, формати даних і часові шкали, що потребує уніфікованих механізмів нормалізації та кореляції інформації [45].

У дослідженні [46] наголошується, що відсутність єдиної семантичної моделі даних істотно знижує ефективність аналітики та унеможливорює коректне прогнозування стану мережі.

Таким чином, аналіз сучасних наукових джерел свідчить, що ефективний моніторинг у гібридних інфраструктурах IoT повинен базуватися на поєднанні розподіленої обробки даних, уніфікованих моделей представлення метрик і тісної

інтеграції з механізмами управління ресурсами. Недостатня адаптивність існуючих хмарних і універсальних рішень зумовлює необхідність розроблення спеціалізованих методів і архітектур моніторингу, орієнтованих на гетерогенні, масштабовані та динамічні середовища IoT. Саме в цьому напрямі доцільно зосередити подальші дослідження, спрямовані на підвищення ефективності, надійності та керованості сучасних систем IoT.

1.3. Методи балансування навантаження в розподілених системах IoT

Подальший розвиток інфраструктури IoT призводить до збільшення інтенсивності інформаційних потоків і нерівномірного розподілу обчислювального та мережевого навантаження між компонентами системи.

За таких умов ключовим фактором забезпечення стабільної роботи середовища IoT стає ефективне балансування навантаження між рівнями тришарової архітектури, що включає рівень пристроїв IoT, рівень граничних обчислень і хмарний рівень.

Незважаючи на широке використання хмарних технологій як центрального рівня обробки даних, їх застосування в масштабних системах IoT обмежується не стільки обсягом корисних сенсорних даних як таким, скільки сумарними накладними витратами на передавання, маршрутизацію, протокольну інкапсуляцію, зберігання та аналіз великої кількості повідомлень у режимі, наближеному до реального часу.[4, 30, 47–49]

У зв'язку з цим балансування навантаження [50–54] дедалі частіше реалізується шляхом перерозподілу обчислювальних задач між хмарним і периферійними рівнями з метою зменшення навантаження на магістральні канали зв'язку та скорочення часу обробки даних. Такий підхід передбачає перенесення частини обчислень і прийняття рішень максимально близько до джерел даних, що відповідає концепції граничних обчислень.

Перехід до граничних та туманних обчислень у тришаровій архітектурі IoT зумовлює необхідність застосування нових методів балансування навантаження, здатних працювати в умовах багаторівневої, гетерогенної та динамічної мережевої

інфраструктури. Наявність проміжного рівня між пристроями IoT та центрами обробки даних ускладнює процеси керування ресурсами й потребує врахування поточного стану обчислювальних вузлів, характеристик мережеских з'єднань і мобільності частини пристроїв. У таких умовах балансування навантаження розглядається не лише як механізм підвищення продуктивності, а й як засіб підвищення надійності та живучості систем IoT шляхом зменшення кількості вузьких місць і потенційних точок відмови.

Таким чином, формується прикладна задача розроблення методів балансування навантаження, орієнтованих на ефективний розподіл обчислювальних і мережеских ресурсів між вузлами тришарової архітектури IoT з урахуванням обмежених ресурсів, динаміки навантаження та вимог до затримок.

У роботі [55] розглянуто методи балансування навантаження та зменшення енергоспоживання в системах туманних обчислень. Запропонований підхід на основі методу DRAM (Dynamic Random Access Memory) демонструє зниження затримок і втрат пакетів, однак у базовій реалізації не повною мірою враховує енергетичні обмеження пристроїв IoT, що є критичним чинником для автономних вузлів. Удосконалений варіант методу дозволяє підвищити ефективність балансування з урахуванням зазначених обмежень.

У роботі [56] акцентовано увагу на мобільності пристроїв IoT, зокрема тих, що розміщуються на транспортних засобах із фіксованими або змінними маршрутами. Автори зазначають, що під час балансування навантаження необхідно враховувати передбачуваність переміщення пристроїв з метою зменшення частоти перевизначення обчислювальних вузлів і зниження додаткового мережевого навантаження. У роботі [57] запропоновано метод маршрутизації та балансування навантаження з використанням багатошляхового передавання даних, що дозволяє зменшити затримки та підвищити стійкість мережі у випадках відмов окремих маршрутів. Вибір вузлів для обробки здійснюється на основі інформації про поточний рівень їх завантаження, яка поширюється за допомогою спеціальних повідомлень–маяків.

Огляд сучасних підходів до балансування навантаження в хмарних і розподілених середовищах IoT наведено в роботі [58], де визначено ключові напрями подальших досліджень. До них належать керування пріоритетами даних з урахуванням часових обмежень, балансування навантаження з урахуванням трафіку та характеристик мережі, оптимізація існуючих алгоритмів, використання туманних обчислень для розподіленої аналітики великих обсягів даних, а також підвищення сумісності між хмарними, граничними й туманними технологіями.

У роботі [59] досліджено методи балансування навантаження в хмарних центрах обробки даних, що використовуються для обслуговування пристроїв IoT. Запропонований підхід забезпечує ефективніший розподіл задач між віртуальними машинами за визначеними параметрами, проте його застосування вимагає інтеграції з механізмами балансування на периферійних рівнях архітектури IoT.

Окремий клас методів балансування навантаження в IoT базується на використанні програмно-конфігурованих мереж (Software-defined Networking, SDN). У роботах [60, 61] показано, що застосування SDN дозволяє централізовано керувати маршрутизацією та навантаженням у розподіленій мережі IoT шляхом поділу її на кластери, керовані SDN-контролером. Результати моделювання підтверджують підвищення ефективності використання ресурсів і подовження часу безперервного функціонування системи. У роботі [62, 63] наведено комплексний огляд SDN-орієнтованих методів балансування навантаження, де підкреслено необхідність подальшого вдосконалення таких рішень для масштабних і гетерогенних середовищ IoT. Одним із перспективних підходів до балансування навантаження в мережах IoT є використання алгоритму мурашиної колонії (Ant Colony Optimization, ACO) у поєднанні з програмно-визначеними мережами. Автори [64] запропонували техніку на основі ACO для балансування навантаження у хмарних центрах обробки даних, довівши ефективність підходу для динамічного розподілу ресурсів. Подальший розвиток цього напрямку представлено у роботі [65], де запропоновано динамічний алгоритм балансування навантаження, що поєднує класифікацію завдань із алгоритмом мурашиної колонії в середовищі хмарних обчислень SDN, що дозволяє враховувати пріоритети та типи завдань при

прийнятті рішень про розподіл. У роботі [66] провели систематичну порівняльну оцінку технік на основі АСО для балансування мережевого навантаження, підтвердивши, що алгоритми мурашиної колонії демонструють стабільно високі результати при вирішенні задач маршрутизації та розподілу ресурсів у сучасних мережових середовищах.

У роботах [67–69] розглянуто підходи до використання програмно–визначених мереж (SDN) у середовищах IoT. Показано, що централізоване управління потоками трафіку дозволяє оперативно перенаправляти дані в обхід перевантажених або скомпрометованих вузлів. Разом із тим, централізація підвищує вимоги до надійності контролера та захисту каналів керування.

У роботах [70–71] досліджено, що поєднання потокової аналітики з алгоритмами навчання з підкріплення забезпечує ефективний вибір стратегій маршрутизації та розподілу ресурсів у динамічних мережах.

1.4. Методи інтелектуальної обробки даних з використанням нейромережових обчислювачів в системах IoT

Зростання складності та обсягів даних в системах IoT зумовило необхідність переходу від традиційних алгоритмічних методів обробки до інтелектуальних підходів, заснованих на методах машинного навчання та нейронних мереж. Нейронні модулі в інфраструктурах IoT застосовуються для аналізу поточкових даних, виявлення прихованих закономірностей, прогнозування станів системи та підтримки прийняття рішень.

Інтелектуальна обробка даних може бути реалізована на різних рівнях системи IoT. У класичному підході нейронні мережі розміщуються на хмарному рівні, де доступні значні обчислювальні ресурси та великі обсяги даних для навчання моделей [72–75]. Така архітектура забезпечує високу точність аналізу, проте супроводжується значними затримками передачі даних, підвищеним мережовим навантаженням і залежністю від стабільності з'єднання з хмарою.

Альтернативним напрямом розвитку є перенесення нейронних модулів на граничний рівень, що дозволяє виконувати інтелектуальну обробку безпосередньо

поблизу джерел даних. Гранично–орієнтовані нейронні модулі застосовуються для задач попередньої класифікації, фільтрації, виявлення аномалій і адаптації параметрів системи в режимі реального часу [76–78]. Такий підхід знижує затримки, підвищує автономність вузлів IoT і зменшує обсяг даних, що передаються до хмари.

З точки зору архітектури, нейронні модулі можуть бути інтегровані у вигляді окремих сервісів, вбудованих апаратно–програмних компонентів або частини гібридних систем, що поєднують програмну реалізацію з апаратним прискоренням. У контексті IoT особливе значення мають компактні моделі нейронних мереж, здатні працювати в умовах обмежених обчислювальних і енергетичних ресурсів.

Розвиток TinyML (Tiny Machine Learning) як підходу до розгортання моделей глибокого навчання безпосередньо на ресурсообмежених пристроях IoT привертає дедалі більшу увагу. У роботі [79–82] автори демонструють, що сучасні техніки стиснення моделей і дистиляція знань дозволяють робити прогнози нейронних мереж на мікроконтролерах із надмалим енергоспоживанням, зберігаючи при цьому прийнятну точність для прикладних задач IoT. Автори [83] аналізують реальні застосування TinyML у сферах охорони здоров'я, промислового моніторингу та розумних міст, а також окреслюють перспективні напрями зокрема федеративне навчання на граничному рівні та адаптивні моделі, здатні навчатися в умовах обмежених ресурсів.

Для інтелектуальної обробки даних IoT найчастіше застосовуються згорткові нейронні мережі [84], рекурентні нейронні мережі, а також автоенкодери та гібридні архітектури. Вони використовуються для прогнозування часових рядів, оцінювання стану сенсорів, виявлення деградації обладнання та оптимізації параметрів функціонування системи.

Важливою архітектурною особливістю є розділення процесів навчання та практичного застосування нейронних моделей. Як правило, навчання виконується на хмарному або серверному рівні з використанням великих масивів даних, тоді як обробка даних реалізується на граничних вузлах у режимі реального часу. Такий

підхід дозволяє поєднати високу точність моделей із вимогами до швидкодії та енергоефективності.

У роботі [85] запропоновано підхід до дослідження якості сну та виявлення нарколепсії з використанням нейронних мереж, які аналізують стадії сну людини. Автори підкреслюють, що традиційні методи ручної обробки даних є трудомісткими, дорогими та характеризуються ймовірністю помилок до 18 %, а також потребують тривалого перебування пацієнта в медичному закладі. Використання нейронних мереж дозволяє суттєво скоротити час первинної діагностики, зменшити вартість обстеження та водночас забезпечити високу точність результатів.

У роботі [86] розглянуто орієнтовану IoT систему виявлення апное сну, що є поширеним порушенням дихання серед людей різного віку. Запропоноване рішення базується на аналізі низки фізіологічних параметрів, зокрема насичення крові киснем, індексу маси тіла, тривалості сну, рівня глюкози та холестерину в крові, віку та інших показників. Автори здійснюють порівняльний аналіз існуючих рішень та наголошують, що застосування нейронних мереж є відносно новим підходом, який забезпечує обробку даних у режимі, близькому до реального часу. Це дає змогу оперативно інформувати медичний персонал або родичів пацієнта про критичний стан і необхідність надання екстреної допомоги. Реалізація системи базується на апаратно-програмному модулі з використанням плати Arduino UNO, сенсорної системи та мобільного застосунку.

У статті [8] описано систему моніторингу якості сну, що використовує дані з мікрофона, акселерометра та пульсоксиметра. Для аналізу зібраних даних застосовано методи машинного навчання, зокрема алгоритм керованої класифікації випадкового лісу (*Random Forest*). За результатами експериментів автори досягли точності класифікації 95,67%, що підтверджує ефективність запропонованого підходу. Зазначається, що система є відносно недорогою завдяки використанню доступних сенсорів, що робить можливим її застосування як у домашніх умовах, так і в медичних закладах. Архітектура рішення включає сенсорний модуль, плату Arduino та серверну частину для обробки даних. Таким чином, аналіз розглянутих

наукових джерел показує, що використання нейронних мереж і алгоритмів машинного навчання в системах на основі IoT дозволяє підвищити швидкість і точність обробки даних, зменшити вартість діагностики медичних проблем та апаратних засобів, а також забезпечити можливість аналізу даних у режимі реального часу.

1.5. Підходи до організації обробки даних та інформаційної безпеки в розподілених системах IoT

У сучасних умовах стрімкого розвитку цифрових технологій та зростаючої залежності суспільства від комп'ютеризованих систем питання забезпечення інформаційної безпеки набуває особливої актуальності. Постійне ускладнення інфраструктур інформаційних технологій, широке використання хмарних сервісів, віддаленого доступу та автоматизованих систем управління призводять до зростання кількості й різноманітності кіберзагроз.

Серед них особливе місце займають автоматизовані атаки з використанням зловмисного програмного забезпечення, цілеспрямовані вторгнення, атаки з елементами соціальної інженерії, фішинг, а також експлуатація вразливостей програмного забезпечення. У зв'язку з цим зростає потреба у впровадженні систем моніторингу, здатних не лише виявляти аномалії та інциденти безпеки, але й автоматично реагувати на них у режимі, близькому до реального часу.

Відомі рішення для багатоцільового моніторингу інформаційної інфраструктури реалізуються, зокрема, на базі системи Zabbix [87–90], яка забезпечує централізований збір і аналіз даних про стан операційних систем, використання ресурсів та мережеву активність. Нові можливості у сфері інформаційної безпеки відкриває інтеграція Zabbix із мережевими шлюзами та міжмережевими екранами, зокрема pfSense. Такий підхід дозволяє не лише здійснювати моніторинг, а й автоматизувати зміни мережевої конфігурації, включаючи динамічну перебудову маршрутів, блокування підозрілих вузлів і зміну IP-адрес клієнтських хостів.

Аналіз сучасних наукових досліджень у сфері кібербезпеки [91–94] свідчить про зростаючий інтерес до інноваційних підходів, спрямованих на підвищення ефективності виявлення загроз, запобігання атакам, мінімізацію їх наслідків, а також оцінку й управління ризиками. У роботі [95] розглянуто застосування методів штучного інтелекту для виявлення та реагування на кіберзагрози в хмарних середовищах. Автори демонструють високий рівень виявлення інцидентів на підготовлених наборах даних, однак зазначають, що ефективність моделей потребує перевірки на великих практичних масивах реальних даних. Крім того, швидка еволюція атак із використанням власних інструментів машинного навчання з боку злоумисників обмежує повноту наявних датасетів і знижує універсальність навчених моделей.

Проблематиці кібербезпеки в системах промислового управління, які внаслідок цифровізації та підключення до Інтернету стали потенційно вразливими, присвячена робота [96]. Автори підкреслюють, що традиційні методи виявлення загроз у таких системах часто виконуються вручну, потребують залучення експертів і мають низьку масштабованість. У відповідь на це запропоновано автоматизований фреймворк пошуку загроз на основі бази знань MITRE ATT&CK for ICS (Industrial Control Systems), що дозволяє формувати гіпотези атак і здійснювати аналіз у режимі реального часу з використанням інструментів Elastic Stack, Kibana та Metasploit.

У роботі [97] пропонують децентралізовану архітектуру безпеки для IoT на основі блокчейну, де розподілений реєстр використовується для автентифікації пристроїв і управління доступом без єдиної точки відмови, що підвищує стійкість системи до атак. Автори [98] розвивають концепцію Zero Trust для масштабних середовищ IoT, запропонувавши архітектуру Zero Trust+, яка інтегрує механізми оцінки рівня довіри до пристроїв із класичним принципом «ніколи не довіряй, завжди перевіряй», адаптуючи його до гетерогенних і динамічних мереж IoT із великою кількістю вузлів.

Окрему увагу в сучасних дослідженнях приділено застосуванню методів штучного інтелекту та машинного навчання в кібербезпеці. У роботі [99]

розглянуто підконтрольне, неконтрольоване навчання та навчання з підкріпленням як основні підходи до аналізу кіберзагроз. Показано, що алгоритми машинного навчання, зокрема нейронні мережі, дерева рішень, згорткові та рекурентні нейронні мережі, успішно застосовуються для виявлення зловмисного програмного забезпечення, моніторингу вторгнень, аналізу поведінки користувачів і протидії фінансовому шахрайству. Водночас автори зазначають низку обмежень, серед яких потреба у великих обсягах якісних даних, складність інтерпретації рішень моделей, загроза атак на самі моделі та проблема зміни шаблонів атак.

1.6. Методи забезпечення достовірності, стабільності та якості сенсорних та технологічних даних

Сучасні системи IoT характеризуються не лише високою інтенсивністю потоків даних, а й гетерогенністю джерел, що формуються великою кількістю сенсорів різних типів. Первинні дані, отримані від сенсорів і датчиків, зазвичай розглядають як «сирі дані» (raw data), які нерідко містять шум, дублювання, неповноту та помилки зчитування, унаслідок чого стають малопридатними для безпосереднього аналізу та прийняття рішень. У системах IoT поява необроблених даних супроводжується шумами в сенсорних сигналах (наприклад, температурними флуктуаціями або електричними завадами), дублюванням вимірювань, збоями під час передавання (втратою пакетів), некоректними значеннями, а також неконсистентністю між джерелами, коли різні сенсори формують несумісні або суперечливі дані [10–12].

З одного боку, надлишкові та помилкові дані ускладнюють аналітику, знижують точність рішень і спричиняють збільшення кількості хибних спрацювань у моніторингових та керуючих підсистемах. З іншого боку, передавання даних без попередньої обробки створює надмірне навантаження на мережеву інфраструктуру та хмарні обчислювальні вузли, що знижує масштабованість і швидкодію систем IoT у цілому, а також погіршує якість прогнозування, зокрема у підсистемах штучного інтелекту.

Для мінімізації впливу зазначених факторів особливої актуальності набувають методи багаторівневої фільтрації на граничному рівні інфраструктури IoT, які дозволяють комбінувати різні типи фільтрації та згладжування сигналів. Такий підхід забезпечує попередню обробку та очищення даних до їх передавання у хмарні або серверні вузли, істотно зменшуючи обсяг трафіку та навантаження на мережеві ресурси. Водночас важливим аспектом багаторівневої фільтрації є реалізація адаптивної фільтрації, за якої параметри та коефіцієнти згладжування можуть автоматично коригуватися в реальному часі відповідно до динаміки сигналів, що надходять від великої кількості гетерогенних сенсорів. Реалізація таких механізмів на програмно-апаратному рівні дає змогу формувати високопродуктивні модулі обробки потоків даних у реальному часі, зокрема на граничному рівні інфраструктури.

У сучасному цифровому світі дані, що генеруються пристроями IoT, перетворилися на ключовий інформаційний і стратегічний ресурс для прийняття рішень у багатьох сферах від промисловості до інфраструктури «розумних» міст. При цьому якість даних виступає визначальним чинником ефективності систем IoT, однак процес збирання та передавання інформації часто супроводжується шумами, перешкодами, пропусками та спотвореннями, що знижує достовірність вимірювань. У науковій літературі запропоновано низку підходів до підвищення якості сенсорних даних, зокрема шляхом усунення перешкод і згладжування сигналів за допомогою комплексної багаторівневої фільтрації.

Методи багаторівневої фільтрації поєднують різні алгоритмічні підходи з метою забезпечення стабільності, точності та надійності даних. Зокрема, медіанний фільтр [100, 101] ефективно усуває імпульсні завади, ковзне середнє [102] зменшує випадкові коливання, а експоненційне згладжування забезпечує адаптивність до темпу надходження даних і динамічних змін характеру сигналів. Комплексне використання цих методів створює основу для підвищення точності аналітичних рішень на граничному і хмарному рівнях, зменшення навантаження на мережеву інфраструктуру та енергоспоживання, а також підвищення надійності систем IoT. У цьому контексті сучасні дослідження спрямовані на задачі часової адаптивності,

стійкості до шумів і оптимізації обчислювальних ресурсів, що безпосередньо впливає на зниження кількості хибних спрацювань у моніторингових сценаріях.

У статті [103] розглянуто застосування медіанного фільтра для обробки даних температури від сенсорів DHT11 та DHT22 у системах моніторингу. Показано, що використання цього методу зменшує коливання та сплески у вимірюваннях, спричинені шумами сенсорів, і підвищує стабільність показників. Результати підтверджують ефективність медіанного фільтра у мінімізації імпульсних перешкод без істотного спотворення ключових характеристик сигналу; для DHT11, як менш точного сенсора з більшою дискретністю, практична значущість такого фільтрування є особливо помітною.

Автори [104] оцінюють ефективність фільтра Калмана для обробки даних ґрунтових сенсорів у реальному часі на ресурсообмежених пристроях IoT і підтверджують його придатність для агрокліматичного моніторингу. А у роботі [105] пропонують оптимізацію фільтра Калмана за допомогою генетичного алгоритму, що дозволяє автоматично налаштовувати параметри фільтрації та підвищує точність обробки сигналів сенсорів IoT.

У статті [106] досліджено підвищення ефективності передавання даних у бездротовій сенсорній мережі моніторингу лісових пожеж на базі LoRaWAN. Запропоновано багаторівневу схему попередньої фільтрації на етапі збору, де використовуються методи згладжування, зокрема ковзне середнє та експоненційне згладжування, з метою зменшення шумів і скорочення обсягів переданої інформації. Показано, що такий підхід дозволяє знизити енергоспоживання вузлів за рахунок меншої кількості передач, зменшити навантаження на канал зв'язку та водночас зберегти інформативність вимірювань, що є важливим для реальних систем екологічного моніторингу.

У роботі [107] розглянуто проблему підвищення точності й надійності позиціонування у системах на базі Ultra-Wideband (UWB), які активно застосовуються в рішеннях IoT для навігації та моніторингу. Автори зазначають, що UWB-вимірювання схильні до похибок через багатопроменевість, відбиття та систематичні відхилення у визначенні відстані, що призводить до неточностей

координат. Для усунення цих недоліків запропоновано дворівневий підхід: калібрування систематичних відхилень (ranging offset calibration) та подальше згладжування залишкових випадкових шумів методом ковзного середнього. Експериментальні результати демонструють, що інтеграція цих методів суттєво зменшує стрибки в траєкторіях та забезпечує більш стабільне позиціонування; хоча ковзне середнє може обмежено впливати на абсолютну точність, воно помітно підвищує стабільність даних і якість відображення руху в практичних застосуваннях.

У статті [108] досліджено вплив параметрів ковзного середнього на якість даних ультразвукових сенсорів. Показано, що збільшення розміру вікна усереднення посилює згладжування шумів і підвищує стабільність сигналу, однак надмірне збільшення вікна призводить до втрати чутливості до реальних змін у середовищі. Отже, оптимальний розмір вибірки має забезпечувати компроміс між точністю та швидкістю реакції сенсорної системи, що є принципово важливим для додатків IoT моніторингу відстаней та навігаційних сценаріїв.

У роботі [109] запропоновано метод згладжування сигналів у режимі реального часу, що поєднує властивості медіанного фільтра та ковзного середнього. Алгоритм Smart–Median працює за принципом: якщо поточне значення суттєво відрізняється від медіани сусідніх значень, воно замінюється середнім, інакше використовується медіана. Експерименти показали ефективне зменшення впливу шумів і викидів зі збереженням плавності сигналу. Автори підкреслюють універсальність підходу та можливість його застосування не лише для обробки музичних сигналів, а й для IoT та сенсорних систем, де критично важливими є стійкість до шумів і мінімальна затримка.

У статті [110] акцент зроблено на прогнозуванні з урахуванням сезонних коливань параметрів води в гідропонних системах. Показано, що параметри якості води (pH , температура, концентрація поживних речовин) суттєво змінюються залежно від пори року, що ускладнює прогнозування за допомогою класичних моделей. Для розв'язання задачі запропоновано модель SARIMA (Seasonal ARIMA), яка враховує сезонні компоненти часових рядів і забезпечує точніші

прогнози порівняно з моделями, що ігнорують сезонність. Практична цінність підходу полягає у можливості масштабування на інші сфери – екологічний моніторинг, енергетику та інші сценарії, де сезонність визначає структуру даних.

1.7. Огляд підходів до використання арифметичних прискорювачів на ПЛІС на граничному рівні інфраструктури IoT

Одним із перспективних напрямів реалізації граничної обробки є використання апаратних прискорювачів, здатних виконувати обчислення з високою продуктивністю та енергоефективністю. У цьому контексті особливу увагу привертає застосування програмованих логічних інтегральних схем як граничних вузлів в системах IoT [16].

У роботі [111] автори систематизують підходи до розгортання вбудованого інтелекту на ПЛІС, охоплюючи апаратні архітектури для прискорення нейронних мереж, та виявляють, що ПЛІС є перспективною платформою для граничного штучного інтелекту завдяки балансу між гнучкістю, енергоефективністю та продуктивністю. Аналіз застосування ПЛІС у мережевій безпеці [112], зокрема для апаратного прискорення криптографічних алгоритмів, систем виявлення вторгнень та міжмережевих екранів, підкреслюючи переваги ПЛІС над програмними рішеннями у швидкодії та детермінованості. У роботі [113] розглянуто ранні застосування ПЛІС в системах IoT і окреслює потенціал реконфігурованих платформ для реалізації протоколів зв'язку та обробки сенсорних даних безпосередньо на рівні пристрою. У роботі [114] представлений огляд тенденцій розвитку програмованих апаратних платформ від класичних ПЛІС до MPSoC та структурованих спеціалізованих інтегральних схем, і обґрунтовано зближення гнучкості ПЛІС з продуктивністю спеціалізованих мікросхем. Автори [9] пропонують ПЛІС реалізацію захищеного IoT граничного пристрою на основі кастомізованого RISC–V процесора, що забезпечує апаратну ізоляцію та ефективне виконання легковагових криптографічних операцій. Робота [13] описує архітектуру для системи домашньої автоматизації IoT, демонструючи, що

реконфігуровані платформи здатні ефективно інтегрувати керування периферією, обробку даних і комунікаційні інтерфейси в єдиному апаратному рішенні.

У роботі [115] підкреслено, що однією з ключових причин використання ПЛІС в IoT є потреба локальної обробки нестабільних і зашумлених сенсорних потоків, зокрема реалізації конвеєрних архітектур цифрової обробки сигналів ковзних вікон, нелінійних і лінійних фільтрів, методів згладжування та попередніх перетворень. Автори акцентують на порівнянні швидкодії: ПЛІС здатні забезпечувати частоти обробки на рівні сотень мегагерц і одночасно працювати з кількома потоками, тоді як традиційні MCU та одноплатні комп'ютери виконують фільтрацію переважно послідовно. Окремо відзначено переваги рішень з використанням SoC (наприклад, Cyclone V або Xilinx Zynq), де апаратна логіка бере на себе обчислювально інтенсивні операції фільтрації та попередньої обробки, а вбудована ОС (наприклад Linux) забезпечує керування, мережеву взаємодію, адаптацію параметрів фільтрів і логічний контроль. Таким чином, у роботі [16] архітектура SoC розглядається як основа для багаторівневого, гнучкого й потенційно адаптивного оброблення даних у реальному часі.

У статті [17] фокус зміщено на енергоспоживання як один із визначальних факторів для пристроїв IoT. Автори демонструють, що поєднання програмованих процесорів (Nios II, MicroBlaze) з апаратними блоками на ПЛІС дає змогу гнучко керувати режимами роботи системи: застосовувати динамічну зміну тактової частоти, часткове вимкнення логіки та перерозподіл потоків, зменшуючи енергоспоживання без критичної втрати точності. Показано, що програмовані CPU під керуванням RTOS може змінювати параметри фільтрів «на льоту» відповідно до умов вхідного потоку або енергетичних обмежень. Результати експериментів у [17] підтверджують, що така програмно-апаратна синергія дозволяє досягати суттєвої економії енергії за мінімальних втрат продуктивності, що є принципово важливим для автономних граничних вузлів та сенсорних мереж.

У роботі [116] розглянуто фундаментальні засади побудови сучасних систем IoT із перенесенням значної частини обчислень на периферію мережі. Автори аргументують, що централізована хмарна обробка не завжди здатна забезпечити

вимоги практичних застосувань щодо низької затримки, локальної аналітики та автономності, зокрема у випадках нестабільного зв'язку з хмарою. З огляду на те, що значна частина сенсорних даних є зашумленою та надлишковою, у [116] підкреслюється необхідність виконання фільтрації на граничному рівні, де вузли мають забезпечувати роботу у реальному часі, мінімальну латентність і безперервність функціонування. У цьому контексті ПЛІС розглядаються як ефективні пристрої для виконання фільтрації, стискання, класифікації та попередньої аналітики завдяки паралелізму й низькій затримці. Огляд моделей взаємодії «сенсори–шлюзи–хмара», сценаріїв розподіленої аналітики й підходів енергетичної оптимізації додатково підтверджує доцільність застосування ПЛІС саме на периферійному рівні.

У статті [14] наведено широкий огляд напрямів використання ПЛІС у різних сферах IoT та зазначено еволюцію ПЛІС від суто DSP–інструментів до критичного компонента граничних та туманних обчислень. Автори виділяють ключові переваги ПЛІС: можливість формування масивно паралельних фільтрів та адаптивних обчислювальних блоків, здатність працювати при високій частоті надходження даних і підтримувати одночасну обробку багатьох потоків із високою продуктивністю та стійкістю. Також аналізуються платформи SoC, де процесор ARM відповідає за мережеву взаємодію та конфігурацію параметрів, а ПЛІС за обчислювально інтенсивні операції (фільтрацію, агрегацію, виявлення аномалій, криптографію, обробку відео та радіосигналів). Поряд із перевагами в [14] відзначаються й виклики: складність HDL–розробки, потреба у спеціалізованих інструментах і дефіцит стандартизованих модулів для IoT, однак загальний висновок полягає у тому, що ПЛІС дедалі більше розглядаються як базова платформа для фільтрації та обробки потоків у реальному часі на граничному рівні.

Практичну спрямованість використання ПЛІС в специфічних застосуваннях демонструє робота [15], у якій розглянуто високоточні медичні вузли IoT для попереднього скринінгу та прогнозування за поточними біометричними даними. Автори зазначають, що такі сигнали часто є зашумленими та потребують локальної обробки, а апаратна реалізація фільтрації, нормалізації та кореляційного аналізу на

ПЛІС забезпечує обробку в реальному часі з низькою затримкою, що є критичним для медичних сценаріїв. Додатково описано енергетичну оптимізацію через частотне масштабування, відключення блоків і розподіл обчислень між CPU та ПЛІС. Результати [15] підтверджують, що застосування ПЛІС може зменшити енергоспоживання порівняно з традиційними рішеннями з використанням SoC без погіршення точності аналізу, що робить ПЛІС доцільними для автономних переносних пристроїв.

1.8. Узагальнення проблематики, аналіз обмежень існуючих підходів та постановка задач дисертаційного дослідження

Сучасні інфраструктури IoT пройшли еволюцію від класичної тришарової моделі до гібридних багаторівневих архітектур із виділенням рівня пристроїв, граничного рівня, туманного та хмарного рівнів. Така трансформація зумовлена різким зростанням обсягів сенсорних даних, гетерогенністю пристроїв, потребою у масштабованості та вимогами до функціонування в реальному часі. Перенесення частини обчислень із хмари на периферійні вузли дозволило зменшити латентність і навантаження на мережу, однак водночас сформувало новий комплекс системних викликів.

Ключовою проблемою є відсутність узгодженості між процесами обробки даних, моніторингу стану інфраструктури, балансування навантаження та інтелектуальної аналітики. У більшості існуючих рішень ці компоненти реалізуються ізольовано, що призводить до зниження ефективності всієї системи.

Первинні сенсорні потоки характеризуються шумами, імпульсними викидами, корельованістю, надлишковістю та нестабільністю параметрів. Передавання «сирі дані» без попередньої фільтрації й агрегації створює перевантаження транспортного рівня та хмарних сервісів, підвищує витрати та знижує масштабованість. Водночас обмежені ресурси граничних вузлів (CPU, пам'ять) не дозволяють застосовувати складні алгоритми без втрати детермінованості та енергоефективності. Таким чином формується конфлікт «точність обробки ↔ швидкодія ↔ енергоспоживання».

Суттєвим обмеженням існуючих підходів є неузгодженість між:

- фільтрацією та агрегацією даних і механізмами маршрутизації та балансування навантаження;
- моніторингом ресурсів (CPU, RAM, мережа) та моніторингом якості сенсорних потоків;
- інтелектуальною аналітикою та керуючим контуром системи;
- адаптивністю алгоритмів і вимогами до детермінованості систем реального часу.

У гібридних архітектурах з'являється додаткова складність: централізовані хмарні засоби моніторингу не забезпечують повної прозорості граничного рівня, а агрегування метрик у центрі збільшує латентність і формує потенційні точки відмови. Балансування навантаження в інфраструктурах IoT також ускладнюється необхідністю врахування не лише обчислювальних ресурсів, але й часових обмежень, енергетичних характеристик, мобільності пристроїв та динаміки топології мережі.

Інтеграція нейронних модулів відкриває можливості для класифікації, прогнозування та виявлення аномалій, однак на практиці вони часто функціонують як ізольовані аналітичні компоненти. Відсутність тісної інтеграції з механізмами моніторингу, реконфігурації та балансування знижує їх практичну цінність у реальному часі. Додатковим обмеженням є висока обчислювальна та енергетична вартість прогнозу на граничному рівні.

Перспективним напрямом подолання зазначених обмежень є використання на граничному рівні арифметичних прискорювачів, які реалізовані на ПЛІС або на гетерогенних платформах SoC/ПЛІС. Це дозволяє реалізувати конвеєрну, паралельну та детерміновану обробку потоків. Проте залишаються не вирішеними такі проблеми, як інженерна складність проектування апаратних прискорювачів за допомогою мов опису апаратури (HDL) та їх інтеграція з операційними системами (ОС) та мережевими сервісами, а також забезпечення адаптивності без втрати детермінованості.

З іншого боку, у системах IoT, які працюють у режимі реального часу, важливими задачами, що постійно виконуються для підтримки інфраструктури, є моніторинг та очищення сенсорних даних, моніторинг апаратних ресурсів та мережевого оточення, забезпечення безпеки та балансування навантаження. Огляд відомих рішень показав, що незалежне виконання цих задач не гарантує високої ефективності функціонування інфраструктури IoT і обумовлює наступні проблеми:

- втрата або неточність сенсорних даних: через перевантаження мережі або збої пристроїв при відсутності контролю ресурсів.
- неповна оцінка стану системи: контроль ресурсів без інтеграції з даними сенсорів не дозволяє адекватно реагувати на зовнішнє середовище чи технологічний процес.
- проблеми безпеки: автономна реалізація заходів безпеки не захищає від помилок або некоректних даних, що можуть призвести до аварійних ситуацій.
- зниження продуктивності та збільшення затримок: балансування навантаження без урахування сенсорних і ресурсних параметрів може негативно впливати на швидкодію та затримки передачі даних.

Отже, основна гіпотеза досліджень полягає в тому, що сучасні інфраструктури IoT потребують комплексного підходу до забезпечення ефективного функціонування, який здатен поєднати в єдиному керуючому контурі:

- ефективне очищення та агрегацію даних;
- розподілений моніторинг якості потоків даних та ресурсів;
- механізми балансування навантаження з урахуванням часових та енергетичних обмежень, включно з динамічною реконфігурацією інфраструктури;
- інтегровану інтелектуальну аналітику;
- апаратне прискорення обчислювальних задач;
- забезпечення безпеки.

Можна припустити, що інтеграція цих завдань у комплексну систему обробки даних дозволить:

- забезпечити надійність і достовірність даних;
- підтримувати стабільність функціонування інфраструктури;
- приймати оперативні керуючі рішення у реальному часі;
- підвищити ефективність використання обчислювальних ресурсів і енергоспоживання.

Таким чином висунута гіпотеза обґрунтовує необхідність розроблення системного підходу до адаптивної інтелектуальної обробки даних системах IoT та формує основу для подальших досліджень.

Дисертаційне дослідження спрямоване на створення цілісної концепції адаптивно керованої та детермінованої гібридної інфраструктури IoT, зокрема у сфері промислового IoT, у якій процеси обробки даних, моніторингу, балансування навантаження та інтелектуальної аналітики інтегровані в єдиний узгоджений керуючий контур.

ВИСНОВКИ ДО РОЗДІЛУ 1

На підставі проведеного огляду літератури та аналізу сучасних підходів до побудови розподілених систем IoT встановлено, що у промислових високотехнологічних і критичних системах функціонують тисячі сенсорів і вимірювальних пристроїв, які безперервно генерують значні обсяги гетерогенних даних, формуючи надвеликі інформаційні потоки. Водночас більшість існуючих рішень, спрямованих на обробку потоків даних і забезпечення функціонування інфраструктури IoT, реалізуються як окремі незалежні компоненти або сервіси. Зокрема, це стосується обробки сенсорних даних, моніторингу стану ресурсів, забезпечення безпеки та балансування обчислювального навантаження. Такі підходи не забезпечують комплексного врахування особливостей функціонування розподілених IoT-систем, що зумовлює зниження оперативності прийняття керуючих рішень і загальної ефективності обробки даних та функціонування інфраструктури IoT.

Таким чином, актуальною є задача підвищення ефективності обробки потоків даних та функціонування інфраструктури на граничному рівні розподілених систем IoT, що функціонують в режимі реального часу та за умов надходження великої кількості різнорідних сенсорних вимірювань, багатопоточності обробки та масштабування інфраструктури шляхом забезпечення достовірності та стабільності сенсорних вимірювань; динамічного та адаптивного керування потоками даних та забезпечення оперативного прийняття керуючих рішень; підвищення безпеки функціонування інфраструктури та ефективності використання обчислювальних ресурсів.

Для досягнення поставленої мети у роботі розв'язуються такі основні завдання:

1. Обґрунтувати системний підхід до адаптивної інтелектуальної обробки даних на граничному рівні розподілених систем IoT, що передбачає побудову адаптивно-інтелектуально керованої інфраструктури IoT, у межах якої процеси обробки потоків даних, моніторингу стану ресурсів та даних, інтелектуальної

аналітики та забезпечення ефективності інфраструктури інтегровані в єдиний керуючий контур.

2. Формалізувати критерії ефективності обробки даних та функціонування інфраструктури, зокрема за показниками:

- продуктивності обробки потоків даних у реальному часі;
- затримок обробки та передавання даних;
- енергоспоживання граничних обчислювальних вузлів;
- рівня балансування обчислювального навантаження;
- надійності та безпеки функціонування моніторингової інфраструктури.

3. Розробити метод та засоби адаптивної інтелектуальної обробки даних, що забезпечують ефективне керування потоками на основі показників сенсорів, стану інфраструктури та ресурсів з урахуванням сформульованих критеріїв ефективності та вимог щодо достовірності та стабільності вимірювань, оперативного прийняття рішень, безпеки інфраструктури та ефективного використання ресурсів.

4. Розробити методику реалізації методу адаптивної інтелектуальної обробки даних на основі використання моніторингової системи Zabbix, яка може бути застосована для організації надійного моніторингу середовища та стану апаратних ресурсів у промислових системах IoT.

5. Розробити модуль багаторівневої фільтрації сенсорних даних та нейромережевий модуль прийняття рішень на гетерогенній платформі SoC/ПЛІС, а також механізми ефективної взаємодії між вбудованим процесором ARM Cortex A та програмно-конфігурованою логікою ПЛІС для ефективного перерозподілу задач і динамічної реконфігурації ПЛІС.

6. Виконати експериментальні дослідження та оцінити ефективність запропонованих рішень в умовах адаптивної інтелектуальної обробки даних на граничному рівні розподіленої інфраструктури IoT.

РОЗДІЛ 2

МЕТОД АДАПТИВНОЇ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ

2.1. Обґрунтування системного підходу до адаптивної інтелектуальної обробки даних

2.1.1. Архітектура та принципи функціонування адаптивно керованої інфраструктури IoT

У дисертаційній роботі запропоновано підхід до адаптивної інтелектуальної обробки даних в системах IoT, орієнтований на застосування в умовах промислового середовища. В рамках цього підходу запропонована адаптивно керована інфраструктура системи IoT в якій процеси обробки даних, моніторингу, балансування навантаження та інтелектуальної аналітики об'єднані в єдиний керуючий контур.

Промислові системи IoT функціонують в умовах критичних виробничих процесів, для яких характерні підвищені вимоги до безпеки, надійності та передбачуваності функціонування. На відміну від споживчих рішень IoT, промислове середовище накладає жорсткі обмеження на затримки обробки даних, автономність роботи та стійкість до відмов, оскільки помилки або затримки можуть призводити до порушення технологічних процесів і матеріальних збитків. У зв'язку з цим сучасні промислові системи IoT, як правило, розгортаються в межах локальної інфраструктури підприємства та функціонують без прямого виходу в публічний Інтернет. Такий підхід дозволяє забезпечити необхідний рівень інформаційної безпеки, зменшити латентність передачі даних та зберегти контроль над критичними виробничими ресурсами.

Запропонований підхід ґрунтується на ідеї тісного зворотного зв'язку між станом виробничого середовища, завантаженістю обчислювальних ресурсів та організацією процесу обробки даних. Дані моніторингу сенсорного середовища та стану обчислювальних вузлів використовуються для формування адаптивних керувальних рішень, які впливають на вибір алгоритмів обробки та апаратну

конфігурацію обчислювальних ресурсів. Такий підхід дозволяє динамічно адаптувати систему до змін умов експлуатації та вимог обчислювальних задач.

Архітектурною основою запропонованого підходу є трирівнева модель промислового IoT, яка забезпечує розподіл функцій збору, обробки та аналізу даних між різними рівнями системи. Трирівнева архітектура включає рівень сенсорних пристроїв, граничний рівень обробки даних та верхній рівень централізованого серверного середовища, кожен з яких виконує визначені функції у процесі адаптивної обробки даних.

У межах запропонованого підходу до адаптивної обробки даних трирівнева архітектура IoT забезпечує розподіл функцій збору, обробки та аналізу даних між рівнями системи з урахуванням специфіки промислового середовища. Структурна схема трирівневої архітектури наведена на рис. 2.1.

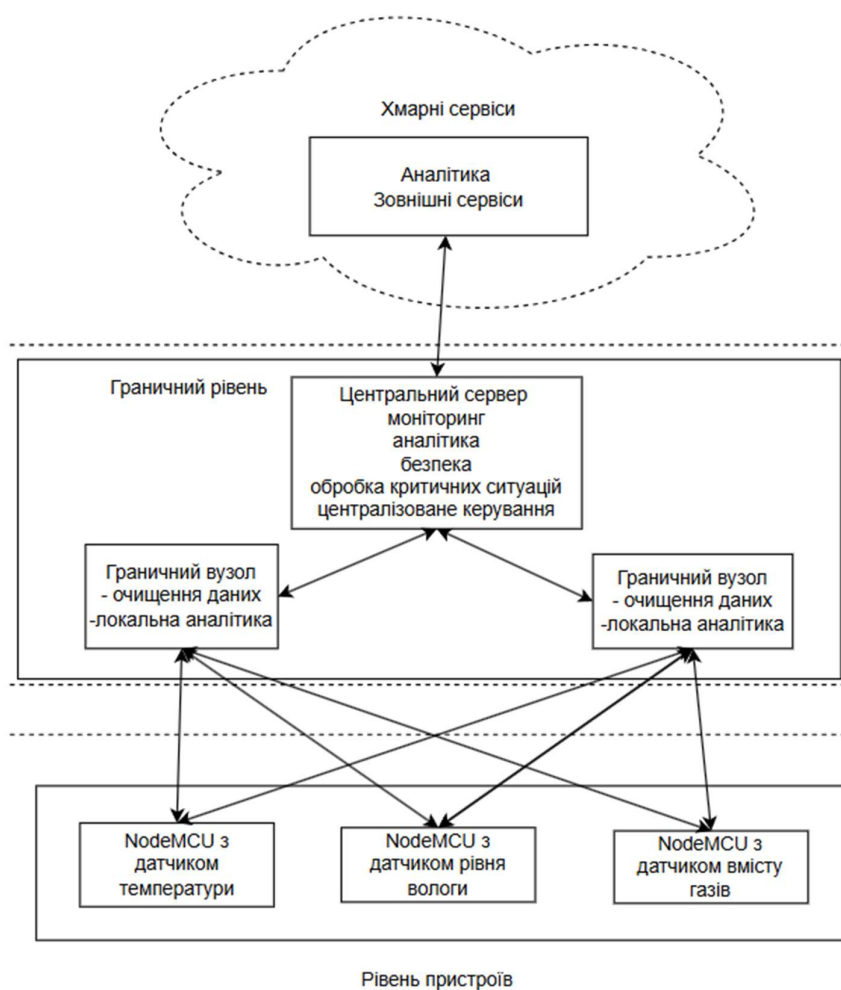


Рис 2.1. Структурна схема трирівневої архітектури

Рівень пристроїв (Device level)

На цьому рівні розташовані сенсори та виконавчі механізми, характерні для різних типів виробничих процесів, таких як конвеєрні лінії, хімічні реактори чи машинобудівні цехи. Основні вимірювані параметри включають температуру, вологість, тиск, вібрації, електричні струми та технологічні параметри процесів, що безпосередньо впливають на якість та ефективність виробництва. На показники сенсорів також впливають зовнішні фактори, які є спільними для різних типів підприємств: сезонні та добові коливання, температурний та фізичний дрейф обладнання, старіння сенсорів, виробничі завади (пил, вібрації, електромагнітні перешкоди) та критичні події.

Параметри зовнішніх впливів безпосередньо впливають на точність, достовірність та своєчасність даних, що збираються. Наявність цих впливів обґрунтовує необхідність використання адаптивних алгоритмів обробки даних на наступних рівнях системи IoT, які забезпечують надійність та точність вимірювань у реальному виробничому середовищі.

На рівні пристроїв у запропонованій адаптивній системі IoT збір даних здійснюється з сенсорів, які контролюють ключові параметри виробничого середовища. Хоча метод адаптивної обробки даних може застосовуватися в будь-яких типах промислових інфраструктур, для експериментальної реалізації були обрані найбільш характерні та вимірювані параметри: температура, вологість. Ці параметри відображають як фізичний стан середовища, так і технологічні процеси.

На точність вимірювань і якість даних в даному контексті суттєво впливають зовнішні фактори. Адаптивна обробка враховує ці фактори шляхом фільтрації даних, формування адаптивних тригерів та ініціації змін конфігурації обчислювальних вузлів, забезпечуючи надійність і точність інформації для аналітики і прийняття рішень.

Граничний рівень (Edge level)

Граничний рівень є ключовим у запропонованому підході. Система поділена на локальні домени (цехи, лінії, підсистеми), кожен з яких оснащений граничними вузлами на базі процесорів ARM та ПЛІС.

Функції граничного рівня включають:

- попередню фільтрацію сенсорних даних;
- прийняття рішень щодо апаратної реконфігурації.

Головний сервер (приватний cloud)

Граничний рівень реалізовано у вигляді дворівневої ієрхії, на верхньому рівні якої реалізовано **головний сервер** підприємства, який виконує наступні функції:

- агрегована аналітика;
- історичне зберігання даних;
- формування адаптивних тригерів;
- балансування навантаження між вузлами;
- довгострокові прогнозні моделі;
- координація локальних доменів.

Адаптивні пороги контролю та якість сенсорних вимірювань

Для підтримки адаптивності системи визначено параметри, що впливають на якість даних: шум, дрейф, нестабільність середовища, зміни режимів виробництва.

Ці параметри відстежуються, агрегуються та перетворюються на адаптивні тригери на основі: статистичних показників, трендів, відхилень від нормального режиму, завантаженості обчислювальних ресурсів.

Балансування навантаження та реконфігурація ПЛІС

Різні типи обробки даних (фільтрація, детекція аномалій, аналіз часових рядів) та різні вимоги до латентності, енергоспоживання й точності забезпечують динамічну зміну конфігурації ПЛІС; перерозподіл навантаження між граничними вузлами; активацію необхідних апаратних модулів.

Нейронні вузли на ПЛІС

На граничному рівні використано нейронні вузли на ПЛІС для: аналізу поточного стану середовища; підтримки прийняття рішень щодо типу обробки, режиму роботи та реконфігурації апаратних ресурсів.

Хмарні обчислення (Cloud)

Основними задачами хмарного рівня є організація взаємодії з зовнішніми інформаційними системами, підтримка глобальних сервісів, а також забезпечення доступу до агрегованих або неперсоналізованих даних. Через хмарну інфраструктуру можуть здійснюватися інтеграція з логістичними платформами, банківськими сервісами, міжпідприємницькими мережами та іншими глобальними інформаційними системами.

З міркувань безпеки та зменшення затримок у хмару передаються лише агреговані або несекретні дані, тоді як критична обробка інформації та прийняття оперативних рішень виконуються на рівні граничних обчислень.

Цей рівень враховується архітектурно, проте не є предметом детального дослідження у даній роботі.

Узагальнення запропоновано підходу до організації рівнів обробки даних в системах IoT наведено в таблиці 2.1, де систематизовано функціональне призначення кожного рівня, типове апаратне забезпечення та основні функції обробки інформації.

Таблиця 2.1

Систематизований опис керованої адаптивної граничної інфраструктури IoT

Функціональне призначення	Типове апаратне забезпечення	Основні функції обробки даних	Результат роботи рівня
1	2	3	4
Хмарний рівень			
Інтеграція з зовнішніми інформаційними системами та сервісами, хмарними платформами, веб-сервісами	Хмарні обчислювальні платформи; зовнішні інформаційні сервіси	інтеграція з глобальними сервісами; обмін агрегованими даними; підтримка міжсистемної взаємодії	Дані для зовнішніх систем, глобальна інтеграція

Продовження таблиця 2.1

Функціональне призначення	Типове апаратне забезпечення	Основні функції обробки даних	Результат роботи рівня
1	2	3	4
Рівень граничних обчислень			
<i>Головний граничний сервер підприємства</i>			
Централізована координація локальних доменів та виконання більш складної аналітики	Високопродуктивний сервер; централізована база даних; центральна система аналітики та моніторингу	довгострокове зберігання даних; централізована аналітика; моніторинг інфраструктури; балансування навантаження; координація граничних вузлів; централізована реконфігурація; обробка аварійних подій	Консолідована інформація про стан системи та керуючі впливи
<i>Граничні вузли (локальні домени)</i>			
Локальна обробка та попередня аналітика даних безпосередньо поблизу джерел їх формування	Одноплатні комп'ютери (Raspberry Pi); гетерогенні платформи SoC/ПЛІС; локальні граничні сервери; Мережеве устаткування	очищення та агрегування потоків; локальна аналітика; виявлення аномалій; локальне прогнозування; моніторинг ресурсів; локальне прийняття рішень; апаратна реконфігурація	Очищені та агреговані дані, локальні керуючі рішення

Продовження таблиця 2.1

Функціональне призначення	Типове апаратне забезпечення	Основні функції обробки даних	Результат роботи рівня
1	2	3	4
Рівень збирання даних			
Формування первинних даних про стан фізичних об'єктів, технологічних процесів та навколишнього середовища	Сенсори та датчики; агрегатори сигналів; мікроконтролер и (STM32); спеціалізовані вимірювальні модулі; промислові контролери	вимірювання технологічних параметрів; первинна агрегація сигналів; формування потоків сирих даних; передача даних до граничних вузлів	Потоки первинних (сирих) даних від фізичних об'єктів

Таким чином, запропонована архітектура системи IoT передбачає багаторівневу обробку даних, де первинні дані формуються на рівні сенсорної інфраструктури, основна частина обчислень виконується на рівні граничних обчислень (граничні вузли та головний граничний сервер), хмарна інфраструктура використовується переважно для інтеграції із зовнішніми інформаційними системами та глобальними сервісами.

Розподіл функцій між рівнем збору даних, граничними обчисленнями та хмарною інфраструктурою створює передумови для підвищення ефективності функціонування систем реального часу. Реалізація основної частини обробки на рівні граничних обчислень дозволяє зменшити затримки передачі даних, знизити навантаження на мережеву інфраструктуру та забезпечити оперативність прийняття керуючих рішень. Граничний рівень з реконфігуровними граничними

вузлами на базі процесорів ARM та ПЛІС, головний сервер, а також підходи до самоналаштування порогів контролю моніторингової системи, апаратні обчислювачі та нейронні вузли формують єдину технологічну екосистему на граничному рівні підприємства, у межах якої показники середовища, завантаженості вузлів та стан апаратних ресурсів трансформуються у керуючі рішення.

Функціонування системи базується на розгалуженій системі моніторингу, яка забезпечує комплексний контроль як параметрів виробничого середовища, так і стану обладнання на локальному та центральному рівні системи, що дозволяє здійснювати повноцінний адаптивний зворотний зв'язок для прийняття керувальних рішень.

2.1.2. Комплексний метод адаптивної інтелектуальної обробки даних у розподілених системах IoT

Описана тришарова архітектура та запропонований підхід до розподілу функцій обробки даних створюють основу для структурованого впровадження підходу до адаптивної інтелектуальної обробки даних.

Системний підхід до адаптивної інтелектуальної обробки даних на граничному рівні розподілених систем IoT ґрунтується на формуванні адаптивно керованої гібридної інфраструктури IoT, у межах якої процеси обробки даних, моніторингу, балансування навантаження та інтелектуальної аналітики інтегровані в єдиний керуючий контур.

Для реалізації розробленого підходу запропоновано комплексний метод адаптивної інтелектуальної обробки даних у розподілених системах IoT.

Комплексний метод адаптивної інтелектуальної обробки даних базується на інтеграції часткових методів та засобів для підвищення якості сенсорних даних, ефективного керування потоками даних, а також підвищення ефективності та безпеки функціонування інфраструктури IoT.

Для реалізації комплексного методі розроблені часткові методи та засоби, які в цілому призначені для

- адаптивного налаштування порогів контролю;
- динамічної реконфігурації ПЛІС;
- автоматичного прийняття рішень у безпекових сценаріях.
- інтелектуальної нейромережевої обробки сенсорних даних;

Часткові методи та засоби інтегровані через систему моніторингу в єдиний керуючий контур, який забезпечує динамічне прийняття рішень на двох рівнях: на рівні граничних вузлів та на рівні головного сервера.

Структура комплексного методу адаптивної інтелектуальної обробки даних представлена на рис. 2.2.



Рис. 2.2. Структура комплексного методу адаптивної інтелектуальної обробки даних

Далі представлено опис та математичні моделі часткових методів і засобів, розроблених для реалізації запропонованого комплексного методу адаптивної інтелектуальної обробки даних у промислових системах IoT. Зокрема, розглянуто механізми моніторингу, балансування навантаження, забезпечення безпеки та оптимізації використання обчислювальних ресурсів, інтегровані у єдину екосистему.

2.2. Удосконалений спосіб адаптивного налаштування порогів контролю у моніторингових IoT системах

Запропонований спосіб базується на інтеграції декількох рівнів очищення сенсорних вимірювань – перший етап, та адаптивного налаштування порогів контролю в моніторинговій системі – другий етап.

На першому етапі для очищення сенсорних вимірювань запропоновано використання комплексу відомих методів фільтрації даних, що визначено в дисертаційній роботі, як багаторівнева фільтрація. Для реалізації багаторівневої фільтрації запропоновано поєднання підходів до згладжування та усунення шумів: медіанного фільтра [117–118], фільтра ковзного середнього [74] і експоненціального згладжування [119]. Комбінування цих методів забезпечує одночасне усунення імпульсних шумів, згладжування випадкових коливань і врахування динамічних змін у часових рядах, на чому засновано формування трендів. *Тренд* – це стійка, довготривала спрямованість зміни параметра або сукупності параметрів у часовому ряді, що відображає закономірну складову динаміки процесу після усунення випадкових флуктуацій, шумів та короткочасних коливань.

Медіанний фільтр

Медіанний фільтр належить до класу нелінійних фільтрів і використовується для видалення імпульсних шумів, характерних для сенсорів із випадковими аномальними сплесками показників.

Призначення: усунення поодиноких викидів, стрибків, аномальних значень, наприклад короткі стрибки температури і вологості через електромагнітні завади.

Вхідні дані: вхідна послідовність вимірювань $x(t)$.

Для послідовності вимірювань $x(t)$, медіанний фільтр з вікном розміру $N = 2k + 1$ визначається як:

$$W(t) = \{x(t - k), x(t - k + 1), \dots, x(t), \dots, x(t + k)\}, \quad (1)$$

де $W(t)$ – вхідна послідовність, $x(t)$ – поточне значення вимірів, $x(t - 1)$ – попереднє значення вимірів.

Тоді вхідна послідовність вимірювань, яка формується через обчислення медіани локального вікна, визначається як:

$$y(t) = \text{median}(W(t)), \quad (2)$$

де $W(t)$ – вхідна послідовність, $y(t)$ – згладжене значення.

Визначено, що

- мале вікно (3 точки): висока чутливість до локальних змін, добре видаляє поодинокі викиди, але може пропускати “шумні” коливання;
- середнє вікно (4–6 точок): забезпечує компроміс між усуненням шуму та збереженням локальних трендів;
- занадто велике вікно: згладжує дані занадто сильно, може “згладити” реальні короткочасні зміни сигналу.

Фільтр ковзного середнього

Фільтр ковзного середнього належить до лінійних фільтрів і виконує перетворення вхідної послідовності $x(t)$ у вихідну $y(t)$ через усереднення локального вікна. Застосування фільтру зменшує випадкові коливання, однак частково розмиває різкі зміни сенсорних даних.

Призначення: зменшення короткочасних шумів, стабілізація показників перед подальшим аналізом.

Вхідні дані: вхідна послідовність вимірювань $x(t)$, після використання медіанного фільтра.

Математичне формулювання для вихідної послідовності даних:

$$y(t) = \frac{1}{N} \sum_{i=0}^{N-1} x(t - i), \quad (3)$$

де N – розмір вікна усереднення, $x(t)$ – вхідна послідовність

Метод експоненціального згладжування

Цей метод належить до адаптивних статистичних методів згладжування. Він використовує рекурсивну формулу, де останні спостереження мають більшу вагу, ніж попередні, що дозволяє враховувати тренди й адаптуватися до змін середовища. У порівнянні з фільтром ковзного середнього, який усереднює

рівномірно N останніх точок, експоненціальне згладжування надає більшу вагу останнім спостереженням, тобто має адаптивний характер.

Призначення: виділення довгострокових тенденцій, адаптація до поступових змін середовища та зменшення впливу короткочасних флуктуацій.

З точки зору обробки сигналів метод експоненціального згладжування реалізується, як рекурсивна фільтрація сигналу, яка визначається, як:

$$y(t) = \alpha * x(t) + (1 - \alpha) * y(t - 1), 0 < \alpha < 1, \quad (4)$$

де $x(t)$ – вхідна послідовність, $y(t)$ – згладжене значення, $y(t - 1)$ – попереднє згладжене значення, $\alpha \in (0, 1)$ – коефіцієнт згладжування (чим більший, тим сильніше враховується останнє спостереження).

На основі розглянутих методів фільтрації побудовано математичну модель багаторівневої фільтрації. Модель побудована таким чином, що вихід одного рівня фільтрації є вхідним сигналом для наступного. Такий підхід забезпечує комплексне очищення даних від різних типів шумів та формування тренду. Математична формалізація етапів багаторівневої фільтрації:

$$y_1(t) = \text{median}(W(t)), \quad (5)$$

$$y_2(t) = \frac{1}{N} \sum_{i=0}^{N-1} y_1(t - i), \quad (6)$$

$$y_3(t) = \alpha * y_2(t) + (1 - \alpha) * y_3(t - 1), \quad (7)$$

де $y_3(t)$ – вихід фільтра

Другий етап способу спрямований на реалізацію адаптивної обробки сенсорних вимірювань шляхом самоналаштування порогів контролю моніторингової системи. Досліджувана гіпотеза полягає в тому, що інтеграція цих механізмів багаторівневої фільтрації з визначенням трендів та відповідне автоматичне самоналаштування порогів контролю дозволить істотно зменшити кількість хибних сповіщень у системі моніторингу і при цьому зберегти інформативність сигналів у динамічних умовах.

Тригер це механізм спрацьовування певного контролюючого впливу в моніторингових системах. Поріг контролю – це значення або умова, при досягненні

якої спрацьовує тригер, наприклад, коли значення вимірюваного параметра перевищує поріг контролю. Традиційно в системах моніторингу широко застосовуються статичні пороги (наприклад, температура більше 70°C), однак такі налаштування не враховують сезонні та добові коливання й призводять до хибних сповіщень навіть за нормальної роботи системи. Для виявлення аномалій і відхилень від трендів запропоновано удосконалення відомих підходів з використанням адаптивних порогів контролю. При цьому поріг контролю автоматично налаштовуються залежно від результатів попередньої фільтрації та визначення тренду. Очищені й стабілізовані дані та визначенні тренди виступають вхідними сигналами для автоматичної адаптації порогів контролю.

Формалізація удосконаленого способу включає:

Перший етап – збирання сирих даних з сенсорів, які надходять на вхід багаторівневого фільтра з усіма шумами та флуктуаціями.

Другий етап – багаторівнева фільтрація, математична модель описана формулами(5–7), стабілізація часового ряду та збереження трендів, що формує базову лінію даних;

Третій етап – налаштування адаптивного порогу контролю на основі збереженої динаміки трендів, що реалізовано у вигляді наступного алгоритму:

Крок 1 – обчислення тренду: використовується потік відфільтрованих даних, за яким формуються тренди, визначаються середні значення параметрів та характерні відхилення.

Крок 2 – розрахунок адаптивних порогів контролю: поріг контролю динамічно змінюється залежно від поточного тренду, сформованого багаторівневою фільтрацією, тобто поріг «рухається» разом із базовою лінією параметра (рис. 2.3).

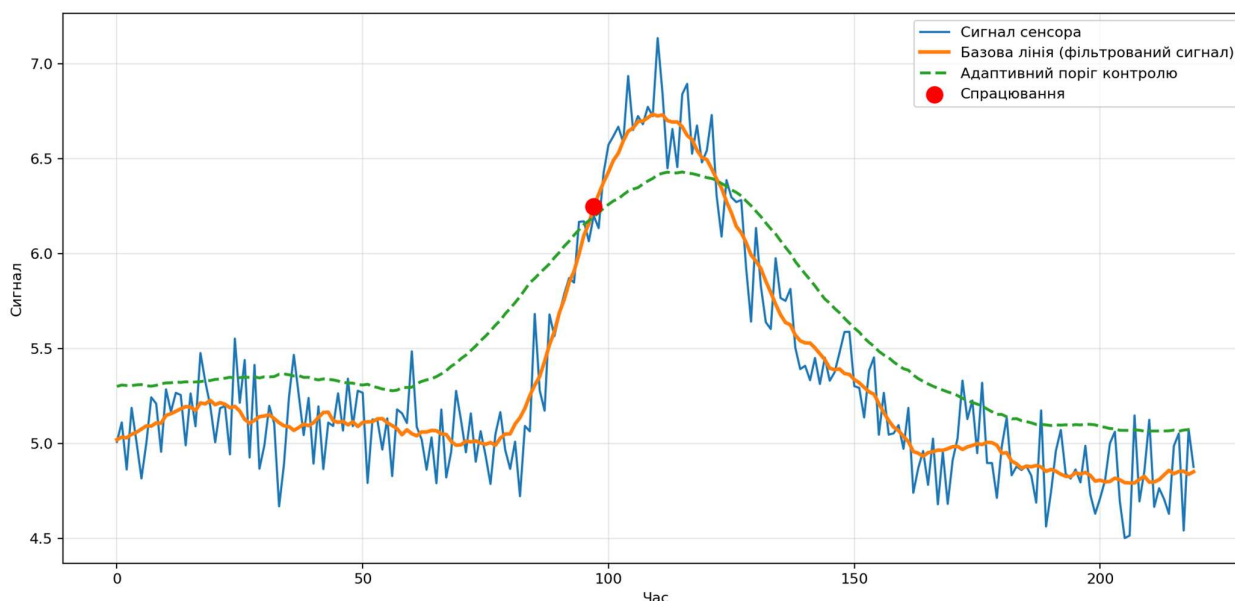


Рис. 2.3. Графік спрацювання порогу контролю на перетині адаптивного порогу та базової лінії

На рис. 2.3. зображено, що базова лінія не спрацьовує на штатні зміни і трактується як умовний рівень сигналу, який відображає реальний стан об'єкта без шуму та короткочасних флуктуацій, і формується саме багаторівневим фільтром.

Крок 3 – автоматична адаптація порогу контролю: на основі тренду система автоматично змінює пороги контролю, на які спрацьовують тригери моніторингової системи.

Блок схема алгоритму налаштування адаптивного порогу контролю зображена на рис. 2.4.

В якості прикладу наведено розрахунок тижневого тренду, що дозволяє зменшити кількість помилкових спрацювань систем моніторингу, наприклад при зростанні середньої вологості протягом тижня поріг коригується через середнє значення та стандартне відхилення з коефіцієнтом чутливості, це дозволяє враховувати сезонність і не реагувати на природні коливання.

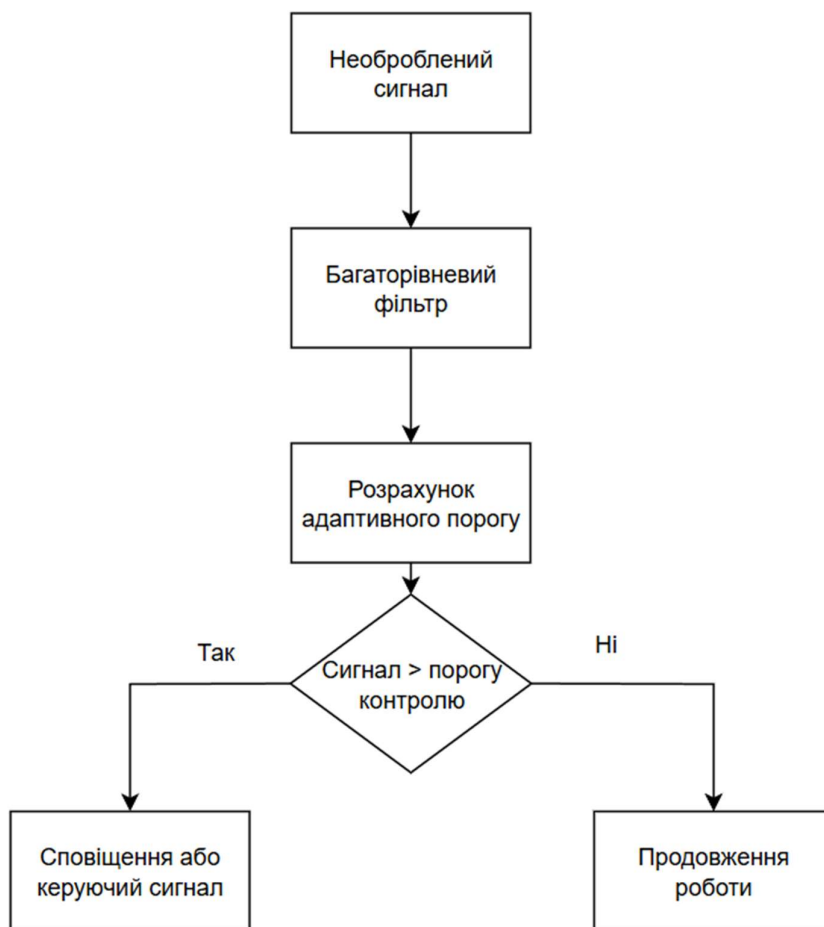


Рис. 2.4. Блок схема алгоритму налаштування адаптивного порогу контролю

Нове граничне значення порогу контролю розраховується за формулою:

$$x_{new} = x_{7day} \pm k * a_{7day}, \quad (8)$$

де, x_{new} – нове значення порогу, x_{7day} – середнє значення параметра за 7 останніх днів, k – коефіцієнт чутливості, a_{7day} – стандартне відхилення.

Запропонований метод може бути застосований для різних трендів, наприклад для добових коливань та різних показників.

Отже, запропонований спосіб адаптивної обробки сенсорних даних у моніторингових системах IoT, базується на багаторівневій фільтрації, що стабілізує часовий ряд і зберігає тренди. Удосконалення стосується інтеграції адаптивних порогів контролю відповідно до динаміки середовища на основі автоматично

налаштовуваних тригерів, що забезпечуючи зменшення хибних сповіщень і підвищення ефективності прийняття рішень у реальному часі.

2.3. Формування критеріїв ефективності використання ресурсів обчислювальних вузлів на граничному рівні системи IoT

2.3.1. Оцінка обчислювальної ефективності обробки сенсорних даних на граничному рівні IoT

Для формалізації взаємодії між елементами системи використовується графова модель, яка дозволяє описати структуру потоків даних, взаємозв'язки між вузлами. Такий підхід широко застосовується при аналізі мережевих систем [120], оскільки дозволяє формально описати топологію системи та оцінити характеристики інформаційних потоків.

$$G = (V, E), \quad (9)$$

де $V = V_D, V_E, V_S, V_C$ – множина вузлів системи, що відповідає рівню пристроїв, граничним вузлам, головному серверу підприємства(локальна хмара) та хмарному рівню, E – множина каналів передавання даних між вузлами.

$$\lambda_i = f_i * m_i, \quad (10)$$

де

λ_i – інтенсивність потоку даних від i -го сенсора за секунду часу;

f_i – частота опитування i -го сенсора;

m_i – кількість параметрів, які формує сенсор за один цикл.

$$\Lambda_D = \sum_{i=1}^N \lambda_i, \quad (11)$$

де

Λ_D – сумарна інтенсивність потоку даних на одному рівні;

N – кількість сенсорів у локальному домені.

Наведені співвідношення дозволяють формально оцінити інтенсивність формування сенсорних даних у локальному домені мережі IoT. Важливою особливістю таких систем є те, що кількість сенсорів та частота їх опитування

можуть значно варіюватися залежно від конкретного сценарію застосування. У промислових системах моніторингу кількість сенсорних вузлів може досягати сотень або тисяч одиниць, що призводить до суттєвого зростання сумарного потоку даних.

$$B_D = \sum_{i=1}^N \lambda_i * l_i, \quad (12)$$

де

l_i – розмір одного повідомлення від i -го сенсора;

B_D – інтенсивність інформаційного потоку.

$$C_{edge} = \Lambda_D * c_f + \Lambda_D * c_a + \Lambda_D * c_n \quad (13)$$

де

C_{edge} – сумарна потреба в обчисленнях;

c_f – середня кількість операцій на фільтрацію одного значення;

c_a – кількість операцій на аналіз та тригер;

c_n – кількість операцій на нейромережеву оцінку.

$$C_{edge} \leq C_{hw} \quad (14)$$

де C_{hw} – доступна обчислювальна продуктивність вузла.

Отримане співвідношення дозволяє оцінити обчислювальне навантаження, яке створюється потоком сенсорних даних на граничному вузлі системи. У запропонованій архітектурі передбачається багаторівнева обробка даних, що включає попередню фільтрацію сенсорних значень, аналіз аномалій або порогових перевищень, а також оцінювання стану середовища за допомогою нейронної моделі. Кожен з цих етапів потребує певної кількості елементарних операцій, тому сумарне навантаження на граничному вузлі зростає пропорційно інтенсивності потоку даних.

$$T_{sum} = T_D + T_{tr} + T_E + T_S \quad (15)$$

де

T_D – затримка формування первинних даних;

T_{tr} – затримка передачі;

T_E – затримка граничної обробки;

T_S – затримка обробки на серверному рівні.

Загальна затримка системи формується як сума затримок на різних рівнях архітектури IoT. У традиційних централізованих архітектурах значна частина обробки виконується у хмарному середовищі, що призводить до збільшення часу передавання даних та часу реакції системи. Перенесення частини обчислень на граничний рівень дозволяє істотно зменшити затримки передавання та підвищити швидкість реакції системи, що є критично важливим для задач моніторингу виробничих процесів, систем безпеки та автоматизованого керування.

Наприклад параметри системи:

- 20 сенсорів;
- 3 параметри на сенсор;
- частота 2 Гц.

Тоді система в одну секунду часу отримує $\Lambda_D = 120$ значень/с

Якщо після багаторівневої обробки для одного значення потрібно умовно 40 елементарних операцій, то $C_{edge} = 4800$ оп./с

Якщо система зростала до 200 сенсорів тоді $\Lambda_D = 1200$ значень/с, а $C_{edge} = 1200 \cdot 40 = 48000$ оп./с

Отримані оцінки демонструють, що навіть для відносно невеликої кількості сенсорів інтенсивність обробки даних може досягати десятків тисяч елементарних операцій за секунду. При подальшому масштабуванні системи, характерному для промислових мереж IoT, це навантаження зростає пропорційно кількості сенсорів. У таких умовах використання лише програмних методів обробки на центральному процесорі може призводити до зростання затримок та зниження енергоефективності системи.

Саме тому доцільним є використання апаратних прискорювачів на базі програмованих логічних інтегральних схем, які дозволяють реалізувати паралельну

обробку поточкових даних та забезпечити необхідну продуктивність при мінімальних затримках.

2.3.2. Комплексний критерій ефективності обчислювальних ресурсів

Для кількісної оцінки ефективності використання обчислювальних ресурсів у розподіленій системі IoT запропоновано комплексний критерій, який інтегрує три базові показники продуктивності: швидкодію обробки даних, енергоефективність та часову затримку. Такий підхід дозволяє враховувати не лише абсолютну продуктивність вузла, але й його енергетичні витрати та відповідність вимогам реального часу, що є критично важливим для гранично-орієнтованих архітектур.

Критерій K_1 – інтенсивність обробки даних

$$K_1 = \frac{N}{T} \quad (16)$$

де N – кількість елементів даних, T – загальний час обробки.

Показник K_1 характеризує обчислювальну продуктивність вузла та визначається як кількість значень (вимірювань, семплів або ознак), опрацьованих за одиницю часу. Фактично цей параметр відображає пропускну здатність обчислювального модуля.

Чим більшим є значення K_1 , тим вищою є швидкодія системи. У контексті реалізації на ПЛІС або граничного модуля цей критерій демонструє здатність системи працювати з поточковими даними без накопичення черги обробки.

Критерій K_2 – енергетичні витрати

$$K_2 = \frac{E}{N} \quad (17)$$

де N – кількість елементів даних, E – витрачена енергія

Показник K_2 характеризує кількість енергії, спожитої обчислювальним вузлом за одиницю часу. Він відображає енергетичну ефективність роботи модуля та є особливо важливим для автономних пристроїв IoT із батарейним живленням.

Менше значення K_2 відповідає більш енергоефективному режиму функціонування. Таким чином, при формуванні інтегрального показника доцільно враховувати нормалізоване або інверсне представлення цього критерію (щоб зменшення енергоспоживання підвищувало загальну оцінку ефективності).

Критерій K_3 – затримка обробки

$$K_3 = \frac{T}{N} \quad (18)$$

де N – кількість елементів даних, T – загальний час обробки.

Показник K_3 відображає часову затримку між моментом надходження даних до системи та отриманням результату обробки. Цей параметр є визначальним для систем реального часу, зокрема для задач моніторингу, адаптивної реконфігурації або нейромережевої обробки на ПЛС.

Менша затримка означає швидшу реакцію системи на зміну стану середовища. Як і у випадку з енергоспоживанням, для інтегрального оцінювання доцільно використовувати нормалізоване представлення показника.

Для отримання комплексного показника спочатку потрібно нормалізувати значення за наступними формулами:

Для швидкодії, де більше значення краще

$$K_{1norm} = \frac{K_1}{K_{1max}} \quad (19)$$

Для витрачання енергії, де менше значення краще

$$K_{2norm} = \frac{K_{2min}}{K_2} \quad (20)$$

Для затримок, де менше значення краще

$$K_{3nor} = \frac{K_{3min}}{K_3} \quad (21)$$

Загальний критерій ефективності K_s визначається як зважена сума трьох складових:

$$K_s = \frac{1}{3}(K_{1norm} + K_{2norm} + K_{3n} \quad) \quad (22)$$

Таким чином, кожен із критеріїв має однакову вагу (33 %), що забезпечує збалансований підхід до оцінювання. Вибір рівних ваг обумовлений необхідністю одночасного врахування продуктивності, енергоефективності та часових характеристик без пріоритетного домінування будь-якого з параметрів.

Умова стабільної роботи вузла:

$$K_1 \geq \Lambda_D \quad (23)$$

де K_1 – інтенсивність обробки даних на вузлі, Λ_D – інтенсивність вхідного потоку.

Метод доцільно застосовувати, якщо одночасно виконується:

$$\Lambda_D > \Lambda_{crit}, \quad (24)$$

$$T_{CPU} > T_{req}, \quad (25)$$

$$E_{CPU} > E_{FPGA}, \quad (26)$$

де

Λ_{crit} – критична інтенсивність потоку, після якої програмна обробка стає неефективною;

T_{req} – допустима затримка системи;

E_{CPU}, E_{FPGA} – енергетичні витрати відповідних платформ.

У випадку спеціалізованих застосувань (наприклад, автономні сенсорні вузли або системи з жорсткими вимогами до латентності) вагові коефіцієнти можуть бути адаптовані відповідно до цільових функціональних вимог.

Запропонований комплексний критерій забезпечує узгоджену оцінку ефективності функціонування обчислювальних ресурсів у тришаровій архітектурі з урахуванням вимог реального часу, енергоефективності та надійності.

2.3.3. Інтегральний показник завантаженості вузла для прийняття рішення щодо реконфігурації

В контексті промислового IoT важливо виділити наступні групи обладнання та параметри цього обладнання які були проаналізовані.

Перша група – системи збирання та обробки даних на границі IoT: обчислювальні вузли на базі мікропроцесорних і мікроконтролерних платформ. Параметри які оцінюються:

- Продуктивність CPU
- Обсяг RAM
- Кількість інтерфейсів (RS-485, Ethernet, CAN)
- Підтримка протоколів (Modbus, OPC UA, MQTT)
- Затримка обробки
- Стійкість до відмов
- Робочий температурний діапазон

Друга група – мережеве устакунання: комутатори, маршрутизатори, модеми, VPN-шлюзи. Параметри:

- Використання RAM
- Пропускна здатність
- Використання CPU
- Кількість помилкових/відхилених пакетів
- Резервування
- Робочий температурний діапазон

Третя група – системи моніторингу та аналітики: сервери, віртуалізовані середовища, хмарні платформи. Параметри:

- Пропускна здатність мережі
- Масштабованість
- Відмовостійкість (кластеризація)
- Доступний об'єм зберігання даних

Для оцінювання ефективності обчислювальних ресурсів за параметрами завантаження запропоновано використання інтегрального показника поточного навантаження граничного вузла.

Для формалізації процесу оцінювання завантаження серверів запропоновано математичну модель моніторингу навантаження, яка базується на аналізі

усереднених значень параметрів за останні n вимірів. У межах моделі визначається завантаженість оперативної пам'яті N_m , завантаженість процесора N_p та завантаженість дискового простору N_d , які обчислюються як доповнення до середнього значення відсотка вільних ресурсів за відповідними параметрами.

$$N_m = 100 - \frac{1}{n} \sum_{i=1}^n x_i \quad (27)$$

де x_i – відсоткове значення виміру кількості вільної пам'яті на пристрої, N_m – завантаженість пристрою по пам'яті.

$$N_p = 100 - \frac{1}{n} \sum_{i=1}^n y_i \quad (28)$$

де y_i – відсоткове значення простоювання процесора на пристрої, N_p – завантаженість пристрою по процесору.

$$N_d = 100 - \frac{1}{n} \sum_{i=1}^n s_i \quad (29)$$

де s_i – відсоткове значення кількості вільного місця на диску, N_d – завантаженість пристрою по диску.

$$P = N_m * 0.15 + N_p * 0.8 + N_d * 0.05 \quad (30)$$

Інтегральний показник поточного навантаження сервера визначається як зважена сума цих складових відповідно до виразу (30). Вибір вагових коефіцієнтів обґрунтовано ступенем впливу кожного ресурсу на продуктивність обчислень. Найбільший коефіцієнт надано процесору, оскільки його повне завантаження призводить до істотних затримок навіть за наявності вільної пам'яті та дискового простору. Менші коефіцієнти обрано для пам'яті та дискового простору з урахуванням характеру обчислень, які не потребують великих буферів даних і використовують дисковий простір переважно для збереження результатів. Залежно від типу даних і принципів їх обробки коефіцієнти можуть бути змінені, також при використанні інших засобів обробки даних, формула розширюється з урахуванням нових параметрів та зміною коефіцієнтів.

Після визначення інтегрального показника поточного навантаження вузла доцільно формалізувати умови прийняття рішення щодо перерозподілу

обчислювальних задач між доступними вузлами системи. Для цього використовується порівняння отриманого значення інтегрального показника з граничними значеннями навантаження, визначеними на основі експериментальних досліджень або вимог до функціонування системи.

У межах запропонованої моделі кожному вузлу системи відповідає значення інтегрального показника навантаження P_i , яке обчислюється за формулою (30).

Умовою перевищення допустимого рівня навантаження є виконання нерівності

$$P_i > P_{max} \quad (31)$$

де P_i – інтегральний показник поточного навантаження i -го вузла системи, P_{max} – гранично допустиме значення навантаження, встановлене для забезпечення стабільної роботи системи.

Якщо виконується умова (31) це свідчить про перевищення допустимого рівня використання ресурсів вузла. У такому випадку формується керуюча дія, спрямована на зменшення навантаження. Така дія може включати перенесення обчислювальних задач на інші вузли системи, зміну маршрутизації потоків даних, запуск резервних сервісів або динамічну реконфігурацію мережевої інфраструктури.

Умовою застосування балансування є умова (31) протягом τ секунд, коли час затримки обробки знаходиться в допустимих межах.

Для визначення вузла, на який доцільно перенести частину обчислювального навантаження, використовується правило вибору вузла з мінімальним значенням інтегрального показника навантаження. Формально цей процес описується виразом:

$$P_j = \min(P_1, P_2, \dots, P_n) \quad (32)$$

де $P_1, P_2, \dots, P_n$ – значення інтегрального показника навантаження для всіх доступних вузлів системи, P_j – мінімальне значення серед них, що відповідає вузлу з найменшим рівнем використання ресурсів.

Таким чином, вузол j визначається як найбільш придатний для прийняття додаткових обчислювальних задач. Перенесення навантаження на вузол із мінімальним значенням інтегрального показника дозволяє забезпечити більш рівномірний розподіл обчислювальних ресурсів у системі та запобігти перевантаженню окремих елементів інфраструктури.

2.4. Метод автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT

Запропонований метод прийняття рішень для реакції на інциденти та динамічної зміни топології мережевої системи базується на поєднанні безперервного моніторингу, формалізованих алгоритмів аналізу стану та автоматизованого виконання керуючих дій. Метод орієнтований на автономне функціонування у розподілених мережах і передбачає мінімізацію участі адміністратора на етапах первинного виявлення та реагування.

Метод реалізується у вигляді набору взаємопов'язаних алгоритмів реакції, кожен з яких відповідає за обробку окремого класу подій безпеки або аномальних станів. Алгоритми мають уніфіковану структуру та реалізують послідовність дій: *«збір даних → аналіз і зіставлення з політиками → прийняття рішення → виконання керуючого впливу → інформування та аудит»*.

На рис. 2.5 наведено схему, яка відображає основні компоненти методу та взаємозв'язок трьох алгоритмів реагування

Далі представлено деталізація алгоритмів.

Алгоритм 1. Реакція на встановлення з'єднання з IP-адресами поза білим списком

Даний алгоритм призначений для автоматизованого виявлення та нейтралізації мережевих з'єднань із віддаленими вузлами, які не відповідають політикам доступу. Така поведінка інтерпретується як потенційна загроза інформаційній безпеці або ознака неконтрольованої активності програмного забезпечення.

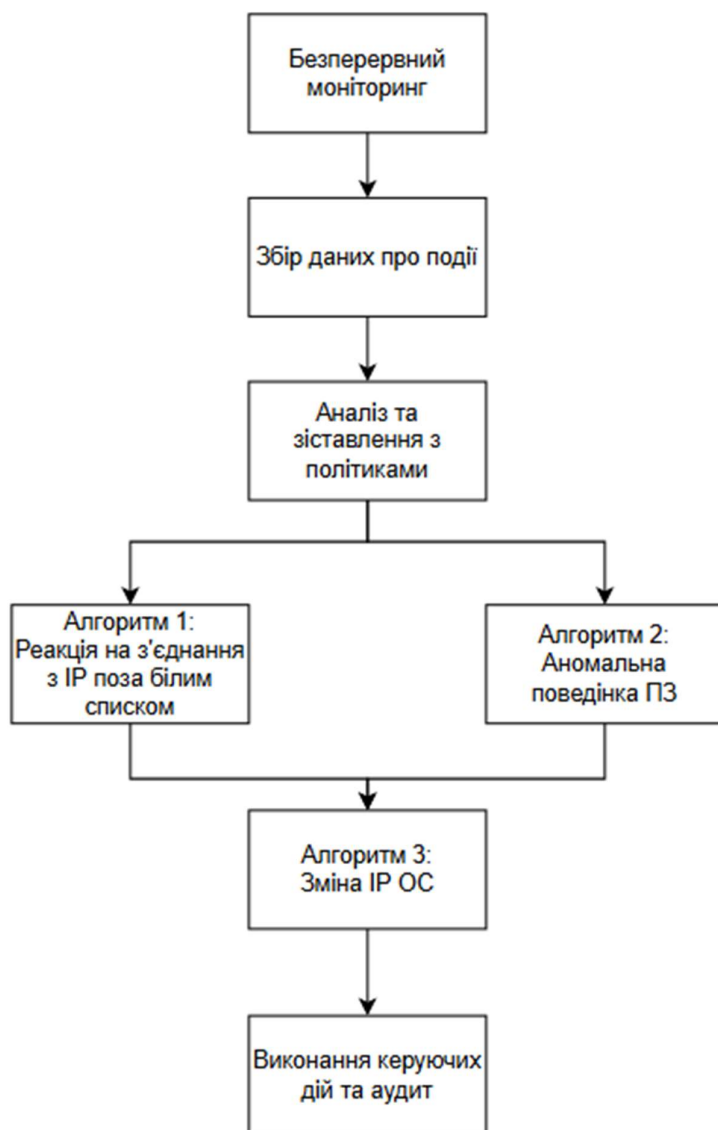


Рис. 2.5. Схема взаємодії основних компонентів методу

Крок 1. Формування політики дозволених мережевих з'єднань

На початковому етапі визначається множина легітимних IP-адрес, з якими дозволено встановлення з'єднань. Ця множина реалізується у вигляді білого списку, що зберігається у файлі. Такий підхід забезпечує централізоване керування політикою доступу та можливість її оперативного коригування без внесення змін у програмну логіку алгоритму.

Крок 2. Автоматизований збір параметрів активних мережевих з'єднань

Для отримання актуальної інформації про стан мережевих підключень на ОС використовується PowerShell-скрипт інтегрований в систему моніторингу.

Відповідний PowerShell-скрипт аналізує таблицю TCP-з'єднань, відбирає лише активні з'єднання у стані Established та формує список віддалених IP-адрес.

Отримані дані передаються до зовнішнього алгоритму обробки, який виконує нормалізацію, уніфікацію та підготовку даних до подальшого аналізу.

Крок 3. Логічний аналіз та зіставлення з білим списком

На цьому етапі виконується поелементне порівняння кожної виявленої IP-адреси з дозволим переліком. Для кожної адреси формується бінарна ознака відповідності політиці безпеки. Адреси, відсутні у білому списку, класифікуються як порушення, що потребують негайної реакції.

Крок 4. Прийняття рішення та ініціація керуючих дій

У разі виявлення IP-адреси, що не відповідає політиці доступу, активується набір інструкцій блокування, який слугує формальним механізмом прийняття рішення. Інструкції включають:

- встановлення змінних з IP адресами, портами та типами підключення які потрібно заблокувати;
- створення правила блокування вхідного або вихідного підключення;
- фіксацію інциденту у системі моніторингу;
- надсилання повідомлень адміністраторам відповідно до рівня критичності події.

Крок 5. Після Інцидентний аналіз та коригування політик

Після автоматичного реагування виконується аудит інциденту. Адміністратори аналізують джерело підключення, пов'язаний процес або сервіс, а також обґрунтованість автоматично застосованих обмежень. За потреби вносяться зміни до білого списку або загальних політик мережевої безпеки.

Алгоритм 2. Реакція на аномальну поведінку програмного забезпечення та перебудову мережевої топології

Другий алгоритм методу орієнтований на виявлення аномальних станів ОС, що проявляються у вигляді нетипового споживання обчислювальних ресурсів і можуть бути індикатором компрометації або несанкціонованої активності.

Крок 1. Збір та структуризація даних про активні процеси

На ОС здійснює збір даних про активні процеси та рівень їх завантаження CPU. Дані передаються у структурованому форматі та зберігаються у відповідному елементі даних системи моніторингу.

Для подальшого аналізу використовується додаткова обробка та відбір процесів, що перевищують задані порогові значення.

Крок 2. Формування профілю допустимої поведінки

Визначається білий список дозволених процесів та граничні значення їх ресурсного споживання. Таким чином формується еталонний профіль нормальної поведінки хоста, з яким порівнюються поточні вимірювання.

Крок 3. Виявлення відхилень та прийняття рішення

У разі виявлення процесів, які перевищують допустимі пороги та відсутні у білому списку, система формує ознаку аномалії та активує відповідний набір інструкцій.

Крок 4. Реакція у вигляді динамічної зміни мережевих параметрів

Після виявлення аномалії автоматично запускається алгоритм зміни IP-адреси хоста з аномальною активністю та перебудови маршрутизації.

Таке рішення дозволяє ізолювати потенційно небезпечний вузол, зберігаючи працездатність загальної мережевої інфраструктури.

Крок 5. Аудит та валідація прийнятого рішення

На завершальному етапі адміністратори аналізують результати автоматичного реагування, оцінюють його ефективність та приймають рішення щодо збереження або скасування внесених змін. За результатами аналізу політики безпеки можуть бути уточнені або розширені.

Алгоритм 3. Зміна IP адреси ОС

Крок 1. Підготовка до зміни IP

На цьому етапі здійснюється вибір параметрів роботи алгоритму а саме визначаються доступні IP адреси, які можна використати, параметри мережевого обладнання на якому будуть внесені зміни.

Крок 2. Встановлення правил

Для окремих елементів системи наявні підготовлені списки обов'язкових доступів для коректної роботи, якщо вузол є критичним для системи, у іншому випадку він може бути повністю заблокованим. На цьому етапі до мережевого обладнання вносяться правила маршрутизації та доступів відповідно до нової IP адреси.

Крок 3. Зміна IP в системах моніторингу

Відповідно для продовження аналізу роботи вузла системи, вносяться зміни IP адреси в наявні системи моніторингу.

Крок 4. Зміна IP адреси на кінцевому вузлі

На цьому етапі виконується зміна IP адреси на кінцевому вузлі за допомогою наявних інструкцій на цьому вузлі та IP адреси яка була визначена для заміни на етапі 1.

Крок 5. Вимкнення старих правил маршрутизації

Після успішної зміни IP адреси кінцевого вузла, проходить процедура вимкнення старих правил маршрутизації, що б зменшити кількість правил які потрібно перевіряти системі маршрутизації для розподілу пакетів в системів.

Крок 6. Аудит та валідація прийнятого рішення

Адміністратори виконують розслідування, оцінюють доцільність збереження змін, фіксують інцидент і за потреби коригують політики безпеки.

Перший алгоритм спрямований на виявлення та блокування мережових з'єднань із IP-адресами, що не входять до білого списку. Він забезпечує автоматичне формування політик доступу, аналіз активних ТСП-з'єднань, прийняття рішення про блокування та фіксацію інциденту.

Другий алгоритм орієнтований на виявлення аномальної поведінки процесів (наприклад, перевищення використання CPU). У разі виявлення відхилень система ініціює зміну мережових параметрів вузла, зокрема IP-адреси, з метою його ізоляції та збереження стабільності мережі.

Третій алгоритм деталізує процедуру зміни IP-адреси: підготовку параметрів, внесення правил маршрутизації, оновлення даних у системах моніторингу, зміну адреси на вузлі та аудит виконаних дій.

У цілому метод забезпечує адаптивне, автоматизоване та політико-орієнтоване реагування на інциденти безпеки з можливістю подальшої валідації та коригування політик.

2.5. Спосіб динамічної реконфігурації граничних обчислювальних вузлів

Реконфігурація обчислювальної структури на базі ПЛІС, що функціонує під керуванням операційної системи Linux, передбачає використання наперед підготовленого набору файлів прошивки та автоматизований механізм їх завантаження. Кожен файл прошивки відповідає певній конфігурації апаратної логіки й формується користувачем з урахуванням конкретної версії плати та вимог до ресурсів.

Система підтримує зберігання декількох файлів прошивки (формату *rbf*), між якими може здійснюватися перемикання залежно від поточного режиму роботи. Ініціація процесу реконфігурації може виконуватися автоматично за розкладом із використанням стандартних механізмів планування завдань в ОС Linux (*cron*) або за зовнішньою командою.

Алгоритм реконфігурації ПЛІС (рис 2.6) включає такі етапи:

Крок 1. Підготовка та розміщення файлів прошивки.

Користувач заздалегідь формує набір файлів прошивки у форматі *rbf*, оптимізованих за розміром та сумісних із конкретною апаратною платформою. Підготовлені файли копіюються у файлову систему Linux та зберігаються у визначеному каталозі для подальшого використання.

Крок 2. Ініціація процесу реконфігурації.

Процес прошивки запускається автоматично за зовнішньою командою або за внутрішнім таймером системи. Перед початком реконфігурації здійснюється коректне завершення поточних операцій ПЛІС.

Крок 3. Зупинка взаємодії між ОС та ПЛІС.

На цьому етапі тимчасово вимикаються канали обміну даними (мости) між операційною системою та ПЛІС для запобігання помилкам доступу під час перепрограмування.



Рис 2.6. Алгоритм реконфігурації ПЛІС

Крок 4. Завантаження обраної прошивки.

Виконується прошивка ПЛІС обраним *rbf*-файлом відповідно до заданого сценарію реконфігурації.

Крок 5. Відновлення взаємодії та перевірка працездатності.

Після завершення прошивки канали обміну між ОС і ПЛІС знову активуються. Далі проводиться перевірка коректності роботи завантаженої

конфігурації, зокрема шляхом контролю доступності пристрою та базових функціональних тестів.

Запропонований алгоритм забезпечує гнучку та безпечну реконфігурацію обчислювальної структури ПЛІС без зупинки всієї системи та дозволяє адаптувати апаратну логіку до поточних потреб у автоматичному режимі.

2.6. Спосіб інтелектуальної нейромережевої обробки сенсорних даних на ПЛІС

2.6.1. Концепція нейромережевого модуля прийняття рішень та обґрунтування формування первинного простору ознак

На різноманітних промислових об'єктах і виробництвах критичними можуть бути параметри стану робочого середовища. При цьому для різних типів об'єктів застосовуються різні сценарії моніторингу та контролю. Зокрема, на виробничих підприємствах важливими є показники безпеки робочого місця, рівень освітленості робочих зон, температура та вологість повітря. На хімічних підприємствах особливу увагу приділяють контролю чистоти повітря, концентрації шкідливих газів і парів, а також рівню вентиляції. На енергетичних об'єктах важливими можуть бути параметри температури обладнання, вібраційні характеристики та рівень шуму. У складських і логістичних комплексах контролюються умови зберігання продукції, зокрема температура, вологість та концентрація пилу.

Запропонований нейромережевий модуль прийняття рішень призначений для розширення функціональних можливостей системи моніторингу та забезпечення інтелектуального аналізу параметрів оточуючого середовища. Його використання дозволяє автоматично визначати відхилення параметрів середовища від нормативних значень, виявляти аномальні ситуації та формувати узагальнені оцінки стану контрольованого середовища.

З метою навчання нейромережевого модуля було проаналізовано декілька відкритих наборів даних, що містять сенсорні вимірювання параметрів навколишнього середовища. На основі цих датасетів було розглянуто різні

комбінації контрольованих параметрів, таких як температура, вологість, рівень освітленості, концентрація газів, рівень шуму та інші характеристики середовища.

Запропонований спосіб інтелектуальної нейромережевої обробки сенсорних даних є універсальним і може застосовуватися для роботи з різними наборами даних, отриманими від різних типів сенсорів та вимірювальних систем.

В якості прикладу для експериментального дослідження запропоновано використати чотири типи сенсорних вимірювань, які є найбільш характерними для систем моніторингу виробничого середовища. Вибір цих параметрів обумовлений тим, що вони безпосередньо впливають на умови перебування людини в робочій зоні, визначають якість повітряного та мікрокліматичного режиму приміщення і можуть бути використані як інформативні показники відповідності середовища встановленим вимогам безпеки та допустимості.

Тоді первинний простір ознак формується з потоку сенсорних вимірювань, що включають такі параметри:

- температура повітря (x_1);
- відносна вологість (x_2);
- рівень освітленості (x_3);
- концентрація вуглекислого газу (CO_2)(x_4);

У кожний момент часу на основі цих вимірювань формується вектор ознак

$$X = (x_1, x_2, x_3, \dots, x_n), \quad (33)$$

де x_i – значення відповідного сенсорного параметра.

Цей вектор описує поточний стан контрольованого середовища і є елементом первинного простору ознак, що використовується для подальшої обробки та формування стабілізованого простору ознак, придатного для аналізу нейронною мережею.

2.6.2. Опис способу інтелектуальної нейромережевої обробки сенсорних даних

Запропонований спосіб інтелектуальної нейромережевої обробки сенсорних даних ґрунтується на концепції поетапного формування інформації для нейронної мережі з урахуванням обмежень і особливостей апаратної реалізації в середовищі ПЛІС. Ключовою ідеєю запропонованого способу є розділення процесів вимірювання, стабілізації та навчання, що дозволяє забезпечити узгодженість статистичних властивостей вхідних даних та підвищити стійкість нейронної моделі до шумів і випадкових збурень.

На першому етапі розглядається формування первинного простору спостережень, який складається з багатовимірних потоків сенсорних вимірювань. Ці вимірювання мають стохастичний характер і містять як корисну інформацію, так і похибки, обумовлені фізичними властивостями сенсорів та впливом зовнішнього середовища. У межах способу передбачається, що безпосереднє використання таких даних для навчання нейронної мережі є недоцільним через нестабільність їх статистичних характеристик.

Другий етап полягає у перетворенні первинних сенсорних сигналів у стабілізований простір ознак. Для цього вводиться апаратний рівень фільтрації, який реалізує операції згладжування та усунення аномальних значень. Теоретично цей етап розглядається як оператор зменшення дисперсії випадкових компонентів сигналу при збереженні його повільних трендів. У результаті формується детерміноване подання середовища, придатне для подальшого аналізу нейронною мережею.

На первинному рівні кожен із зазначених параметрів надходить до системи у вигляді часової послідовності вимірювань, що містить як корисну інформацію про фактичний стан середовища, так і випадкові коливання, шумові компоненти та можливі одиничні аномальні відхилення. Саме тому безпосереднє використання таких значень у процедурі інтелектуальної оцінки є недоцільним. На другому етапі способу над цими даними виконується фільтрація та стабілізація, внаслідок чого

формується згладжене представлення кожного фізичного параметра, придатне для подальшого аналізу.

Після виконання фільтрації отримані значення сенсорних параметрів зіставляються з наперед заданими пороговими межами безпечного робочого середовища. Такі межі визначають еталонні або нормативно допустимі інтервали для температури, вологості, освітленості та концентрації CO₂, в межах яких умови середовища розглядаються як прийнятні для функціонування системи та перебування людини. У випадку виходу параметра за допустимі межі формується числова характеристика відхилення, яка відображає ступінь невідповідності поточного стану еталонним вимогам.

Для подальшого використання вектору первинних ознак у задачах інтелектуальної обробки необхідно врахувати, що його компоненти мають різну фізичну природу, різні одиниці вимірювання та різні діапазони зміни. Зокрема, температура задається у градусах Цельсія, відносна вологість – у відсотках, освітленість – у люксах, а концентрація вуглекислого газу – у ppm. Без попереднього приведення таких величин до узгодженого масштабу їх безпосереднє використання в межах єдиної моделі призводить до ситуації, коли ознаки з більшим числовим діапазоном починають домінувати над іншими не через більшу інформативність, а лише через масштаб представлення. Це ускладнює навчання нейронної мережі, погіршує збіжність алгоритмів оптимізації та знижує інтерпретованість результатів.

З метою усунення зазначеного недоліку в межах запропонованого способу вводиться процедура нормалізації первинного простору ознак. Нормалізація розглядається як відображення початкового простору фізичних вимірювань у безрозмірний простір, придатний для об'єднання різнорідних параметрів у межах одного вхідного вектора. Для кожного параметра x_i формується нормалізоване значення $\tilde{x}_i$, що належить інтервалу $[0, 1]$, за правилом перетворення:

$$\tilde{x}_i = \frac{x_i - x_i^{\min}}{x_i^{\max} - x_i^{\min}}, \quad (34)$$

де $x_i^{\min}$ та $x_i^{\max}$ – відповідно мінімально та максимально допустимі або характерні значення i -го параметра, визначені на основі технічних характеристик сенсора, нормативних вимог або статистики попередніх спостережень.

У результаті нормалізації та інтеграції параметрів формується узагальнений показник стану середовища, значення якого належить інтервалу від 0 до 1. Значення, близькі до 1, відповідають режимам, у яких параметри середовища перебувають у межах безпечних або близьких до еталонних значень. Натомість значення, близькі до 0, характеризують середовище з істотними відхиленнями від допустимих умов. Таким чином, результат нормалізації набуває змісту інтегральної оцінки відповідності робочого середовища критеріям безпеки, що надалі використовується або як цільова змінна для навчання нейронної мережі, або як еталонний індикатор під час верифікації моделі.

Такий підхід дозволяє перейти від аналізу окремих фізичних величин до оцінювання цілісного функціонального стану середовища. На відміну від розгляду кожного параметра ізольовано, інтегральний нормалізований показник враховує їх сукупний вплив і відображає загальний ступінь відповідності умов праці встановленим критеріям безпечного робочого середовища. Це створює методичну основу для подальшого формування часових векторів ознак та побудови нейромережевого модуля інтелектуальної інтерпретації стану середовища.

На третьому етапі здійснюється агрегування відфільтрованих параметрів у часові вектори ознак. Кожен навчальний приклад описує не миттєвий стан системи, а усереднений стан за певний часовий інтервал, що дозволяє враховувати інерційні властивості фізичних процесів. Додатково вводяться похідні характеристики, які відображають відхилення поточних параметрів від індивідуально заданих комфортних значень. Таким чином, простір ознак набуває семантичного змісту, пов'язаного з оцінкою якості середовища.

Важливою складовою запропонованого способу є формування навчального датасету, який забезпечує узгоджений перехід від первинних сенсорних вимірювань до задачі нейромережевого оцінювання стану середовища. На відміну від підходів, у яких для навчання використовуються миттєві неструктуровані

значення сенсорів, у межах даної роботи датасет формується на основі попередньо відфільтрованих, агрегованих та нормалізованих даних. Такий підхід дозволяє знизити вплив випадкових збурень, зменшити варіативність вибірки, обумовлену шумами й одиничними викидами, та забезпечити статистичну узгодженість вхідного простору ознак.

Формально навчальний датасет подається у вигляді множини пар

$$\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad (35)$$

де $x_i \in \mathbf{R}$ – вектор ознак, сформований за результатами обробки сенсорних даних на певному часовому інтервалі, y_i – цільова змінна, що відображає інтегральну оцінку стану середовища, а N – кількість сформованих прикладів у вибірці. Кожен вектор x_i включає як абсолютні значення контрольованих параметрів після фільтрації, так і похідні характеристики, що описують ступінь відхилення цих параметрів від заданих порогових значень. Зазначені порогові значення можуть трактуватися як комфортні межі функціонування середовища або як нормативно допустимі межі відповідно до вимог безпеки праці та умов експлуатації.

Таким чином, у межах датасету реалізується поєднання двох типів інформації: фактичного сенсорного стану середовища та його нормативно-орієнтованої інтерпретації. Це дозволяє подати на вхід нейронної мережі не лише поточні числові вимірювання, а й ознаки, які мають прикладний зміст з точки зору оцінювання якості середовища. У результаті простір ознак стає більш інформативним, а модель набуває здатності апроксимувати не окремі фізичні параметри, а узагальнений функціональний стан об'єкта спостереження.

Четвертий етап пов'язаний з формуванням цільової змінної для навчання нейронної мережі. В якості еталонного значення використовується суб'єктивна оцінка якості середовища або його інтегральна характеристика. У теоретичному сенсі така оцінка розглядається як відображення латентного стану системи, який не може бути безпосередньо виміряний сенсорами, але може бути апроксимований на основі їх сукупності за допомогою нейронної моделі.

На п'ятому заключному етапі здійснюється навчання нейронної мережі у програмному середовищі з використанням сформованої навчальної вибірки. Навчання полягає у мінімізації функції втрат, що відображає різницю між прогнозованими та еталонними значеннями, з подальшою фіксацією вагових коефіцієнтів. Отримана модель розглядається як параметризований оператор прийняття рішень, придатний для перенесення у середовище апаратної реалізації на ПЛІС, де вона використовується на етапі обробки даних без зміни структури.

Для забезпечення коректності навчання та подальшої верифікації моделі сформований датасет доцільно розділяти на навчальну, валідаційну та тестову підвибірки. Навчальна підвибірка використовується для налаштування параметрів мережі, валідаційна – для контролю узагальнювальної здатності моделі та запобігання перенавчанню, а тестова – для незалежної перевірки точності програмної і апаратної реалізацій. Така організація вибірки забезпечує методичну коректність оцінювання якості моделі й дозволяє відокремити ефект навчання від ефекту перевірки.

Після навчання:

- Вагові коефіцієнти експортуються.
- Передаються в систему для обробки (прогнозування).
- Реалізується апаратна реалізація нейронного шару у ПЛІС.

Така реалізація дозволить опрацьовувати вхідні вектори ознак за детермінований час.

2.6.3. Структура нейромережевого модуля прийняття рішень

Для реалізації способу було розроблено нейромережевий модуль, який забезпечує поетапну обробку даних та інтеграцію з апаратним середовищем ПЛІС. Модуль складається з таких блоків:

- Формування та нормалізація вхідних даних;
- Програмна реалізація навчальної мережі;
- Модуль експорту вагових коефіцієнтів;
- Інтерфейс для апаратної реалізації та обробки.

Далі наведено формальні описи та формули, що визначають структуру модуля та його функціонування у межах способу.

Запропонований нейронного модуль представляє собою багат шарову повнозв'язну нейронну мережу (Multi-Layer Perceptron, MLP), побудовану з використанням фреймворку PyTorch[121, 122]. Модуль призначений для обробки вхідного вектору фіксованої розмірності та формування одного нормалізованого вихідного значення.

Структурно нейромережевий модуль складається з послідовності лінійних шарів та нелінійних функцій активації, об'єднаних у єдиний обчислювальний ланцюг:

$$Input \rightarrow Linear1 \rightarrow ReLU1 \rightarrow Linear2 \rightarrow ReLU2 \rightarrow Linear3 \rightarrow Sigmoid \rightarrow Output$$

Мережа має три лінійні (повнозв'язні) шари, між якими застосовуються дві функції активації ReLU, а на виході сигмоїдна функція.

На вхід нейронного модуля подається вектор:

$$x \in R \quad (36)$$

де R – вектор розмірністю 11 вхідних ознак. Кожен елемент вектора відповідає окремій характеристиці вхідних даних.

Перший повнозв'язний шар виконує афінне перетворення вхідного вектора:

$$z_1 = W_1 * x + b_1 \quad (37)$$

де:

- W_1 – матриця ваг розмірністю 8×11 ,
- b_1 – вектор зміщень розмірністю 8,
- z_1 – вихід першого лінійного шару розмірністю 8.

Таким чином, на цьому етапі відбувається зменшення розмірності з 11 до 8.

До результату першого лінійного шару застосовується функція ReLU:

$$a_1 = ReLU(z_1) = \max(0, z_1) \quad (38)$$

Функція ReLU виконує нелінійне перетворення, обнуляючи всі від'ємні компоненти вектора z_1 , що:

- підвищує здатність мережі моделювати складні залежності,
- зменшує проблему зникнення градієнтів,
- прискорює процес навчання.

Другий лінійний шар приймає активований вектор a_1 та виконує наступне афінне перетворення:

$$z_2 = W_2 * a_1 + b_2 \quad (39)$$

де:

- W_2 – матриця ваг розмірністю 4×8 ,
- b_2 – вектор зміщень розмірністю 4,
- z_2 – вихід другого лінійного шару розмірністю 4.

На цьому етапі розмірність даних зменшується з 8 до 4, що дозволяє поступово концентрувати інформацію у більш компактному представленні.

До виходу другого лінійного шару знову застосовується функція ReLU:

$$a_2 = ReLU(z_2) = \max(0, z_2) \quad (40)$$

Цей етап виконує аналогічну роль, забезпечуючи додаткову нелінійність та стабільність навчання.

Третій лінійний шар формує скалярне значення:

$$z_3 = W_3 * a_2 + b_3 \quad (41)$$

де:

- W_3 – вектор ваг розмірністю 4,
- b_3 – зміщень,
- z_3 – вихід третього лінійного шару.

Це значення є нефільтрованим (логітом) виходу моделі.

На фінальному етапі до виходу застосовується сигмоїдна функція:

$$y = \sigma(z_3) = \frac{1}{1+e^{-z_3}} \quad (42)$$

Сигмоїда обмежує вихідне значення у діапазоні:

$$y \in (0,1) \quad (43)$$

Це робить модель придатною для:

- бінарної класифікації,
- оцінювання ймовірностей,
- задач, де вихід інтерпретується як нормалізований показник.

Використання даного способу можливе при виконанні умов (24–26)

У контексті даної роботи вихідне значення у інтерпретується не лише як абстрактний нормалізований показник, а як інтегральна оцінка відповідності поточного стану середовища встановленим критеріям якості. Таким чином, значення, близькі до нуля, відповідають умовам із суттєвими відхиленнями від допустимих меж, тоді як значення, близькі до одиниці, характеризують стани, що відповідають комфортному або нормативно допустимому режиму. Така інтерпретація робить вихід модуля придатним для безпосереднього використання у підсистемах моніторингу та формування керуючих впливів.

У підрозділі розроблено спосіб інтелектуальної обробки сенсорних даних, орієнтований на використання в системах IoT з апаратним прискоренням на базі ПЛІС. Спосіб базується на поетапному формуванні простору ознак і включає послідовні етапи стабілізації сенсорних вимірювань, фільтрації випадкових шумів, агрегування параметрів у часові вектори та формування цільової змінної для навчання нейронної мережі. Такий підхід дозволяє підвищити стійкість моделі до зашумленості даних та забезпечити узгодженість статистичних характеристик вхідної інформації.

Для реалізації способу запропоновано нейромережевий обчислювач на основі багатоварової повнозв'язної мережі типу MLP з 11 вхідними параметрами та сигмоїдною функцією активації на виході, що формує нормалізовану оцінку стану середовища у діапазоні від 0 до 1. Після програмного навчання мережі вагові коефіцієнти експортуються та використовуються в апаратній реалізації модуля обробки у ПЛІС, що забезпечує детермінований час обробки вхідних даних і робить запропонований спосіб придатним для використання в системах моніторингу та керування середовищами IoT в режимі реального часу.

ВИСНОВОК ДО РОЗДІЛУ 2

Розглянуто та обґрунтовано архітектуру і принципи функціонування адаптивної інтелектуальної системи IoT, орієнтованої на інтелектуальну обробку великих обсягів сенсорних потоків в умовах жорстких вимог до безпеки, надійності та передбачуваності. Запропоновано підхід, що ґрунтується на замкненому контурі зворотного зв'язку між станом інфраструктури, параметрами моніторингу та завантаженістю обчислювальних ресурсів, у межах якого результати спостереження трансформуються в адаптивні керувальні рішення щодо вибору алгоритмів обробки та зміни апаратної конфігурації граничних вузлів. Показано, що для більшості сценаріїв доцільним є розгортання системи в локальній інфраструктурі підприємства без прямого доступу до публічного Інтернету, що зменшує латентність, підвищує керованість критичних ресурсів і забезпечує необхідний рівень інформаційної безпеки.

Архітектурною основою запропонованого підходу визначено трирівневу модель інфраструктури IoT, яка забезпечує розподіл функцій збору, попередньої обробки та аналітики даних. На рівні пристроїв узагальнено вплив типових для виробництва факторів (шум, дрейф, завади, старіння сенсорів, режимні зміни процесів), що безпосередньо погіршують якість вимірювань і обґрунтовують необхідність застосування адаптивних механізмів очищення та контролю. На граничному рівні, який є ключовим у запропонованій концепції, сформовано структуру локальних доменів із граничними вузлами на базі процесорів ARM та ПЛІС і визначено їх функції: багаторівнева попередня фільтрація сенсорних даних, формування адаптивних тригерів, балансування навантаження та ініціація динамічної апаратної реконфігурації. Верхній рівень представлено як локальний сервер (приватна хмара) для агрегованої аналітики, історичного зберігання та координації доменів, який архітектурно враховується як частина екосистеми, хоча не є центральним об'єктом детального дослідження.

Розроблено та формалізовано удосконалений метод адаптивного налаштування порогів контролю в моніторингових системах IoT, який

відрізняється інтеграцією багаторівневої фільтрації сенсорних часових рядів із механізмами побудови трендів і автоматичного самоналаштування порогів. Обґрунтовано застосування послідовного поєднання медіанного фільтра, ковзного середнього та експоненціального згладжування як комплексу, що забезпечує одночасне усунення імпульсних завад, зменшення випадкових флуктуацій і виділення довготривалої спрямованості зміни параметра (тренду). Обґрунтовано, що використання трендів як базової лінії контролю дозволяє переходити від статичних порогів до «рухомих» адаптивних меж, знижуючи кількість хибних спрацювань за добових/сезонних коливань та зберігаючи інформативність сигналів у динамічних режимах виробництва.

Сформовано критерії оцінювання ефективності використання ресурсів граничних обчислювальних вузлів на основі комплексного показника, що інтегрує швидкодію обробки, енергоефективність і затримку, а також наведено правила нормалізації складових для отримання узгодженої оцінки в діапазоні $[0;1]$. Додатково обґрунтовано інтегральний показник поточного навантаження граничного вузла як зважену агрегацію параметрів CPU/RAM/диску, що забезпечує формалізовану основу для прийняття рішення щодо перерозподілу задач і ініціації реконфігурації в умовах обмежених ресурсів.

Запропоновано метод автоматичного прийняття рішень у безпекових сценаріях розподілених систем IoT, який реалізує політико-орієнтований цикл *«збір даних → аналіз і зіставлення з політиками → прийняття рішення → виконання керуючого впливу → інформування та аудит»* і мінімізує участь адміністратора на етапах первинного реагування. Метод включає взаємопов'язані алгоритми блокування з'єднань поза білим списком, виявлення аномальної процесної активності та ізоляції вузлів шляхом динамічної зміни мережевих параметрів (зокрема IP-адресації) з подальшою валідацією та коригуванням політик. Окремо описано спосіб керування динамічною реконфігурацією ПЛС в ОС Linux на основі наперед підготовленого набору прошивок (*rbf*) та формалізованої процедури безпечного перемикання конфігурацій із відключенням/відновленням каналів обміну та контролем працездатності.

Розроблено структуру нейромережевого модуля MLP, призначеного для формування нормалізованого вихідного показника в діапазоні $(0;1)$, та обґрунтовано концепцію поетапного формування даних для навчання і подальшого перенесення моделі в апаратне середовище. Показано, що розділення процесів вимірювання, стабілізації (апаратної фільтрації) та навчання забезпечує узгодженість статистичних властивостей вхідних ознак і підвищує стійкість моделі до шумів, а експорт ваг і реалізація обробки, прогнозування на ПЛС дозволяють отримати детермінований час обробки, критичний для сценаріїв наближених до реального часу.

У цілому запропоновано системний підхід до адаптивної інтелектуальної обробки даних у системах IoT, який поєднує трирівневу організацію, багаторівневу фільтрацію та тренд-орієнтоване самоналаштування порогів, формалізовані критерії ефективності ресурсів, автоматизовані безпекові сценарії й механізми динамічної реконфігурації ПЛС, а також нейромережевий модуль прийняття рішень. Сукупність запропонованих методів і алгоритмів формує цілісну технологічну екосистему, де параметри середовища та стан ресурсів перетворюються на керуючі дії, забезпечуючи підвищення надійності, зменшення хибних сповіщень, адаптацію до змін виробничих режимів і придатність до реалізації на реконфігуровних граничних вузлах.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ АДАПТИВНОЇ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ НА ГРАНИЧНОМУ РІВНІ ІНФРАСТРУКТУРИ ІoT

3.1. Методика реалізації адаптивно керованої інфраструктури ІoT з використанням моніторингової системи Zabbix

У даному розділі розроблено архітектуру адаптивної інтелектуальної системи обробки даних у розподілених системах Інтернету речей, орієнтовану на функціонування в умовах надходження великої кількості гетерогенних сенсорних вимірювань, багатопоточності обробки, підвищених вимог до швидкодії, надійності та масштабованості інфраструктури. Запропонована архітектура ґрунтується на системному підході до адаптивної інтелектуальної обробки даних на граничному рівні та передбачає інтеграцію взаємопов'язаних компонентів у межах єдиного керуючого контуру, який об'єднує багаторівневу фільтрацію та адаптивне налаштування порогів контролю сенсорних даних, інтелектуальну нейромережеву обробку, моніторинг стану апаратних і мережевих ресурсів, балансування обчислювального навантаження та автоматичне прийняття рішень у безпечових сценаріях. Така архітектура орієнтована на реалізацію на гетерогенних платформах SoC/ПЛІС із використанням засобів граничних обчислень і забезпечує детерміновану обробку потоків даних у реальному часі, підвищення достовірності та стабільності сенсорних вимірювань, ефективне використання обчислювальних ресурсів і підвищення стійкості функціонування інфраструктури ІoT.

Архітектура системи представлена на рис. 3.1 та відображає повний ланцюг взаємодії між фізичними пристроями, мережевою інфраструктурою, вузлами попередньої обробки даних і серверною частиною, що виконує функції зберігання, аналітики та адміністрування. Усі компоненти архітектури поділено на декілька логічних рівнів: фізичний рівень збору даних, рівень граничної обробки, мережевий рівень та серверний або хмарний рівень.

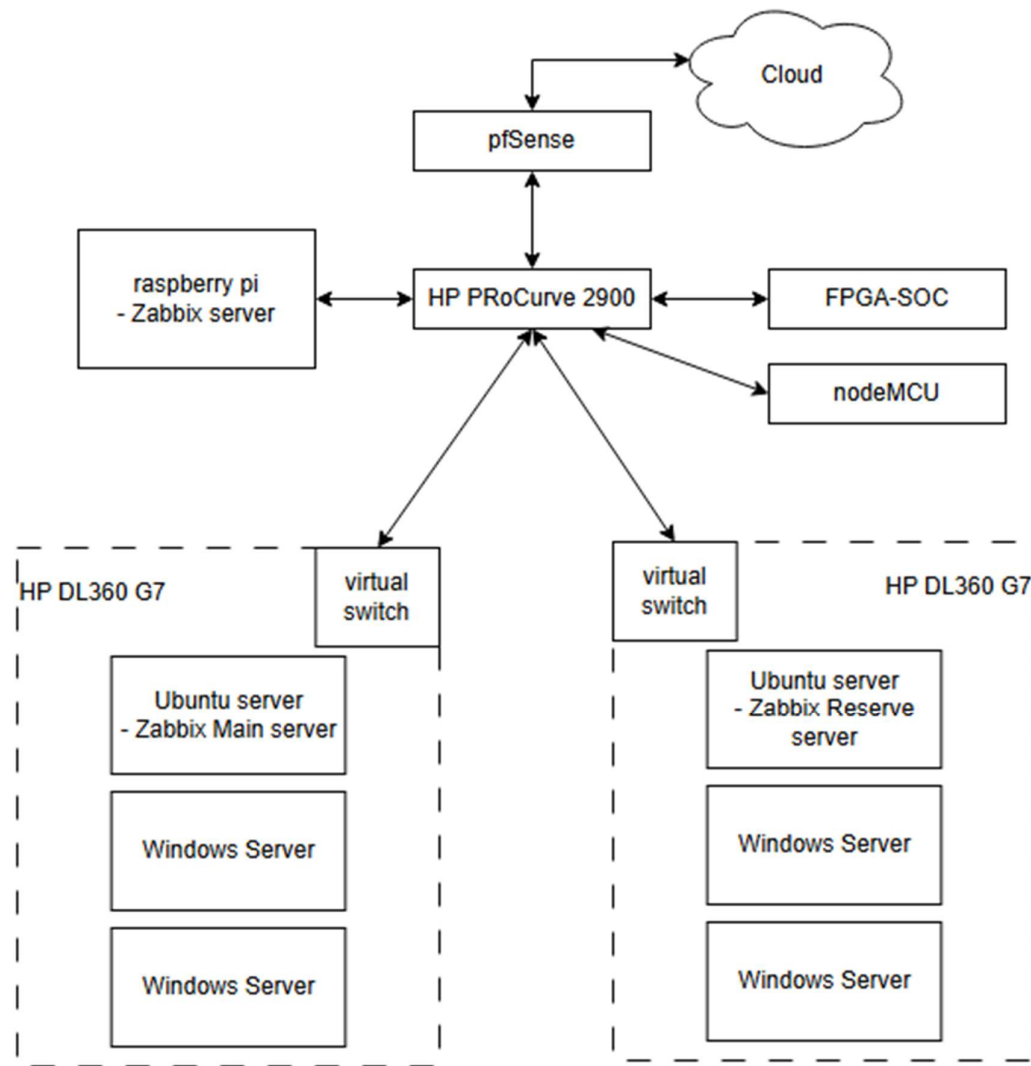


Рис 3.1. Архітектура адаптивно керованої IoT інфраструктури з використанням моніторингової системи Zabbix

Фізичну частину мережевої інфраструктури складають пристрої pfSense та керований комутатор HP ProCurve 2900, які забезпечують базові функції маршрутизації, комутації та контролю мережевого трафіку. Пристрій pfSense виконує роль центрального маршрутизатора та міжмережевого екрану, реалізуючи механізми NAT, фільтрації пакетів, сегментації мережі та контролю доступу між різними підмережами системи. Це дозволяє ізолювати пристрої IoT від серверної частини та підвищити загальний рівень безпеки.

Комутатор HP ProCurve 2900 відповідає за агрегацію мережевих підключень, підтримку VLAN та стабільну передачу даних між вузлами збору інформації,

граничними пристроями та серверною інфраструктурою. Використання керованого комутатора дає змогу гнучко масштабувати систему, розділяти трафік різних типів пристроїв та оптимізувати пропускну здатність мережі.

На рівні збору даних використовуються мікроконтролерні модулі NodeMCU, які безпосередньо підключені до сенсорів та виконавчих пристроїв. NodeMCU відповідають за первинне зчитування фізичних параметрів середовища, попередню підготовку даних (нормалізацію, фільтрацію шумів базового рівня) та передачу результатів вимірювань у мережу. Завдяки використанню бездротових або дротових інтерфейсів зв'язку дані передаються до вузлів граничної обробки з мінімальними затримками.

Функції граничної обробки покладено на пристрої Raspberry Pi, які виконують роль проміжних обчислювальних вузлів між сенсорними пристроями та серверною частиною системи. На цьому рівні реалізується більш складна обробка даних, зокрема агрегація показників з кількох джерел, застосування адаптивних алгоритмів фільтрації, виявлення аномалій та підготовка даних до подальшого аналізу.

Окрім обчислювальних функцій, Raspberry Pi використовується для тимчасового зберігання проміжних даних та взаємодії з сервером моніторингу Zabbix. Це дозволяє зменшити навантаження на центральний сервер і забезпечити стійкість системи у разі тимчасової втрати зв'язку з серверною інфраструктурою.

Для прискорення обчислень у задачах інтелектуальної обробки даних до Raspberry Pi підключається модуль ПЛІС, який використовується для апаратної реалізації обчислювально складних алгоритмів, таких як фільтрація поточкових даних або елементи нейромережевої обробки. Застосування ПЛІС дозволяє суттєво знизити затримки обробки та енергоспоживання у порівнянні з програмною реалізацією на CPU.

Центральну серверну частину системи реалізовано на фізичному сервері HP DL360 G7, який виконує роль платформи віртуалізації. На даному сервері розгорнуто головний сервер Zabbix, що забезпечує централізований моніторинг стану пристроїв IoT, мережевого обладнання та вузлів граничної обробки. Zabbix

здійснює збір телеметричних даних, їх візуалізацію, формування сповіщень та підтримку механізмів автоматичної реакції на критичні події.

Крім того, на сервері HP DL360 G7 функціонують віртуальні сервери під керуванням операційної системи Windows, які виконують допоміжні функції, зокрема зберігання даних, виконання прикладних сервісів, інтеграцію з іншими інформаційними системами. Такий підхід забезпечує гнучкість архітектури, спрощує масштабування та дозволяє розподіляти навантаження між різними сервісами.

3.2. Організація адаптивної обробки даних на граничному рівні інфраструктури IoT

У запропонованій архітектурі IoT процес збору даних є базовим функціональним компонентом, що забезпечує отримання актуальної інформації про стан фізичного середовища, мережевої інфраструктури та обчислювальних ресурсів системи. Збір даних реалізується у безперервному режимі та охоплює як сенсорні вимірювання, так і системні метрики, необхідні для подальшого аналізу, моніторингу та керування.

Архітектурно підсистема збору даних орієнтована на роботу в умовах великої кількості розподілених джерел інформації та високої частоти надходження телеметрії. З огляду на це, дані формуються у вигляді потоків, які передаються до граничних або централізованих вузлів обробки з урахуванням вимог до затримок, пропускнуої здатності мережі та надійності передачі.

Отримані дані використовуються не лише для пасивного спостереження за станом системи, але й як вхідна інформація для реалізації механізмів адаптивного керування та інтелектуальної обробки в межах інфраструктури IoT.

3.2.1 Збір даних з датчиків

Збір даних з датчиків є базовим етапом функціонування засобів моніторингу в інфраструктурі IoT та безпосередньо визначає достовірність і практичну цінність усієї подальшої обробки інформації. У межах запропонованої архітектури

сенсорний рівень спроектовано з урахуванням вимог до масштабованості, енергоефективності та можливості подальшого апаратного прискорення обчислювальних операцій на вищих рівнях системи.

Сенсорний рівень системи IoT формується сукупністю фізичних датчиків, які підключаються до мікроконтролерних платформ типу NodeMCU та інші, які виконують вимірювання параметрів навколишнього середовища й технічного стану обладнання. До основних контрольованих параметрів належать:

- температура;
- вологість;
- рівень освітленості;
- показники стабільності та коректності роботи пристроїв;
- додаткові технологічні та експлуатаційні параметри, що залежать від специфіки прикладної задачі.

NodeMCU виступає первинним вузлом збору даних, забезпечуючи опитування підключених сенсорів із заданою періодичністю, початкову обробку вимірювань та формування телеметричних повідомлень. Отримані дані передаються до наступного рівня архітектури для подальшої фільтрації, агрегації та інтеграції з системними метриками обчислювальних вузлів.

Такий підхід дозволяє забезпечити гнучке розширення кількості сенсорних вузлів без суттєвого ускладнення архітектури, а також створює основу для ефективної обробки великих обсягів сенсорних даних у розподіленій інфраструктурі IoT.

3.2.2 Збір даних з операційної системи вузлів інфраструктури IoT

Окрім збору фізичних та сенсорних параметрів, важливою складовою засобів моніторингу в інфраструктурі IoT є контроль стану обчислювальних вузлів, що функціонують під керуванням операційних систем Linux та Windows Server. У розподілених системах IoT порушення безпеки, деградація продуктивності або збої в обробці даних часто зумовлені не апаратними відмовами, а некоректною або несанкціонованою роботою програмного забезпечення та мережевих з'єднань.

У межах запропонованої архітектури збір даних з обчислювальних вузлів розглядається як інтегрований механізм моніторингу, що забезпечує отримання інформації про програмне середовище, мережеву активність і поточне навантаження ресурсів, та передусє прийняттю рішень щодо адаптації й балансування системи.

На кожному граничному вузлі або сервері під керуванням Linux чи Windows розгортається агент моніторингу, який функціонує у фоновому режимі та забезпечує безперервний збір даних про стан програмного середовища. Зокрема, агент здійснює контроль:

- активних процесів операційної системи;
- запущених служб;
- виконуваних бінарних файлів та їхніх ідентифікаторів;
- рівень використання ресурсів системи (CPU, оперативна пам'ять, системні диски).

Збір зазначених параметрів реалізується з використанням стандартних механізмів операційної системи та допоміжних скриптів, що забезпечує універсальність підходу та його незалежність від конкретної платформи. Отримана інформація нормалізується та приводиться до уніфікованого формату, що дозволяє виконувати подальший аналіз у єдиному інформаційному просторі інфраструктури IoT.

На рівні операційної системи агент моніторингу періодично отримує середні та пікові значення завантаження CPU, а також розподіл навантаження між окремими процесами, що дозволяє ідентифікувати першопричини перевантаження.

Особливістю запропонованої архітектури є використання ПЛІС як засобу апаратного прискорення обчислень. У зв'язку з цим система моніторингу включає збір даних про:

- активність обчислювальних модулів ПЛІС;
- затримки обробки даних у ПЛІС-модулях.

Крім того, враховуються мережеві параметри, що впливають на ефективність балансування:

- поточне навантаження на мережеві інтерфейси;
- затримки передавання даних;
- кількість активних з'єднань;
- рівень втрат пакетів.

Для підвищення ефективності аналізу зібрані показники агрегуються у вигляді інтегральних метрик навантаження, які використовуються як вхідні параметри для алгоритмів балансування та адаптації обробки даних.

Особлива увага приділяється виявленню процесів, які перевищують визначені порогові значення споживання ресурсів, оскільки саме такі процеси можуть свідчити про аномальну поведінку, помилки конфігурації або виконання небажаного програмного забезпечення.

Паралельно з аналізом програмного середовища здійснюється збір даних про мережеву активність обчислювальних вузлів. Моніторинг охоплює параметри, що характеризують мережеві взаємодії операційної системи, зокрема:

- IP-адреси віддалених вузлів, з якими встановлено активні з'єднання;
- напрямки передавання трафіку (вхідний та вихідний);
- кількість одночасних з'єднань та їхню тривалість.

Зібрані дані дозволяють сформувати актуальну модель мережевих взаємодій кожного граничного вузла або сервера. Такий підхід є особливо важливим у середовищах з динамічною адресацією або за умови взаємодії вузлів із великою кількістю зовнішніх сервісів.

Інформація про активні процеси, виконувані файли та IP-адреси використовується як вхідний набір даних для механізму перевірки відповідності білим спискам. У межах архітектури білі списки формуються на основі:

- дозволених процесів і служб;
- довірених IP-адрес або підмереж;
- дозволених типів мережевих з'єднань.

Передача зібраних даних до централізованої системи моніторингу дозволяє виконувати перевірку відповідності як у режимі реального часу, так і під час відкладеного аналізу. У разі виявлення невідповідностей формуються події або

тригери, які можуть ініціювати подальші керуючі дії, зокрема обмеження доступу або зміну параметрів обробки.

3.3. Апаратний багаторівневий конвеєр на ПЛІС для фільтрації даних

Для реалізації програмно апаратної платформи адаптивної обробки сенсорних даних розроблено апаратний багаторівневий конвеєр на ПЛІС для фільтрації даних та отримання трендів, необхідних для реалізації адаптивних порогів. Розроблений модуль фільтрації розглядається як ключовий функціональний компонент граничного рівня, призначений для попередньої інтелектуальної нейромережевої обробки сенсорних даних до їх передачі в систему моніторингу та прийняття рішень. Архітектурний концепт модуля базується на ідеї перенесення часово-критичних та ресурсоемних операцій обробки безпосередньо до джерела даних, що дозволяє мінімізувати навантаження на мережу та підвищити загальну стійкість системи. Структуру модуля зображена на рис. 3.2

Розроблений модуль фільтрації реалізовано у вигляді апаратно-програмного комплексу на базі платформи SoC/ПЛІС Cyclone V. Архітектура рішення передбачає поділ обчислювальних функцій між програмною частиною, що виконується під керуванням операційної системи Linux на вбудованому процесорі ARM, та апаратною частиною, реалізованою у вигляді спеціалізованого модуля фільтрації у програмованій логіці ПЛІС.

Програмна частина відповідає за:

- зчитування вхідних даних із файлу або потоку та перетворення даних у фіксований формат;
- передавання даних у апаратний фільтр через інтерфейс Avalon-MM;
- зчитування результатів фільтрації;
- формування вихідного файлу або потоку результатів.

Апаратна частина реалізує багатоступеневий фільтр, який складається з:

- медіанного фільтра;
- ковзного середнього;

– експоненційного згладжування.

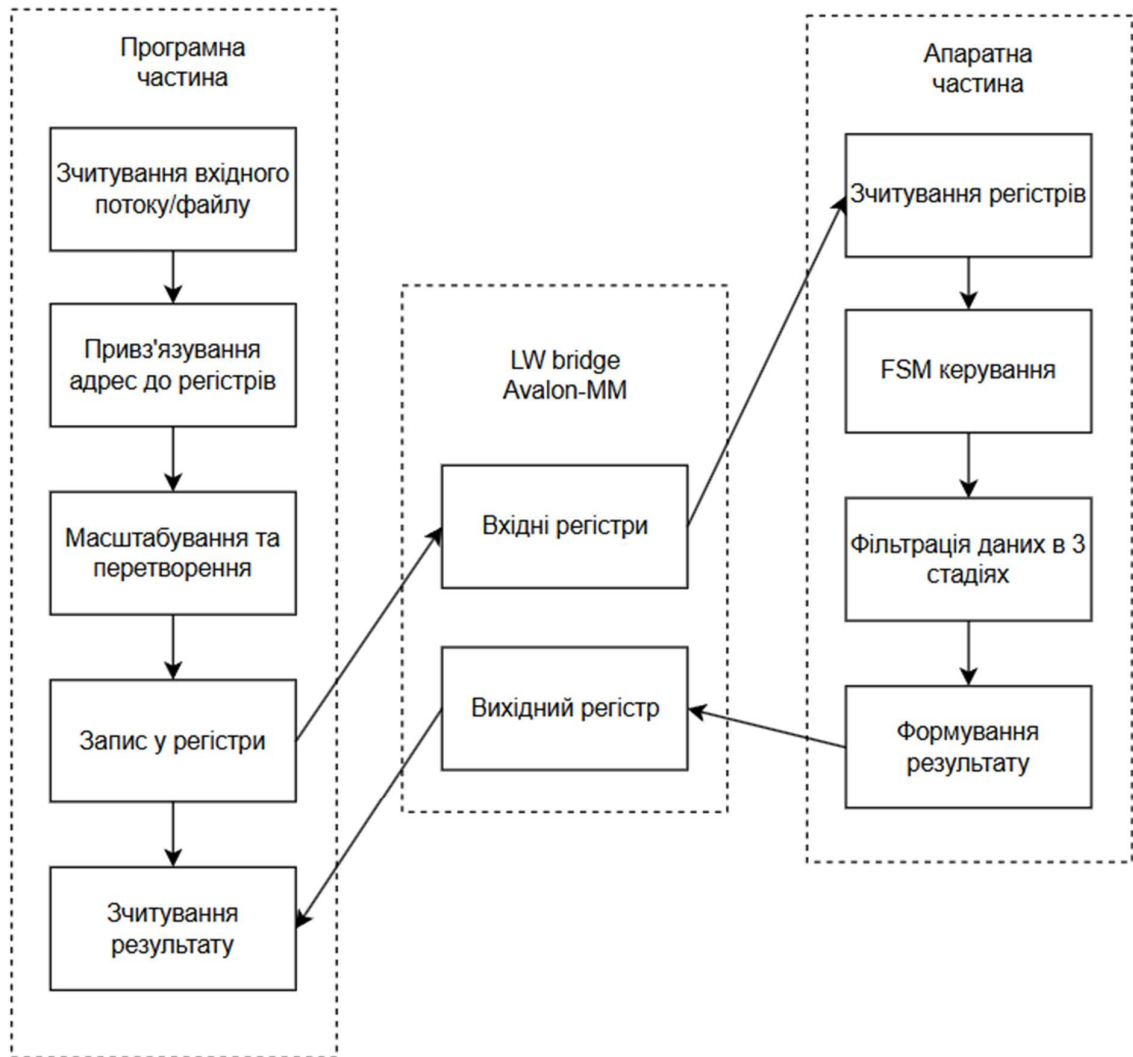


Рис. 3.2 Структурна схема модуля фільтрації

Взаємодія між процесором ARM та ПЛІС здійснюється через LW bridge, що забезпечує прямий доступ до регістрів апаратного модуля з користувацького простору Linux.

Для взаємодії з апаратним фільтром у Linux використовується механізм прямого доступу до фізичної пам'яті через файл `/dev/mem`. Базова адреса *lightweight*-мосту визначається як:

```
#define LW_BRIDGE_BASE 0xFF200000
#define LW_BRIDGE_SPAN 0x00010000
```

Апаратний модуль фільтрації розміщений у просторі адрес *lightweight*-мосту зі зсувом:

```
#define FILTER_OFFSET 0x00000100
```

Таким чином, програмна частина виконує відображення (*memory mapping*) відповідного діапазону фізичної пам'яті у віртуальний адресний простір процесу за допомогою системного виклику *mmap()*. Це дозволяє здійснювати доступ до регістрів ПЛІС як до звичайного масиву `uint32_t`.

Для узгодження виконання задач на апаратній і програмній складовій розроблено програмний модуль для операційної системи Linux (Linux-модуль).

З боку Linux-модуля апаратний фільтр розглядається як набір 32-бітних регістрів: регістр керування, регістр стану, вхідні дані, вихідні дані.

Запис у регістр `REG_IN` ініціює обробку нового семпла апаратним фільтром, а готовність результату визначається за допомогою біта `ready` у регістрі стану.

Алгоритм роботи Linux-додатка включає такі етапи:

1. Зчитування вхідного потоку даних.
2. Відкриття вихідного CSV-файлу для запису результатів.
3. Відображення *lightweight*-мосту у пам'ять користувачького простору.
4. Перетворення значень із формату з плаваючою комою у фіксований формат з масштабуванням (множення на 100).
5. Запис фіксованого значення у регістр.
6. Очікування встановлення біта готовності.
7. Зчитування результату та зворотне масштабування.
8. Запис відфільтрованих значень у CSV-файл.

Такий підхід дозволяє ізолювати складні обчислення в апаратному модулі та мінімізувати навантаження на центральний процесор.

Апаратний модуль `filter` реалізований як Avalon-MM slave-компонент, що працює синхронно з тактовим сигналом `clk` та має асинхронний сигнал скидання `reset`.

Основні складові модуля:

- інтерфейс Avalon-MM;

- реєстри керування та стану;
- три послідовні стадії фільтрації;
- скінченний автомат керування (FSM).

В апаратному модулі визначено такі реєстри:

Біт `ready_out` сигналізує програмній частині про завершення обробки поточного семпла, а біт `busy` використовується для індикації зайнятості модуля.

Перша стадія реалізує медіанний фільтр з вікном розміром `MED_WIN = 5`. Значення зберігаються у кільцевому буфері. Для обчислення медіани використовується комбінаційна логіка, яка:

- копіює вміст буфера у тимчасовий масив;
- виконує сортування методом «бульбашки»;
- обирає центральний елемент як результат медіани.

Такий підхід дозволяє ефективно пригнічувати імпульсні завади у сенсорних даних.

Друга стадія реалізує фільтр ковзного середнього з вікном `MA_WIN = 4`. Для зменшення апаратних витрат використовується акумулятор суми `ma_sum`, який оновлюється інкрементально.

Результат фільтрації обчислюється за допомогою арифметичного зсуву вправо, що еквівалентно діленню на розмір вікна.

Третя стадія реалізує експоненційний фільтр першого порядку. Коефіцієнти α та $1-\alpha$ задані у вигляді цілих чисел (`ALPHA_NUM`, `ONE_MINUS_NUM`) з подальшим діленням на `ALPHA_DEN`. Це дозволяє уникнути використання операцій з плаваючою комою та зменшити апаратну складність.

Керування проходженням даних через стадії фільтрації здійснюється за допомогою FSM з такими станами:

- `S_IDLE` – очікування нового семпла;
- `S_MED` – виконання медіанної фільтрації;
- `S_MA` – виконання ковзного середнього;
- `S_EXP` – виконання експоненційного згладжування;

- S_DONE – формування результату та сигналу готовності.

FSM забезпечує детерміновану послідовність обчислень і коректну синхронізацію з програмною частиною.

Запропонована апаратно-програмна реалізація модуля фільтрації дозволяє ефективно обробляти сенсорні дані в режимі близькому до реального часу. Перенесення обчислювально інтенсивних операцій на ПЛІС забезпечує зменшення затримок та навантаження на центральний процесор, а використання інтерфейсу Avalon-MM спрощує інтеграцію модуля в системах IoT.

3.4. Розроблення підсистеми балансування навантаження на граничному рівні

В умовах розподілених систем IoT із застосуванням граничних обчислень ефективність функціонування інфраструктури значною мірою визначається її здатністю адаптуватися до змін обчислювального та мережевого навантаження в режимі реального часу. Нерівномірні потоки сенсорних даних, поява короткочасних і тривалих пікових навантажень, а також деградація окремих граничних вузлів призводять до порушення часових обмежень обробки, зростання затримок, втрати пакетів і зниження якості сервісу. На відміну від хмарних середовищ, інфраструктури IoT функціонують в умовах обмежених обчислювальних, мережевих та енергетичних ресурсів, що робить проблему балансування навантаження критично важливою.

Запропонована підсистема балансування навантаження розглядається як інтегрований модуль загальної системи адаптивної обробки даних. Вона тісно пов'язана з підсистемами моніторингу, попередньої фільтрації сенсорних потоків та реконфігурації обчислювальних і мережевих ресурсів. Балансування реалізується не лише шляхом перерозподілу обчислювальних задач між граничними вузлами, але й через адаптацію параметрів обробки даних безпосередньо на крайовому рівні.

У межах архітектури балансування навантаження здійснюється безперервний збір експлуатаційних показників граничних вузлів на основі агентного підходу. Для цього використовується система моніторингу Zabbix, агенти якої розгортаються на кожному вузлі та забезпечують отримання актуальних даних про завантаження центрального процесора, використання оперативної пам'яті, інтенсивність і характеристики мережевого трафіку, затримки обробки даних. Додатково враховуються показники стабільності роботи вузлів у часі, що дозволяє фіксувати ознаки деградації ресурсів.

Зібрані метрики агрегуються та нормалізуються з метою формування уніфікованого вектора стану кожного граничного вузла. Оцінювання здійснюється на основі поєднання порогових та інтегральних критеріїв із урахуванням часової динаміки параметрів, що зменшує вплив короточасних пікових навантажень і запобігає прийняттю помилкових керуючих рішень. Такий підхід дозволяє визначати нормальний режим функціонування вузлів, виявляти перевантаження або деградацію окремих ресурсів, а також ідентифікувати вузли з резервом обчислювальної потужності.

Для кількісної оцінки стану граничних вузлів у запропонованій архітектурі використовується інтегральний зважений показник завантаженості. Він формується на основі декількох ключових нормалізованих метрик, зокрема відносного завантаження CPU, використання оперативної пам'яті, інтенсивності мережевого трафіку та затримки обробки або відповіді. Вагові коефіцієнти визначаються залежно від типу застосування та характеру навантаження: для потокової обробки сенсорних даних більша вага надається затримкам і завантаженню процесора, тоді як для задач агрегації - мережевим характеристикам. Нормалізація забезпечує приведення показників до єдиного діапазону та коректність інтегральної оцінки.

Центральним елементом архітектури є блок прийняття рішень, який формує керуючі впливи на основі поточного стану граничних вузлів, заданих політик, системних обмежень і пріоритетів. У межах реалізації підтримуються як детерміновані правила адаптації, так і можливість інтеграції інтелектуальних

методів, що створює основу для подальшого впровадження нейронних модулів прийняття рішень. Система Zabbix при цьому використовується не лише як засіб пасивного моніторингу, але й як джерело даних для керування - через тригери адаптивного типу, обчислювані елементи та зовнішні скрипти.

У разі перевищення порогових значень інтегрального показника завантаженості ініціюється процедура балансування. Алгоритм включає збір і нормалізацію метрик, обчислення зваженого показника для кожного вузла, порівняння станів між вузлами, виявлення перевантажених елементів та формування рішень щодо перерозподілу навантаження або переходу до альтернативних режимів обробки.

Реалізація керуючих рішень здійснюється через набір механізмів, що включають перенесення частини обчислень між граничними вузлами, перенаправлення потоків даних, коригування мережевих маршрутів і обмеження трафіку, активацію або деактивацію ПЛІС-модулів прискореної фільтрації, а також тимчасове зниження глибини та деталізації обробки даних. Агентний підхід забезпечує автономність виконання керуючих дій без прямого втручання в роботу операційної системи, що підвищує масштабованість і надійність архітектури.

Важливою особливістю запропонованого підходу є тісна інтеграція модуля балансування з підсистемою попередньої фільтрації сенсорних даних. Результати фільтрації використовуються для зменшення обсягу даних, що підлягають обробці в умовах перевантаження, та для адаптації обчислювального навантаження без суттєвої втрати інформативності. Таким чином, балансування реалізується не лише за рахунок перерозподілу ресурсів, але й шляхом керування інтенсивністю обробки даних на граничному рівні.

Запропонована архітектура та метод балансування навантаження забезпечують підвищення стійкості інфраструктури IoT до пікових і нерівномірних навантажень, зменшення ймовірності деградації окремих вузлів, ефективніше використання обчислювальних і мережевих ресурсів, а також формують основу для впровадження інтелектуальних методів адаптивного керування обчисленнями в розподілених системах IoT.

3.5. Розроблення методу автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT

3.5.1 Призначення та роль модуля автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT

Модуль перебудови мережі призначений для автоматизованого керування топологією та правилами доступу в розподіленій інфраструктурі IoT з метою підвищення надійності сервісу при змінному навантаженні, локалізації підозрілої активності, обмеження доступу до критичних вузлів на основі політик довіри.

На відміну від модуля балансування, який оптимізує розподіл потоку запитів/даних, модуль перебудови мережі виконує керуючі дії на мережевому рівні: змінює правила маршрутизації та фільтрації, оновлює списки дозволених підключень, ініціює блокування/ізоляцію вузлів або перенесення доступу на резервні канали.

Таким чином, модуль перебудови є «виконавчим» компонентом, який реалізує політики безпеки та стійкості на основі даних моніторингу (Zabbix) і зовнішньої керованої інфраструктури (pfSense з використанням API).

Вхідні дані:

- телеметрія стану вузлів (CPU/RAM, доступність сервісів, події тригерів);
- дані мережевих підключень: активні сесії, джерела IP;
- дані про ПЗ: список процесів, показники споживання ресурсів;

конфігураційні політики:

- список дозволених IP-адрес і підмереж (глобальні та на рівні вузла/сервісу);
- білий список ПЗ (глобальний/ролевий/вузловий);
- винятки (технічне обслуговування, тимчасові дозволи, “допустимі аномалії”);

правила реагування: пороги, пріоритети подій, сценарії перебудови.

Вихідні керуючі дії:

- блокування IP/підмереж на периметрі або локально;
- оновлення правил мережевого екрану (pfSense), NAT, політик маршрутизації, таблиць псевдонімів;
- переведення вузла в ізольований сегмент (карантин/VLAN);
- перемикання маршруту на резервний шлюз або зміну дозволених напрямків трафіку;

3.5.2 Логічні компоненти архітектури модуля

Архітектуру модуля доцільно описувати як набір взаємодіючих підсистем (логічних шарів), кожна з яких має чіткі функції та інтерфейси.

Підсистема збору фактів (*Data Acquisition Layer*) організована з використанням агента Zabbix.

Підсистема політик довіри (*Policy & Trust Layer*)

Список дозволених IP адрес:

- глобальні дозволи (внутрішні сегменти, шлюзи, DNS/NTP, бекенд-сервіси);
- ролеві дозволи (для типу вузла: шлюз, граничний вузол, мережеве сховище);
- вузлові дозволи (конкретні пари “вузол та дозволені джерела”);

Білий список програм – перелік дозволених процесів/служб.

Правила винятків:

- технічне обслуговування;
- тимчасова довіра (time-boxed allow);
- критичні сервіси, що мають бути доступні за будь-яких умов.

Політики зберігаються у вигляді файлу налаштувань з датою та часом до внесення змін, щоб модуль перебудови завжди знав, за якою версією приймалося рішення, і міг скасувати зміни політик.

Підсистема кореляції та оцінки ризику (*Correlation & Risk Scoring*)

Це «аналітичне ядро» модуля, що перетворює необроблені дані в оцінку ситуації.

Типові правила кореляції:

- мережеве відхилення: IP джерела відсутній в списку дозволених для даного вузла/сервісу;
- відхилення ПЗ: процес не знаходиться у білому списку або процес у білому списку, але шлях/параметри не відповідають дозволеним;
- аномалія ресурсу: процес $>$ порога CPU/RAM і не входить у білий список “дозволених важких” процесів;
- комбінована подія: одночасно “підозрілий IP” + “новий/невідомий процес” + “зростання мережевого трафіку”.

Результат кореляції - ризиковий індекс R , що визначає клас реакції:

- $R0$: норма;
- $R1$: підозра (логування, збір доказів, підвищений моніторинг);
- $R2$: загроза (точкове блокування IP/портів, rate-limit);
- $R3$: критично (карантин вузла/сегмента, блокування підмереж, аварійна перебудова).

Підсистема прийняття рішень (Decision & Orchestration)

Виконує вибір сценарію перебудови на основі:

- ризикового індексу R ;
- пріоритету вузла (критичність сервісу);
- поточного стану навантаження (щоб не “вбити” систему перебудовою);
- історії подій (уникнення циклів блокування/розблокування).

Підсистема виконання (Enforcement Layer)

Реалізує дії через керовані інтерфейси:

- pfSense API v2 (внесення правил, псевдоніми, оновлення таблиць блокування);
- локальні скрипти на вузлах (зупинка процесів, мережевий екран на хості, зміна конфігів);
- механізми сегментації: VLAN/окремі правила маршрутизації.

Додаткова функція: підтвердження застосування - перевірка, що правило реально активне (ping/health-check, запит стану через API, повторний збір метрик).

Журналювання та аудит (Logging & Audit)

Містить:

- подію (що сталося);
- політику (які білі списки застосовані, версії);
- рішення (чому саме цей сценарій);
- дію (які правила змінено, де, коли);
- результат (успіх/помилка, відкат, повтор).

3.5.3 Механізм списків дозовлених IP адрес: модель та правила використання

Структура білого списку організована багаторівневою переліком дозовлених IP:

Глобальний список дозовлених IP адрес (інфраструктурні сервіси):

- шлюзи/маршрутизатори;
- DNS/NTP;
- сервери моніторингу (Zabbix);
- репозиторії оновлень (якщо дозволяються).

Ролевий список дозовлених IP адрес:

- для вузлів сенсорного рівня дозволяються лише підключення до граничних вузлів;
- для граничних вузлів дозволяються підключення до storage/аналітики та керування;
- для керуючих вузлів - обмежені адмінські доступи з mgmt-сегмента.

Вузловий список дозовлених IP адрес:

- точні дозволи для конкретних пристроїв або сервісів (найбільш суворий рівень).

Для кожного мережевого факту (сесії/спроби) виконується перевірка:

- чи входить IP джерела до дозовлених для цього вузла;
- чи відповідає підключення дозовленому напрямку (inbound/outbound);

- чи дозволений порт/сервіс.

Результатом є мітка:

- *Allowed* (в списку дозволених IP адрес);
- *Unknown* (немає у списках, але без явних ознак атаки);
- *Denied/Suspicious* (явне порушення політики або аномальна активність).

3.5.4 Механізм білого списку ПЗ: модель та правила використання

Білий список ПЗ виконує дві функції:

1. Класифікація дозволених процесів (що має право працювати на вузлі).
2. Контроль «ресурсно небезпечних» процесів: якщо процес споживає >40% CPU, то він має бути:
 - або у білому списку як дозволений,
 - або підлягати реакції (обмеження/зупинка/ізоляція вузла залежно від контексту).

Для підвищення точності білий список може включати:

- ім'я процесу;
- шлях до виконуваного файлу;
- параметри запуску;
- користувача/контекст виконання;
- (за можливості) хеш/підпис/версію.

Чим більше ознак - тим менше хибних спрацювань і тим надійніша детекція підміни.

3.5.5 Типові сценарії перебудови мережі

Сценарій S1: Точкове блокування IP (R2)

Умова:

- IP джерела відсутній у списку дозволених адрес;
- повторні спроби/аномальна частота;
- відсутні ознаки компрометації вузла.

Дія:

- додати IP до таблиці блокування (pfSense alias/table);
- застосувати правило на периметрі для конкретного сервісу/вузла;
- увімкнути посилене логування на вузлі-цілі.

Очікуваний ефект:

- зниження небажаних підключень без впливу на легітимний трафік.

Сценарій S2: Реконфігурація доступу до вузла (R2→R3)

Умова:

- підозрілий IP та збільшення трафіку або помилки сервісу;
- підвищення ризику компрометації.

Дія:

- обмежити доступ до вузла лише з mgmt-сегмента та необхідних граничних вузлів;
- закрити всі непотрібні порти;
- встановити тимчасовий “мінімальний профіль доступу” (hardening policy).

Сценарій S3: Карантин вузла (R3)

Умова:

- невідомий процес відсутній у білому списку;
- процес споживає ресурси (CPU/RAM);
- паралельно фіксуються підозрілі мережеві з’єднання.

Дія:

- перевести вузол у карантинну зону (обмежений VLAN/правила);
- дозволити лише зв’язок із Zabbix і forensic-сервісами;
- ініціювати локальні дії: завершення процесу, зупинка сервісів, збір артефактів.

Очікуваний ефект:

- локалізація загрози та запобігання поширенню.

Сценарій S4: Перемикання маршруту/шляху даних (стійкість, не лише безпека)

Умова:

- перевантаження вузла або деградація каналу;
- необхідність зберегти сервісність.

Дія:

- змінити правил маршрутизації або маршруту на резервний шлюз;
- перенаправити потоки на альтернативний граничний вузол;
- тимчасово знизити «вартість» вузла в балансуванні (взаємодія з твоїм зваженим показником).

3.5.6 Взаємодія з Zabbix і pfSense: логіка керування

У запропонованій реалізації Zabbix формує події (*triggers*), які є ініціаторами перебудови. Для кожної події модуль перебудови виконує:

- ідентифікацію вузла та контексту;
- отримання актуальних метрик/списків підключень/процесів;
- порівняння з списками дозволених IP адрес або білим списком;
- розрахунок рівня ризику R ;
- вибір сценарію S ;
- застосування через pfSense API та/або локальні скрипти;
- перевірку результату (пост-контроль);
- аудит і, за потреби, скасування змін.

Критична властивість: модуль не повинен бути «сліпим виконавцем» - він має перевіряти наслідок (наприклад, чи справді підозрілий IP перестав досягати сервісу; чи відновилась доступність; чи не заблоковано легітимний трафік).

3.5.7 Механізми захисту від помилок і «флапінгу» правил

Щоб уникнути нестабільності (часті блокування/розблокування), вводяться:

- гістерезис (різні пороги для входу в стан загрози та виходу);
- таймаути реакцій;
- ліміти змін конфігурації за інтервал часу;

- пріоритети (залежно від ролі вузла);
- режими: *normal / alert / incident / maintenance*.

3.6. Модуль інтелектуальної нейромережевої обробки для прийняття рішень на основі оброблених сенсорних сигналів

Нейромережевий обчислювач є спеціалізованим апаратно-програмним компонентом граничного рівня, призначеним для оперативної інтерпретації багатовимірних сенсорних та контекстних даних і формування інтегрального показника стану, який використовується системою моніторингу як компактний критерій для подальших керуючих дій. На відміну від традиційних підходів, коли попередня обробка й оцінювання виконуються на CPU або на зовнішньому сервері, запропонована архітектура переносить обчислення нейромережевої моделі безпосередньо в програмовану логіку ПЛІС. Це забезпечує скорочення затримок, підвищення детермінованості часу обробки та зменшення навантаження на центральний процесор і мережеві канали.

Модуль реалізує етап прогнозування (прямого поширення) багатошарової нейромережі малої розмірності, орієнтованої на роботу з фіксованим набором вхідних ознак. Результатом роботи є значення в інтервалі, інтерпретованому як рівень “comfort” (або інший узагальнений індикатор), який може безпосередньо застосовуватись у тригерній логіці моніторингу.

Архітектура нейронного модуля базується на принципі розділення функцій між трьома логічними підсистемами:

1. **Підсистема обміну даними** (інтерфейс пам’яті/регістрова модель), яка забезпечує взаємодію між HPS (Linux) та апаратним ядром через стандартизований інтерфейс внутрішньої шини (Avalon-MM). Вона реалізує набір керуючих і статусних регістрів, а також регістрів для послідовного введення ознак і зчитування результату.
2. **Обчислювальне ядро нейромережі**, що виконує послідовність операцій над вхідним вектором: обчислення проміжних представлень у прихованих шарах і формування вихідного логіту. Реалізація побудована як керований

автомат станів, що проходить етапи обробки шар за шаром, гарантуючи повторюваність та контрольовану латентність.

3. **Блок нелінійної активації виходу**, який виконує перетворення логіту в нормоване значення ймовірнісного типу. З огляду на апаратні обмеження та прагнення мінімізувати ресурсоемні операції, нелінійність реалізована табличним методом (LUT), що є типовим підходом у вбудованих і ПЛІС рішеннях.

Загалом нейромережевий обчислювач функціонує як завершений периферійний ІР-блок у системі-на-кристалі. Центральний процесор виступає «керівником сеансу» – готує дані, передає їх на ПЛІС та отримує результат для подальшої логіки моніторингу або керування.

Для інтеграції з операційною системою Linux нейромережевий обчислювач представлений у вигляді набору регістрів, доступних через пам'яттєво відображений інтерфейс. Такий підхід дозволяє працювати з модулем стандартними засобами користувацького простору (через */dev/mem* та *mmap*) без обов'язкового створення специфічного драйвера на рівні ядра, що важливо на етапі експериментальної апробації та швидких ітерацій.

Регістрова модель включає:

- керуючий регістр, призначений для функцій керування (ініціалізація, очищення станів, синхронізація);
- статусний регістр, що відображає стан виконання обчислень, готовність результату, а також прогрес заповнення вхідного буфера;
- вхідний регістр, через який здійснюється послідовне завантаження компонентів вхідного вектора ознак;
- вихідний регістр, що містить готовий результат обробки у нормованому форматі.

Протокол обміну побудований на ідеї пакетного введення: програмна частина послідовно подає всі елементи вхідного вектора, після чого апаратна частина автоматично запускає обчислення. Завершення обробки сигналізується статусним прапорцем готовності, а результат доступний для зчитування з

вихідного реєстра. Така схема спрощує інтеграцію з системою моніторингу: алгоритм на HPS зводиться до «записати ознаки - дочекатися готовності - зчитати індикатор».

Оскільки апаратна реалізація орієнтована на виконання обчислень з мінімальним використанням ресурсів, у модулі застосовується подання чисел із фіксованою комою. Вхідні ознаки, вагові коефіцієнти та зміщення нейронів представлені в одному узгодженому форматі квантування. З позицій системної архітектури це означає, що на програмному боці обов'язковим етапом є приведення значень сенсорів та контекстних параметрів до нормованого діапазону і подальше квантування до цільового формату.

Такий підхід має дві ключові переваги. По-перше, забезпечується детермінований час обробки і відсутність залежності від апаратної підтримки плаваючої коми. По-друге, спрощується перенос моделі з середовища навчання (де домінує float-арифметика) у середовище виконання на ПЛІС. Водночас квантування накладає вимоги на процедуру підготовки моделі: експорт ваг і масштабування ознак повинні бути узгоджені, а точність - перевірена на тестовому наборі, що є складовою методики верифікації.

Нейромережеве ядро реалізує багатошарову структуру з фіксованою кількістю входів і нейронів у прихованих шарах. Обраний розмір мережі є компромісом між виразною здатністю моделі та апаратними витратами. У межах граничних застосувань, де важливими є швидкодія та сталість виконання, малі MLP-моделі часто демонструють достатню якість для задач оцінювання стану або класифікації типових режимів середовища.

Обчислення організовані як послідовність етапів, що керується скінченним автоматом:

- етап обчислення першого шару;
- етап використання функції активації;
- етап обчислення другого шару;
- етап використання функції активації;
- етап обчислення третього шару;

- етап формування вихідного логіту;
- етап фіксації результату та переведення системи у стан готовності.

Для шарів використано просту і апаратно ефективну нелінійність, що обрізає від’ємні значення до нуля. Ця функція активації оптимально підходить для апаратної реалізації з фіксованою комою, оскільки не потребує складних обчислень, а її вплив на стабільність моделі є передбачуваним.

Важливою архітектурною особливістю є використання ширших внутрішніх акумуляторів під час проміжних сумувань. Це рішення зменшує ризик втрат точності та переповнення при складанні внесків від багатьох входів і дозволяє надійно підтримувати заданий формат квантування без деградації результату на типових робочих даних.

Завершальний етап обробки потребує перетворення вихідного значення в нормовану оцінку, придатну для інтерпретації як індикатор у межах $[0; 1]$. У реалізаціях на ПЛІС обчислення експоненційних функцій є ресурсомістким, тому застосовано табличний метод попередньо підготовлений масив значень нелінійності, який зчитується за індексом. Індекс формується шляхом приведення вихідного логіту до обмеженого діапазону та лінійного відображення на адресний простір таблиці.

Табличний підхід забезпечує:

- стабільність та повторюваність результату;
- малу латентність (операція доступу до пам’яті LUT);
- контрольовану точність, що визначається розміром таблиці і схемою квантування.

Програмна частина в середовищі Linux виконує функції підготовки даних та організації обміну з ПЛІС. Взаємодія реалізована як користувацький застосунок, який:

- читає вхідні ознаки з файлу або отримує їх від підсистеми збору сенсорних даних;
- парсить та нормалізує значення, після чого виконує квантування до формату модуля;

- послідовно записує компоненти вхідного вектора у реєстр введення;
- контролює завершення обробки за статусним прапорцем готовності;
- зчитує результат з вихідного реєстра і перетворює його у числовий індикатор прикладного рівня;
- зберігає результати у вигляді журналу (наприклад, CSV), що забезпечує можливість подальшого аналізу, калібрування та порівняння з програмною еталонною моделлю.

Такий режим корисний не лише для експлуатації, але й як інструмент верифікації: він дозволяє масово проганяти тестові вектори та оцінювати стабільність і точність апаратної реалізації.

Архітектурно важливим є узгодження трактування керуючих сигналів між програмною та апаратною частинами. У дослідних прототипах типово закладають можливість явного скидання стану вводу, примусового старту та очищення прапорця готовності. Це спрощує інтеграцію у складні програмні сценарії, де можливі повторні запуски, обробка помилок, переривання або асинхронні джерела даних. Водночас практична реалізація може опиратися на автоматичний запуск після отримання повного вектора ознак, що зменшує кількість керуючих операцій і робить протокол компактнішим.

ВИСНОВОК ДО РОЗДІЛУ 3

У межах проведеного дослідження розроблено комплексну багаторівневу архітектуру інтелектуальної системи обробки даних для Інтернету речей, яка поєднує принципи граничних обчислень, апаратного прискорення, адаптивної фільтрації, інтегрованого моніторингу, динамічного балансування навантаження та автоматизованої перебудови мережевої інфраструктури. Запропоноване архітектурне рішення є системно узгодженою моделлю побудови розподіленої IoT-інфраструктури, орієнтованої на роботу в умовах обмежених ресурсів, змінного навантаження, підвищених вимог до затримок та необхідності забезпечення інформаційної безпеки.

Архітектура побудована за ієрархічним принципом із чітким розподілом функцій між фізичним, граничними, мережевим і серверним рівнями. Такий підхід забезпечує:

- локалізацію часово-критичних обчислень на периферії мережі;
- зменшення обсягу переданих даних завдяки попередній фільтрації;
- централізований контроль і координацію через серверну інфраструктуру;
- ізоляцію та сегментацію мережі для підвищення безпеки.

Фізичний рівень, представлений мережевими пристроями pfSense та HP ProCurve, створює кероване й сегментоване середовище передачі даних. Граничний рівень (NodeMCU, Raspberry Pi, ПЛІС) реалізує розподілену обробку, тоді як серверний рівень (HP DL360 G7 із віртуалізацією та Zabbix) забезпечує централізований моніторинг, агрегацію метрик і координацію керуючих впливів.

У результаті формується замкнений контур *«збір даних → аналіз і зіставлення з політиками → прийняття рішення → виконання керуючого впливу → інформування та аудит»* що є характерною ознакою кіберфізичних систем нового покоління.

Одним із концептуально важливих результатів є реалізація принципу перенесення обчислювально складних операцій максимально близько до джерела даних. Це проявляється у трьох взаємодоповнюючих компонентах:

1. Апаратний модуль багатоступеневої фільтрації на ПЛІС.
2. Нейромережевий обчислювач прогнозування у програмованій логіці.
3. Локальна агрегація й оцінювання стану перед передачею в центральну систему.

Апаратна реалізація фільтрації (медіанний фільтр, ковзне середнє, експоненційне згладжування) дозволяє:

- мінімізувати затримки обробки;
- зменшити навантаження на CPU;
- знизити енергоспоживання порівняно з програмною реалізацією;
- забезпечити детермінованість часу виконання.

Розділення функцій між ARM-процесором та ПЛІС логікою на базі Avalon-MM та LW bridge формує ефективну гетерогенну обчислювальну платформу. Такий підхід відповідає сучасним тенденціям побудови вбудованих систем, де апаратна й програмна частини працюють у тісній кооперації.

Розроблена архітектура моніторингу виходить за межі традиційного спостереження за ресурсами. Вона об'єднує:

- сенсорні дані фізичного середовища;
- метрики обчислювальних вузлів (CPU, RAM, диски);
- параметри ПЛІС-прискорювачів;
- мережеву активність (IP, сесії, затримки, втрати);
- дані про програмне середовище (процеси, служби, бінарні файли).

Уніфікація цих даних у вигляді нормалізованих векторів стану створює інформаційну основу для:

- адаптивного балансування навантаження;
- виявлення аномалій;
- автоматизованої перебудови мережі;
- формування інтегральних індикаторів.

Таким чином, моніторинг у запропонованій архітектурі є не пасивним інструментом спостереження, а активним елементом керуючого контуру.

Важливим науково-практичним результатом є введення інтегрального зваженого показника завантаженості граничних вузлів. Його використання дозволяє:

- формалізувати стан вузла в багатовимірному просторі метрик;
- враховувати специфіку застосування через систему ваг;
- зменшувати вплив короткочасних пікових навантажень;
- приймати рішення на основі агрегованої, а не фрагментарної інформації.

Балансування реалізується не лише шляхом перерозподілу задач, а й через адаптацію глибини обробки (наприклад, регулювання інтенсивності фільтрації або активації ПЛІС). Це свідчить про перехід від статичної моделі керування до адаптивної, контекстно-залежної моделі.

Модуль перебудови мережі є логічним продовженням концепції адаптивності, але вже на рівні мережевої топології та політик доступу. Його ключові особливості:

- багаторівнева система білих списків та списків дозволених IP адрес (глобальна, ролева, вузлова);
- кореляція подій і формування ризикового індексу R ;
- класифікація ситуацій за рівнями критичності;
- сценарний підхід до реакції ($S1-S4$);
- інтеграція з pfSense через API;
- механізми гістерезису й антифлапінгу.

Таким чином, архітектура підтримує не лише балансування продуктивності, але й активну кіберстійкість. Мережа перестає бути статичною інфраструктурою та набуває властивостей самокерованої системи, здатної до локалізації загроз і динамічної сегментації.

Інтеграція нейронного модуля прогнозування у ПЛІС формує якісно новий рівень узагальнення інформації. Модуль:

- обробляє багатовимірний вектор ознак;
- формує нормований інтегральний показник;

- працює у фіксованій точці з контрольованою латентністю;
- забезпечує детерміновану продуктивність.

Застосування LUT-методу для нелінійності та використання арифметики з фіксованою комою, демонструє раціональний компроміс між точністю й апаратною складністю. Архітектурно це дозволяє:

- зменшити залежність від серверних ресурсів;
- виконувати оцінювання в режимі близькому до реального часу;
- інтегрувати результат у тригерну логіку Zabbix і систему керування.

Нейромережевий обчислювач фактично виступає компактним “когнітивним сенсором”, який перетворює складний багатофакторний стан у єдиний числовий критерій.

Запропонована архітектура має такі системні переваги:

- Масштабованість - підтримка розширення сенсорного рівня та граничних вузлів без суттєвої перебудови логіки.
- Модульність - кожен компонент (фільтрація, балансування, перебудова, нейромодуль) може розвиватися незалежно.
- Стійкість до відмов - локальна обробка та сегментація зменшують вплив відмов центрального вузла.
- Безпека за принципом zero-trust - використання білих списків і сценарного реагування.
- Детермінованість - апаратна реалізація критичних алгоритмів гарантує передбачуваний час виконання.
- Верифікованість - можливість порівняння апаратної та програмної реалізацій моделей.

РОЗДІЛ 4

ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА МЕТОДУ ТА ЗАСОБІВ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДАНИХ

4.1. Організація та методика проведення експериментів

Необхідність проведення експериментальних досліджень обумовлена результатами аналітичного огляду та розроблених моделей, представлених у розділах 1–3. У першому розділі було визначено особливості архітектури мереж Інтернету речей, їх багаторівневу структуру (сенсорний, мережевий, прикладний / граничний та хмарний рівні) та ключові проблеми обробки поточкових даних, зокрема гетерогенність джерел, ресурсні обмеження та потребу в обробці в режимі реального часу. Було обґрунтовано доцільність переходу до гібридної архітектури з використанням SoC/ПЛІС як засобу забезпечення детермінованості та зменшення навантаження на центральні вузли.

У другому розділі сформовано математичні моделі та методи адаптивної обробки даних, зокрема метод багаторівневої фільтрації на основі поєднання медіанного фільтра, ковзного середнього та експоненціального згладжування. Також було розроблено метод оцінювання стану обчислювальних ресурсів граничних вузлів і модель прийняття рішень щодо зміни топології системи, а структура нейронного модуля визначена як компонент інтегральної оцінки багатофакторного стану середовища.

У третьому розділі реалізовано архітектуру інтелектуальної обробки даних, що включає:

- модуль моніторингу сенсорних і системних параметрів;
- апаратний модуль багатоступеневої фільтрації на ПЛІС;
- модуль балансування навантаження;
- модуль перебудови мережі з використанням білих списків;
- нейромережевий обчислювач прогнозування у програмованій логіці.

Таким чином, до початку експериментальної частини була сформована цілісна гібридна архітектура IoT з замкненим контуром керування «збір даних →

аналіз і зіставлення з політиками → прийняття рішення → виконання керуючого впливу → інформування та аудит».

Мета експериментів

Метою експериментальних досліджень є перевірка ефективності запропонованої архітектури та методів з точки зору:

Зменшення впливу шумів і аномалій у сенсорних даних завдяки багаторівневій фільтрації.

- Скорочення навантаження на CPU шляхом перенесення обробки на ПЛІС.
- Підвищення детермінованості часу обробки.
- Зменшення кількості хибних тригерів у системі моніторингу.
- Підвищення швидкості реагування системи на зміну стану середовища.
- Порівняння апаратної та програмної реалізацій моделей (верифікація).

Предметом дослідження виступають:

- метод багаторівневої апаратної фільтрації;
- метод адаптивного налаштування тригерів;
- архітектура балансування навантаження;
- механізми автоматизованої перебудови мережі;
- нейромережевий обчислювач інтегральної оцінки стану.

Експериментальні дослідження організовані у вигляді послідовної перевірки окремих компонентів системи та їх інтегрованої роботи:

- Перевірка коректності апаратної фільтрації
- Порівняння вихідних та відфільтрованих сигналів, аналіз усунення імпульсних шумів та коливань.
- Оцінювання продуктивності ПЛІС-підсистеми
- Вимірювання часу обробки масиву даних, завантаження CPU, енергоспоживання та затримок.
- Перевірка адаптивної тригерної логіки в Zabbix

- Аналіз кількості хибних спрацювань до і після впровадження багаторівневої фільтрації.
- Дослідження роботи нейронного модуля
- Оцінювання стабільності інтегрального показника та його впливу на прийняття рішень.
- Аналіз сценаріїв перебудови мережі
- Перевірка реакції системи на перевантаження вузлів або виявлення аномальної активності.

Для кількісного аналізу використовуються такі показники:

- середній час обробки одного вимірювання;
- пропускна здатність системи;
- відсоток зменшення хибних спрацювань;
- завантаження CPU до та після впровадження ПЛІС-прискорення;
- енергоспоживання;
- стабільність інтегрального індикатора стану.

Проведення експериментів дозволяє емпірично підтвердити, що поєднання:

- апаратної багаторівневої фільтрації,
- адаптивної тригерної логіки,
- нейронного модуля прогнозування або обробки,
- механізмів балансування і перебудови мережі,
- формує масштабовану, детерміновану та стійку до збоїв архітектуру IoT з реалізацією принципу граничного інтелекту.

4.2. Дослідження способу адаптивного налаштування порогів контролю у моніторингових IoT системах

4.2.1. Загальна структура експериментів

Метою проведених експериментальних досліджень є оцінювання ефективності використання апаратного модуля фільтрації на ПЛІС для первинної обробки поточкових сенсорних даних у системах Інтернету речей.

Основними завданнями експерименту є:

- визначення рівня покращення якості сенсорних даних після застосування багаторівневої фільтрації;
- оцінювання ефективності апаратної реалізації фільтрації на ПЛІС у порівнянні з програмною реалізацією аналогічних алгоритмів на центральному процесорі граничного вузла (HPS-процесор SoC/ПЛІС).

Об'єктом дослідження є система збору та первинної обробки даних сенсорного середовища, що функціонує на рівні граничного вузла.

Предметом дослідження виступають алгоритми багаторівневої цифрової фільтрації поточкових сенсорних даних, реалізовані у вигляді апаратного модуля на ПЛІС та їх вплив на якість сигналів і ефективність використання обчислювальних ресурсів.

Архітектура експериментального стенду включає два основних рівні:

1. Сенсорний рівень

- розподілені сенсорні вузли;
- датчики температури та вологості різних типів.

2. Граничний рівень

- граничний вузол на базі платформи SoC/ПЛІС;
- апаратний модуль фільтрації, реалізований у логічній частині ПЛІС;
- програмні модулі збору та передавання даних.

Загальна схема потоків даних експериментальної системи наведена на рис.

4.1.

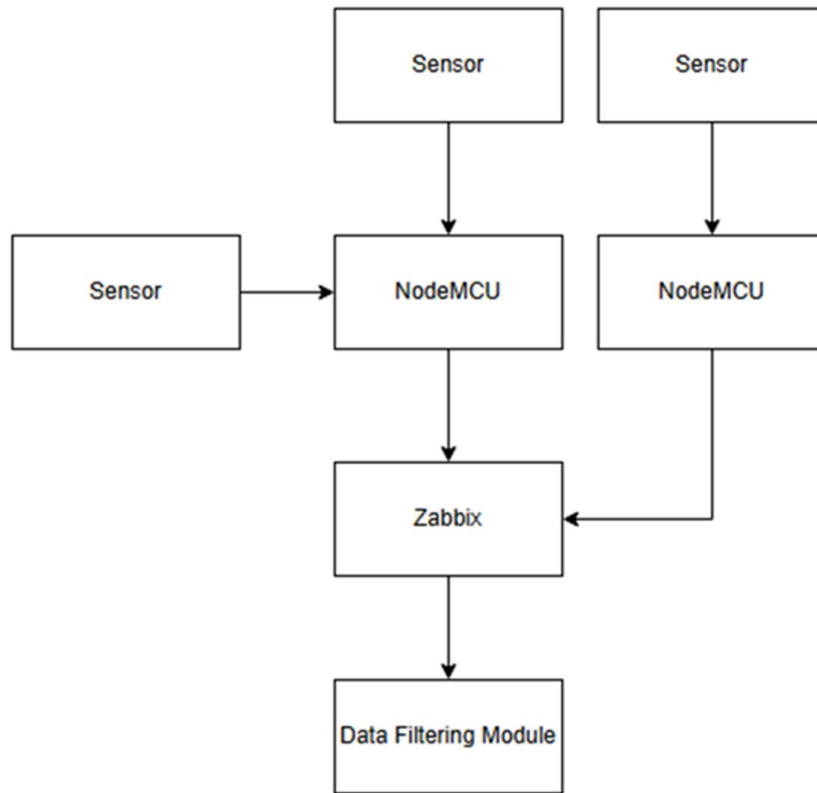


Рис. 4.1 – Схема потоків даних експериментальної системи

Сенсорні вузли формують потоки первинних вимірювань, які передаються до граничного вузла. На цьому рівні дані можуть оброблятися двома способами:

- програмно на процесорі HPS;
- апаратно у вигляді спеціалізованого фільтруючого модуля на ПЛІС.

Результати обробки передаються до системи моніторингу для подальшого аналізу, зберігання та порівняння.

Експериментальні дослідження були організовані у вигляді двох основних груп експериментів:

1. експеримент перевірки покращення якості сенсорних даних після фільтрації;
2. експеримент оцінювання ефективності використання обчислювальних ресурсів при апаратній реалізації фільтрації.

4.2.2. Перевірка ефективності очищення сенсорних даних

Метою даного експерименту є дослідження впливу багаторівневої фільтрації на якість сенсорних даних та оцінювання здатності запропонованого підходу зменшувати рівень шумів і випадкових відхилень у часових рядах вимірювань.

Для організації централізованого збору, зберігання та контролю стану сенсорної інфраструктури використано систему моніторингу Zabbix, яка виступає базовою інфраструктурною платформою експериментального середовища.

У межах дослідження система Zabbix виконувала такі функції:

- збір телеметричних даних у режимі реального часу з розподілених сенсорних вузлів;
- довготривале зберігання та логування як первинних («сирих»), так і оброблених значень;
- виявлення аномальних або нестабільних показників за допомогою системи тригерів;
- порівняльний аналіз часових рядів до та після фільтрації;
- формування адаптивних порогових значень на основі статистичних характеристик сигналів;
- забезпечення подальшої аналітики та оцінювання стабільності сенсорної мережі.

Таким чином, система Zabbix використовувалася не лише як інструмент моніторингу, але і як платформа експериментального аналізу ефективності алгоритмів фільтрації.

У дослідженні використано гетерогенний набір сенсорів, що відрізняються принципом дії, рівнем точності, чутливістю та динамікою відгуку. Такий підхід дозволяє оцінити універсальність запропонованої методики фільтрації як у відносно стабільних умовах вимірювання, так і в умовах динамічних змін параметрів середовища.

Зокрема, застосовано резистивний датчик вологості HR31, який характеризується високою чутливістю до змін відносної вологості повітря. Водночас для нього притаманні нестабільність показників унаслідок

температурного впливу та деградації активного шару. За наявності конденсату або різких змін вологості можливі короточасні імпульсні відхилення, що формують пікові значення у часовому ряді. Це обґрунтовує необхідність застосування згладжувальних фільтрів із пригніченням одиничних викидів.

Аналоговий датчик температури КТУ81-210 забезпечує квазі-лінійну залежність опору від температури, що спрощує калібрування та математичну інтерпретацію сигналу. Проте він є чутливим до електромагнітних завад та нестабільності живлення. Довгі сигнальні лінії між сенсором і контролером підсилюють вплив індукованих шумів, а також збільшують похибку АЦП. У таких умовах необхідна попередня аналогова та цифрова фільтрація з урахуванням спектральних характеристик шуму.

Цифровий датчик температури DS18B20 характеризується підвищеною точністю (до $\pm 0,5$ °C) та стійкістю до перешкод завдяки цифровому інтерфейсу передачі даних. Однак для нього характерна інерційність вимірювань, що проявляється у затримці реакції на різкі зміни температури. У часових рядах це призводить до фазового зсуву відносно реального процесу. Застосування адаптивних фільтрів дозволяє мінімізувати вплив короточасних відхилень без втрати загальної тенденції сигналу.

Модуль DHT11 є недорогим комбінованим сенсором температури та вологості з відносно низькою точністю (порядку ± 2 °C та ± 5 %). Він схильний до випадкових стрибків значень через помилки зчитування або електричні завади, що може призводити до хибних спрацювань тригерів у системі моніторингу. Це обґрунтовує доцільність реалізації фільтрації безпосередньо на рівні контролера перед передачею даних у централізовану систему.

Модуль DHT22 забезпечує вищу точність (до $\pm 0,5$ °C та $\pm 2-3$ %) і кращу стабільність порівняно з DHT11, проте також демонструє інерційність та підвищення похибки за умов високої вологості. Для компенсації зазначених ефектів доцільно застосовувати адаптивні коефіцієнти згладжування, які змінюються залежно від дисперсії поточного сигналу.

Логіка потоків даних експериментальної системи передбачає послідовне формування, передачу, збереження та повторну обробку інформації. Сенсорні вузли генерують первинні вимірювання, які через контролер передаються до системи моніторингу, де фіксуються «сирі» значення. Після цього дані проходять багаторівневу фільтрацію на граничному рівні або у спеціалізованому обчислювальному модулі, а результати повторно надходять до Zabbix для порівняльного аналізу та оцінювання стабільності сигналу. Такий підхід забезпечує можливість кількісного визначення ефективності фільтрації за критеріями зменшення дисперсії, пригнічення пікових відхилень та збереження довгострокових трендів.

Проведення експерименту здійснювалося у декілька послідовних етапів, що забезпечували контрольоване порівняння стану сигналу до та після застосування адаптивної обробки. На першому етапі здійснювався накопичувальний збір «сирих» даних протягом визначеного інтервалу спостереження.

На другому етапі проводилося попереднє статистичне оцінювання отриманих часових рядів. Для кожного сенсора визначалися базові характеристики: середнє значення, медіана, дисперсія, середньоквадратичне відхилення, амплітуда коливань та частота появи аномальних пікових значень. Окремо фіксувалися випадки короткочасних імпульсних викидів, які перевищували локальне середнє більш ніж на встановлений коефіцієнт відхилення. Це створило базову «контрольну» вибірку, з якою надалі порівнювалися результати після фільтрації.

Третій етап передбачав активацію багаторівневої фільтрації на граничному рівні. Послідовне застосування медіанного фільтра, ковзного середнього та експоненціального згладжування дозволило сформувати стабілізований сигнал із збереженням основної тенденції. Після кожного рівня обробки здійснювався проміжний аналіз змін дисперсії та структури коливань, що дало змогу оцінити внесок кожного фільтра в загальний результат. Було встановлено, що медіанний фільтр є найбільш ефективним для усунення одиничних імпульсних стрибків, тоді як ковзне середнє забезпечує зменшення випадкових флуктуацій, а

експоненціальне згладжування формує інерційну стабільність сигналу без суттєвої втрати трендової інформації.

Даний експеримент потрібно розділити на 2 частини:

1. Перевірка працездатності фільтра в синтетичному програмному середовищі на тестовому наборі даних
2. Перевірка роботи модуля фільтрації на даних з різних датчиків.

Перша частина була проведена в програмі ModelSim, було перевірено роботу модуля фільтрації рисунок 4.2

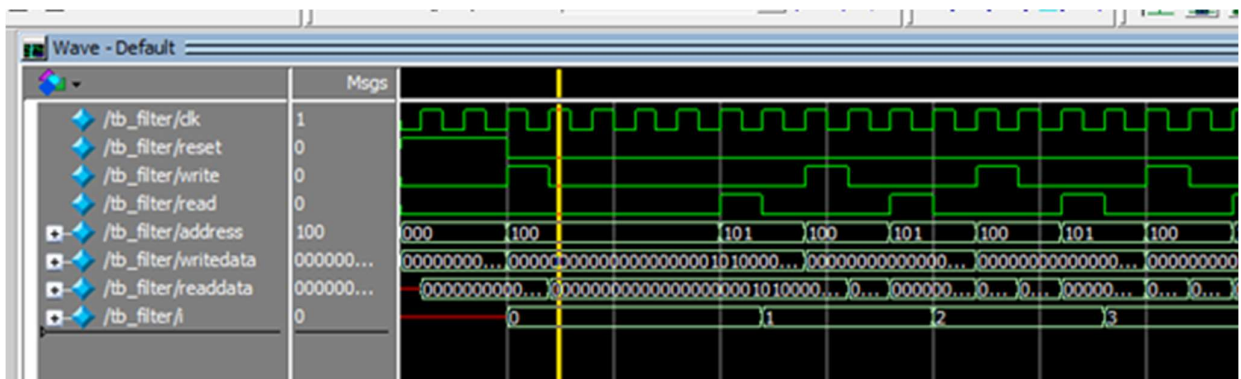
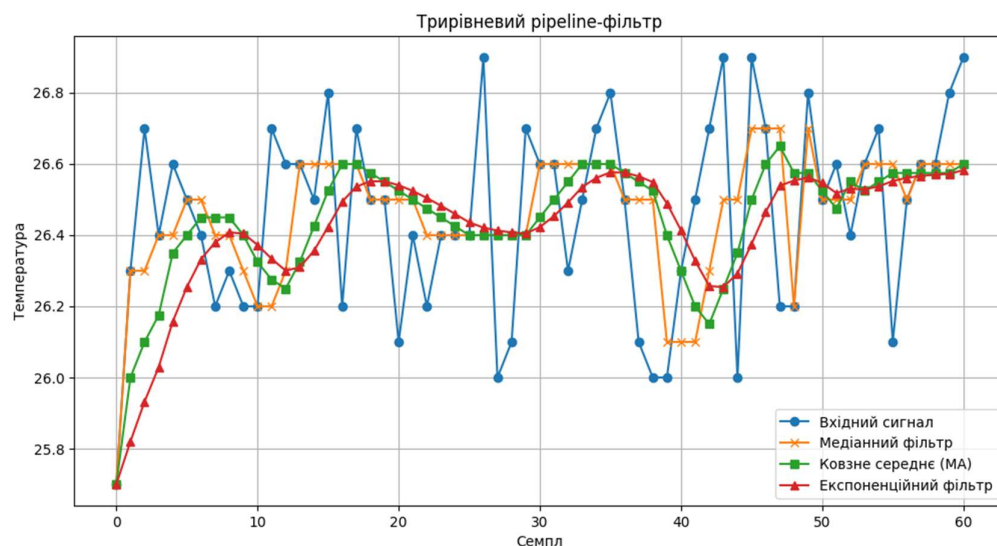


Рисунок 4.2. Перевірка роботи модулю фільтрації в ModelSim

Тестові дані та результати роботи фільтрів зображені на рисунку 4.3. та таблиці 4.1



Рисунку 4.3. Графік послідовної роботи фільтра

Таблиця 4.1

Опис графіку послідовної роботи фільтра

Лінія на графіку	Характеристика сигналу	Ефект фільтрації
Синя - вхідний сигнал	Реальні значення температури з датчика, що подаються по одному семплу	Чітко виражений шум і короткочасні стрибки; на ділянках 10–20 та 40–50 семплів спостерігаються різкі коливання
Помаранчева - медіанний фільтр	Результат нелінійної фільтрації з вікном 5	Усунення імпульсних перешкод; різкі піки та провали згладжені без втрати загального тренду
Зелена - ковзне середнє	Усереднений сигнал після медіанного фільтра	Додаткове згладжування залишкових флуктуацій; формування стабільніших переходів між значеннями
Червона - експоненційне згладжування	Фінальний вихід трирівневого фільтра	Формування плавного та стабільного тренду; практично повне усунення шумів, придатність для систем контролю

Друга частина перевірки здійснювалась на описаному вище наборі сенсорних даних.

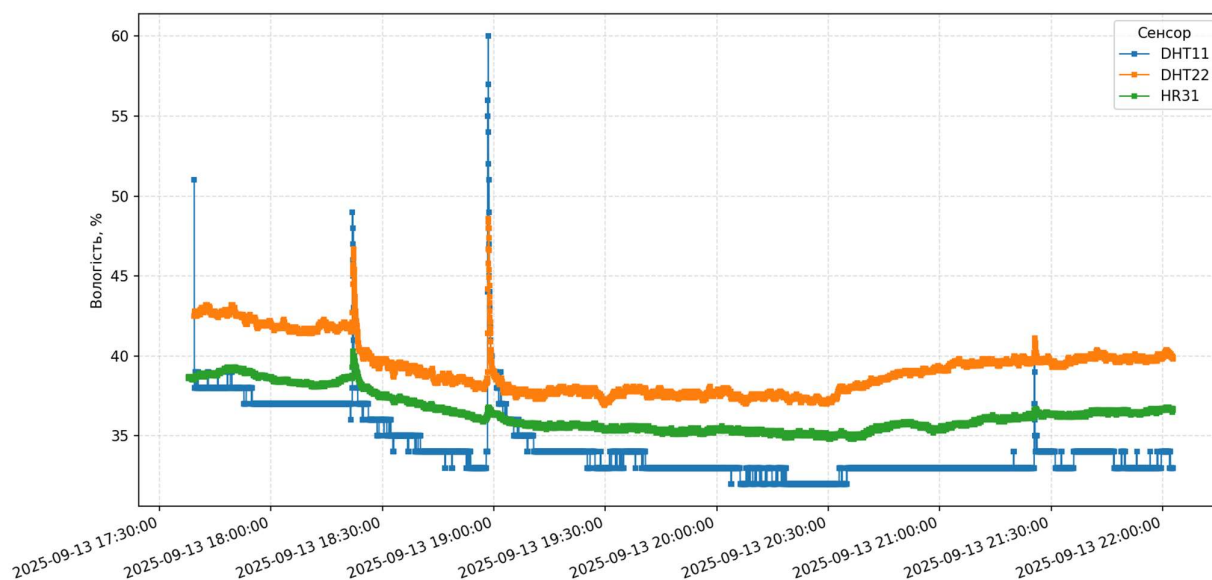


Рис. 4.4. Загальний графік вимірювання вологості

На рис. 4.4 наведено узагальнений часовий ряд вимірювань відносної вологості, отриманий від трьох різних сенсорів - DHT11, DHT22 та HR31 - за однакових умов експерименту. Аналіз представлених графіків дозволяє встановити істотні відмінності у статистичних характеристиках сигналів, що безпосередньо впливають на можливість їх використання в системах автоматизованого моніторингу.

Сенсор DHT11 характеризується найбільшою варіативністю сигналу. Для нього зафіксовано підвищену дисперсію, наявність імпульсних викидів та локальних стрибків значень, які не відповідають фізично обґрунтованій динаміці зміни вологості. Подібні сплески особливо помітні під час швидких змін мікроклімату або при короткочасних перешкодах у каналі передачі даних. З огляду на невисоку точність і внутрішню дискретизацію вимірювань, сигнал має виражену «зернисту» структуру, що ускладнює безпосереднє застосування таких даних у тригерній логіці.

DHT22 демонструє значно кращу стабільність: часовий ряд є більш згладженим, амплітуда коливань менша, а тренд простежується чіткіше. Проте навіть для цього сенсора фіксуються короткочасні флуктуації, хоча їх інтенсивність

істотно нижча порівняно з DHT11. Таким чином, DHT22 забезпечує вищу якість первинних даних, однак не гарантує повної відсутності шумової складової.

Резистивний сенсор HR31 формує відносно рівномірний сигнал без виражених імпульсних піків. Водночас упродовж експерименту спостерігається поступове зміщення показників (дрейф), що проявляється як повільна тенденція до зниження або підвищення значень незалежно від реальної зміни умов. Цей ефект пов'язаний із температурною залежністю матеріалу чутливого елемента та процесами його повільної адаптації до середовища. На відміну від DHT11, проблема HR31 полягає не у випадкових стрибках, а у довготривалих систематичних відхиленнях.

Отже, рис. 4.4 підтверджує, що для однієї і тієї ж фізичної величини різні сенсори формують сигнали з різним типом нестабільності:

- для DHT11 критичними є імпульсні викиди та високочастотний шум;
- для HR31 - повільний дрейф;
- для DHT22 - залишкові коливання середньої інтенсивності.

Це обґрунтовує доцільність застосування багаторівневої фільтрації, здатної одночасно компенсувати як короткочасні імпульсні перешкоди, так і повільні трендові викривлення.

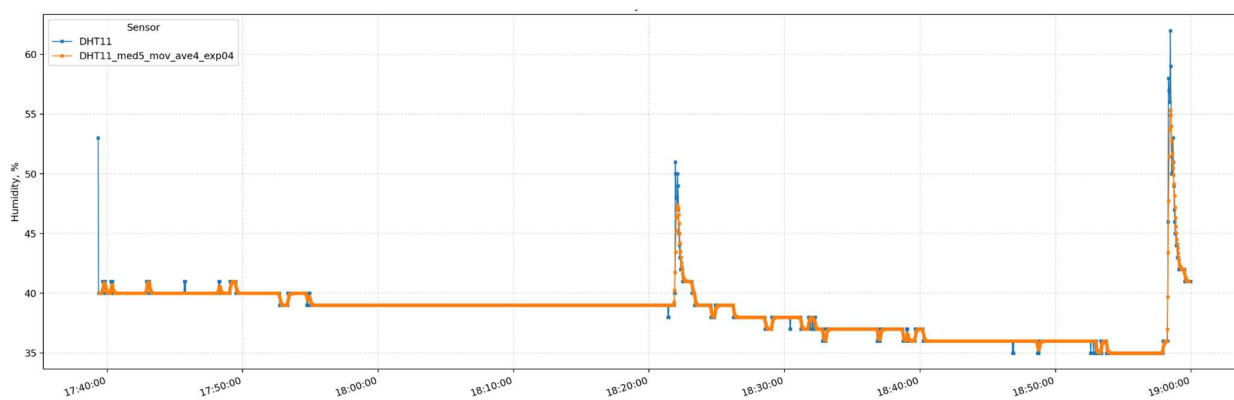


Рис. 4.5. Графіки фільтрованих даних з датчиків DHT11

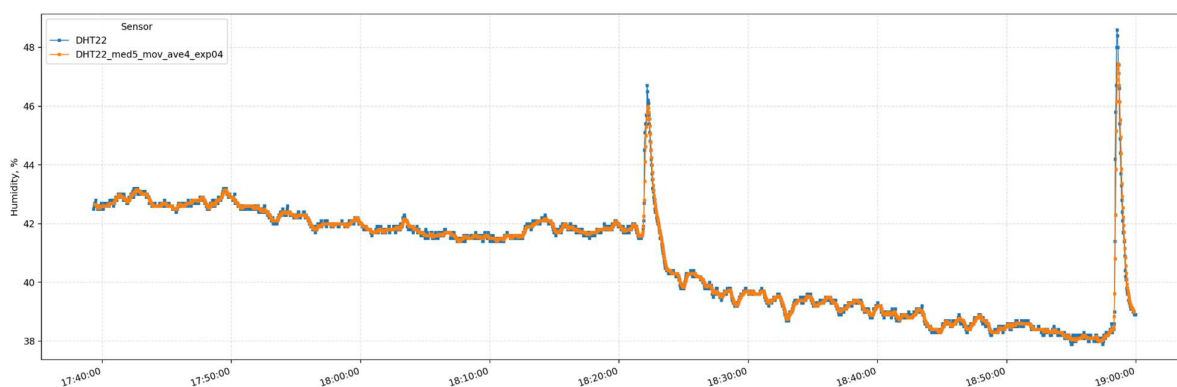


Рис. 4.6. Графіки фільтрованих даних з датчиків DHT22

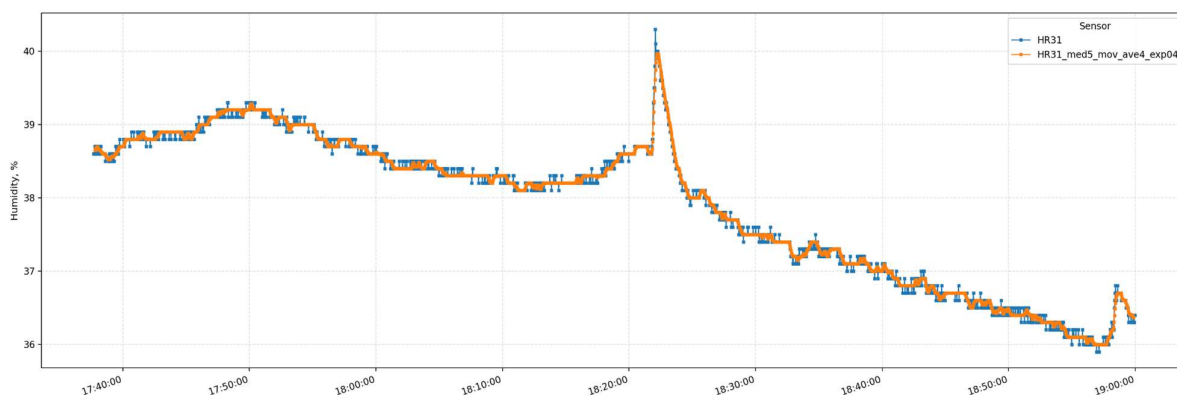


Рис. 4.7. Графіки фільтрованих даних з датчика HR31

На рис. 4.5–4.7 наведено результати порівняння часових рядів вологості до та після застосування трирівневої фільтрації. Візуальний та статистичний аналіз демонструє суттєве зменшення дисперсії сигналу та приглушення локальних сплесків.

Для DHT11 після обробки спостерігається найбільш помітне покращення: імпульсні викиди практично усунуті, а випадкові коливання згладжені без втрати загальної тенденції. Тренд стає чітко вираженим, що дозволяє використовувати дані у системі моніторингу без надмірного спрацювання тригерів. З огляду на масове використання DHT11 у бюджетних IoT-рішеннях, отриманий ефект має значну практичну цінність.

Для DHT22 обробка забезпечує додаткове приглушення високочастотної складової, що підвищує стабільність тренду та покращує інтерпретацію повільних

змін вологості. Хоча вихідний сигнал цього сенсора є відносно якісним, фільтрація підвищує його придатність до автоматизованого аналізу.

Для HR31 фільтрація зменшує вплив короточасних відхилень і частково стабілізує сигнал, однак довготривалий дрейф залишається помітним. Це свідчить про те, що для резистивних сенсорів доцільно доповнювати фільтрацію процедурами періодичної калібровки або температурної компенсації.

Таким чином, результати на рис. 4.5 підтверджують універсальність запропонованого підходу: багаторівнева фільтрація знижує рівень шуму, приглушує випадкові викиди та забезпечує більш достовірне відображення реальної динаміки вологості.

Порівняльний аналіз сенсорів температури



Рис. 4.8. Загальний графік вимірювання температури

На рис. 4.8. представлено часові ряди температури, отримані від чотирьох сенсорів: DHT11, DHT22, DS18B20 та KTY81. Аналіз показує, що характер спотворень температурних сигналів є різномірним і залежить від принципу дії сенсора.

DHT11 формує найбільш нестабільний сигнал із помітними стрибками та шумовою складовою. Низька роздільна здатність і внутрішня обробка даних зумовлюють наявність різких переходів між значеннями, що ускладнює формування коректної тригерної логіки.

DHT22 демонструє більш плавну поведінку, однак характерною особливістю є «сходинковість» сигналу, що пов'язана з дискретною природою вимірювання та алгоритмами внутрішнього квантування. Це створює штучну нерівномірність, яка не завжди відповідає реальній зміні температури.

DS18B20 відзначається високою стабільністю та точністю. Проте при різких змінах середовища проявляється інерційність, що призводить до часової затримки реакції. Крім того, можуть виникати поодинокі відхилення, пов'язані з особливостями цифрової шини.

КТУ81, як аналоговий датчик, формує плавний тренд, однак є чутливим до електромагнітних завад, нестабільності живлення та похибок аналого-цифрового перетворення. У часовому ряді це проявляється як шум або локальні спотворення.

Таким чином, рис. 4.8 підтверджує необхідність попередньої обробки температурних сигналів незалежно від типу сенсора.

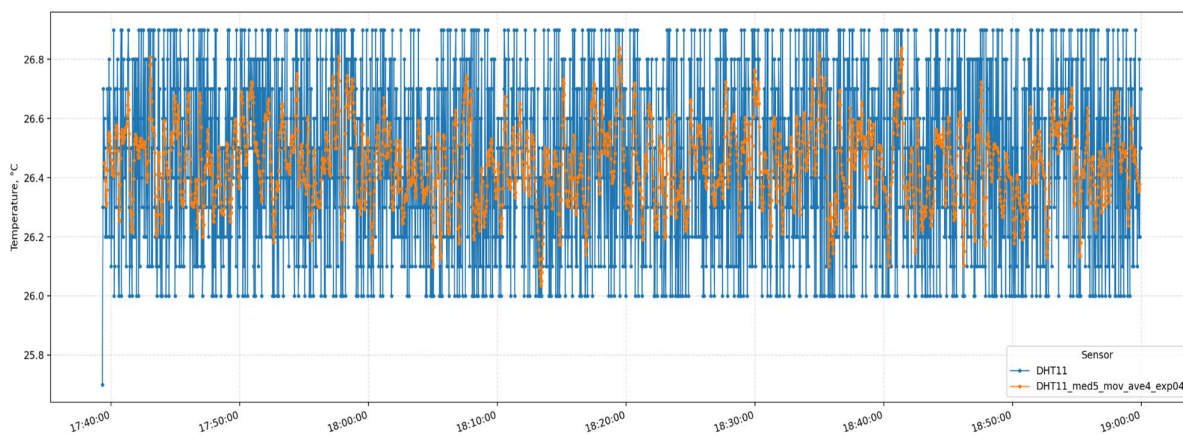


Рис. 4.9. Графіки фільтрованих даних з датчиків DHT11

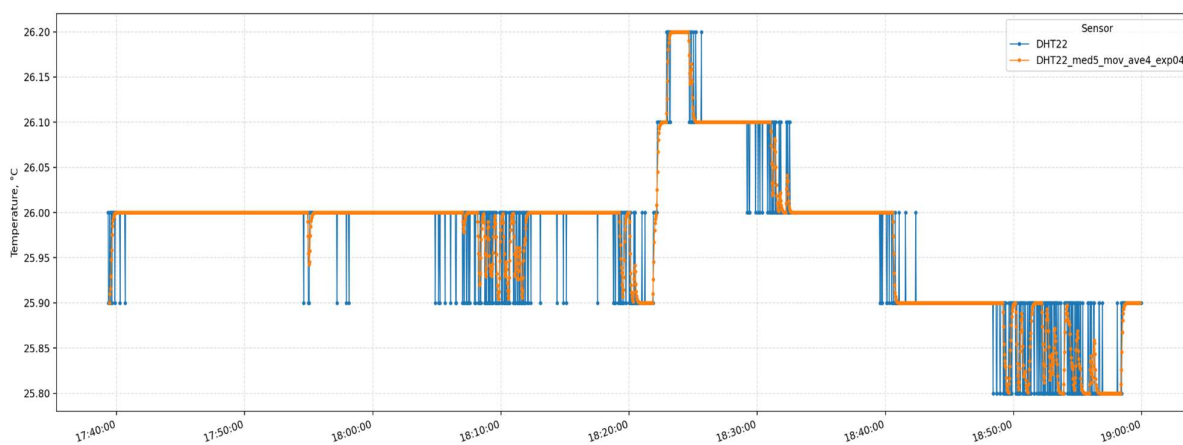


Рис. 4.10. Графіки фільтрованих даних з датчиків DHT22

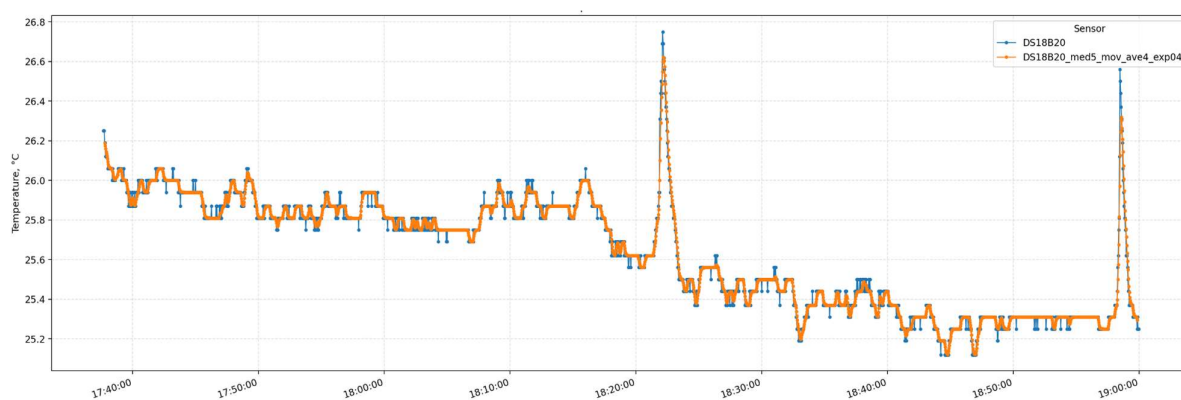


Рис. 4.11. Графіки фільтрованих даних з датчиків DS18B20

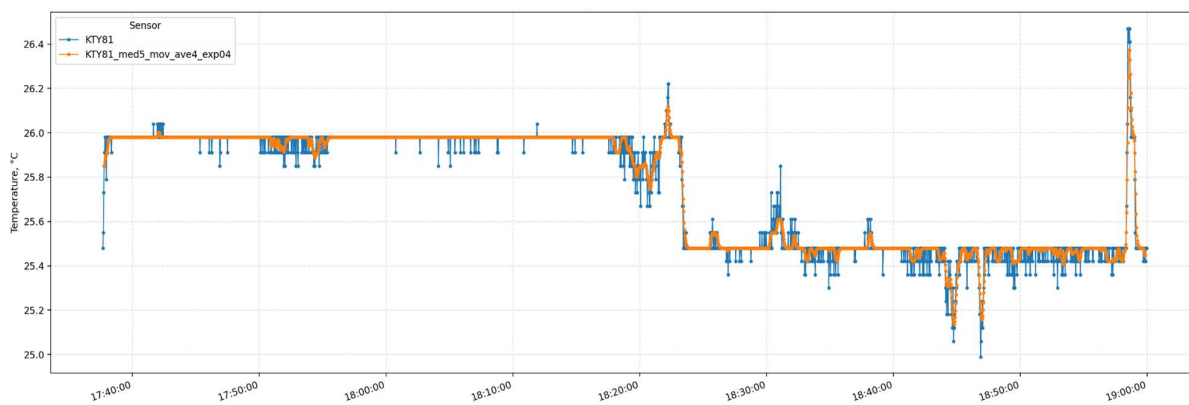


Рис. 4.12. Графіки фільтрованих даних з датчиків KTY81

На рис. 4.9–4.12 представлено порівняння температурних сигналів до та після застосування багаторівневої фільтрації. Після обробки спостерігається зменшення амплітуди випадкових коливань та усунення поодиноких імпульсних відхилень.

Найбільш виражений ефект знову зафіксовано для DHT11: значно зменшується кількість різких стрибків, а тренд набуває безперервного характеру. Для DHT22 фільтрація зменшує «сходинкову» структуру сигналу, що робить часовий ряд більш плавним і фізично обґрунтованим.

Для DS18B20 та KTY81 обробка виконує переважно стабілізуючу функцію: прибирає локальні флуктуації та підвищує чіткість тренду без спотворення його динаміки. Це особливо важливо при застосуванні динамічних порогів контролю, оскільки стабілізований сигнал знижує ризик хибних спрацювань.

Для математичної перевірки було обраховано середнє значення за формулою (44), середнє квадратичне відхилення за формулою (45) для показників температури та вологості кожного датчика, результати наведені в таблиці. 4.2.

$$\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad (44)$$

де $\bar{x}$ – середнє значення вибірки, N – кількість вимірювань, x_i – i -те значення у вибірці.

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (45)$$

де $\bar{x}$ – середнє значення вибірки, N – кількість вимірювань, x_i – i -те значення у вибірці.

Аналіз результатів показав суттєве зниження варіативності сигналів після багаторівневої обробки. Для сенсорів із підвищеною нестабільністю (зокрема DHT11 та HR31) зафіксовано найбільший ефект у частині приглушення імпульсних викидів і зменшення частоти короткочасних відхилень. У випадку більш стабільних цифрових сенсорів (DS18B20, DHT22) фільтрація забезпечила покращення візуальної інтерпретації трендів та усунення незначних високочастотних коливань, що раніше спричиняли некоректні тригерні події.

Таблиця 4.2

Порівняння середніх значень та середньоквадратичних відхилень
до та після фільтрації

Назва датчика та показник	Середнє до фільтрації	СКВ до фільтрації	Середнє після фільтрації	СКВ після фільтрації	Різниця
DHT11 температура	26.4476	0.2912	26.4465	0.1366	53.06%
DHT22 температура	25.9756	0.0803	25.9772	0.0786	2.21%
DS18B20 температура	25.6412	0.2788	25.6416	0.2752	1.28%
KTY81 температура	25.7463	0.2587	25.7508	0.2543	1.68%
HR31 вологість	37.8965	0.9788	37.8997	0.9756	0.32%
DHT11 вологість	38.3669	2.2165	38.3493	2.072	6.52%
DHT22 вологість	40.7872	1.7343	40.7882	1.7189	0.89%

Окрему увагу приділено порівнянню поведінки системи у різних сценаріях експлуатації: стабільний температурний режим, різкі зміни через провітрювання приміщення, підвищення вологості під час зволоження повітря. У всіх сценаріях адаптивна методика продемонструвала здатність відокремлювати короточасні перешкоди від довготривалих змін стану середовища.

4.2.3. Оцінка ефективності використання обчислювальних ресурсів

Подальший аналіз експериментальних результатів був спрямований на кількісну оцінку ефективності апаратної реалізації трирівневого фільтра з точки зору ресурсних витрат, часових характеристик та енергоефективності в порівнянні з програмною реалізацією на HPS-процесорі.

Синтез та компіляція проєкту у середовищі Quartus показали, що запропонований фільтруючий модуль займає помірну частину логічних ресурсів ПЛІС, що робить його придатним для інтеграції в більш складні обчислювальні структури. Основне навантаження припадає на:

- логічні елементи (ALMs/LE), що реалізують порівняльні схеми медіанного фільтра;
- регістрові ресурси, необхідні для конвеєрної організації;
- арифметичні блоки для реалізації операцій додавання та множення у форматі з фіксованою точкою.

Застосування фіксованої точки дозволило відмовитися від використання ресурсомістких блоків обчислень з плаваючою комою, що суттєво зменшило апаратні витрати та спростило маршрутизацію сигналів. У результаті модуль може бути масштабований (наприклад, для паралельної обробки кількох сенсорних потоків) без критичного перевищення апаратного бюджету.

Проведений статичний часовий аналіз підтвердив коректність синхронізації між стадіями конвеєра та відсутність критичних затримок на тактовій частоті, що використовувалася в експерименті. Після заповнення конвеєра система працює в режимі повного перекриття стадій, що забезпечує сталість затримки та детермінований характер обробки.

Фіксована латентність у три такти після подачі першого семплу дозволяє прогнозувати часову поведінку модуля при інтеграції у більш складні системи керування. У режимі усталеної роботи кожен тактовий цикл формує новий відфільтрований результат, що особливо важливо для задач реального часу, де необхідна гарантована швидкодія без варіативності затримки.

В ході експерименту було використано формули 16–22 та дані показників вимірювального пристрою який показував скільки електроенергії було використано пристроєм за час обробки визначеного синтетичного набору даних.

Вхідні дані:

- Виміри температури зібрані в синтетичний набір даних об'ємом 5 мільйонів вимірів.
- Програмна та апаратна реалізація фільтрування.

Проведення експерименту:

1. Увімкнення ПЛІС.
2. Запуск процесу фільтрації
3. Фіксація часу експерименту та затраченої енергії після закінчення фільтрації набору даних.
4. Повторення експерименту 10 разів.
5. Обчислення середніх значень витраченої енергії та часу обробки.

Результати наведені в таблиці 4.3

Відповідно після обчислення коефіцієнтів продуктивності бачимо що використання CPU має продуктивність на 13.01% меншу ніж використання ПЛІС та збільшене енергоспоживання на 12,5 %.

Порівняльне тестування з програмною реалізацією на HPS-процесорі показало, що при послідовній обробці кожен новий семпл потребує завершення повного циклу обчислень. У випадку ПЛІС реалізації обробка відбувається потоково, що усуває простой між операціями та забезпечує істотно вищу пропускну здатність при збереженні стабільного часу реакції.

Окремий блок експериментів було присвячено оцінюванню енергетичних витрат при масовій обробці даних. Апаратна реалізація на ПЛІС продемонструвала менший сумарний час виконання обчислень при близькому рівні споживаної потужності, що в перерахунку на один оброблений семпл забезпечує покращений показник енергоефективності.

Цей результат пояснюється тим, що ПЛІС виконує операції паралельно та без накладних витрат операційної системи, контекстних перемикачів і керування

пам'яттю, притаманних процесорній архітектурі. Таким чином, при великих обсягах потокових даних апаратна фільтрація є більш доцільною з точки зору співвідношення продуктивності та енергоспоживання, що особливо актуально для граничних вузлів з обмеженим енергетичним бюджетом.

Таблиця 4.3

Показники ефективності роботи ПЛІС та ЦП

Показник	ПЛІС(DE1-SoC)	ЦП (ARM Cortex-A9)
Час обробки	68.9с	79.4с
Використана енергія	453.6 Дж (0.126 Вт-год)	518.8 Дж (0.1441 Вт-год)
K_1	72568,9405	62972,2922
K_2	0.00009072	0.00010376
K_3	0.00001378	0.00001588
K_{1norm}	1	0.86775818643
K_{2norm}	1	0.87432536623
K_{3norm}	1	0.86775818639
K_s	1	0.86994724635

З практичної точки зору це означає, що модуль може бути інтегрований у реальну інфраструктуру IoT без ризику деградації даних при пікових навантаженнях або нестабільності вхідного потоку.

Сукупність проведених досліджень підтверджує, що реалізація трирівневого фільтра у вигляді конвеєрного апаратного модуля на платформі DE1-SoC забезпечує:

- детерміновану часову поведінку;
- високу пропускну здатність (1 семпл за такт у сталому режимі);
- ефективне приглушення різних типів шуму;
- збереження довгострокових трендів сигналу;
- покращену енергоефективність при обробці великих масивів даних;

- можливість масштабування та інтеграції у більш складні системи.

4.3. Експериментальна перевірка механізму балансування навантаження

Метою даного етапу експериментальних досліджень є перевірка працездатності та оцінювання ефективності механізму балансування навантаження у розподіленій тришаровій архітектурі IoT в умовах нерівномірного обчислювального навантаження на граничні вузли.

Основними завданнями дослідження є:

- перевірка здатності системи виявляти перевантаження окремого вузла на основі даних моніторингу;
- дослідження алгоритму ухвалення рішення щодо перерозподілу обчислювальних задач;
- оцінювання швидкості реакції системи на зміну навантаження;
- аналіз впливу балансування на стабільність функціонування вузлів;
- перевірка зниження пікових значень завантаження процесора, оперативної пам'яті та супутніх системних ресурсів після виконання перебалансування.

Об'єктом дослідження є процес функціонування розподіленої гранично-орієнтованої системи IoT в умовах змінного обчислювального навантаження.

Предметом дослідження є алгоритм балансування навантаження, реалізований на основі моніторингових даних, що надходять від системи Zabbix, та механізму перерозподілу задач між вузлами.

Відповідно до сформованої архітектурної концепції розподіленої тришарової системи IoT було розроблено експериментальний прототип, реалізований у вигляді комплексу віртуальних та фізичних серверів.

Прототип складається з віртуальних машин, розгорнутих у середовищі віртуалізації VMware ESXi. Фізично платформа функціонує на серверах HP DL360 G7, оснащених чотирма гігабітними мережевими інтерфейсами та підключених до комутатора HP J9050A ProCurve 2900. На рівні гіпервізора використано єдиний

віртуальний комутатор, який забезпечує внутрішню мережеву взаємодію між усіма віртуальними та фізичними серверами та без залучення зовнішньої маршрутизації для внутрішніх віртуальних машин. Такий підхід дозволив мінімізувати вплив мережових затримок на результати експерименту та зосередитися на аналізі обчислювального навантаження.

Функціональний розподіл ролей у прототипі реалізовано таким чином:

- два сервери на базі Ubuntu Server 24.04 виконують функції граничних вузлів (Main Server та Reserve Server);
- інші вузли такі як Windows server та Raspberry Pi виступають джерелами даних.

Граничні сервери мають по шість обчислювальних ядер і 12 ГБ оперативної пам'яті, що забезпечує достатній резерв ресурсів для обробки поточкових даних. Сервери-джерела даних обладнані чотирма ядрами та 4 ГБ оперативної пам'яті й моделюють роботу периферійних вузлів генерації телеметрії.

У запропонованій моделі хмарний рівень використовується виключно для зберігання кінцевих результатів обробки та централізованого моніторингу параметрів граничних вузлів. Це дозволяє мінімізувати ризик втрати даних у разі відмови одного з обчислювальних пристроїв та відокремити функції довготривалого зберігання від задач оперативної обробки.

Кожен граничний вузол оснащено екземпляром системи моніторингу Zabbix, який виконує такі функції:

- збір метрик навантаження (CPU, використання пам'яті, мережевої активності);
- обчислення інтегрального показника завантаженості;
- аналіз порогових значень;
- автоматичний запуск керуючих скриптів у разі перевищення допустимих меж.

Таким чином, Zabbix у даному прототипі виступає не лише інструментом спостереження, але й елементом активного керування, що реалізує механізм прийняття рішень у межах граничного рівня.

Архітектура орієнтована на сценарії значного навантаження, які характерні для реальних систем IoT у випадках:

- підключення нових джерел даних;
- різкого збільшення частоти надходження телеметрії;
- запуску ресурсоємних алгоритмів обробки (аналітика, фільтрація, нейронні модулі).

Інтегральний показник формується за формулою (30). У рамках експерименту вимірювання виконувалися кожні 20 секунд, при цьому для згладжування короточасних коливань використовувалося усереднення останніх $n=9$ вимірів. Такий підхід забезпечує компроміс між швидкістю реагування та стійкістю до імпульсних перевантажень.

Порогове значення інтегрального показника встановлено на рівні $P_{max} = 80\%$ для основного сервера та на рівні $P_{ma} = 95\%$ для резервного. Перевищення цього значення свідчить про наближення вузла до критичного рівня використання обчислювальних ресурсів і використовується як умова активації механізму автоматичної реконфігурації системи. У разі виконання умови (31) у системі активується тригер оцінювання стану обчислювальної інфраструктури. Алгоритм реагування включає такі етапи:

1. Виконується повторний розрахунок інтегрального показника навантаження для всіх доступних вузлів відповідно до формули (30).
2. Визначення вузла з мінімальним рівнем завантаження. Для вибору вузла, здатного прийняти додаткове навантаження, застосовується формула (32) що дозволяє визначити найбільш вільний вузол обчислювальної інфраструктури..
3. Ініціація процедури міграції джерела даних. Після визначення цільового вузла створюється новий об'єкт джерела даних на резервному сервері.
4. Оновлення конфігурації генератора даних. Серверу генерації даних передається нова IP-адреса вузла, на який перенесено обробку.

- Деактивація попереднього джерела. Після успішного завершення перенесення попередній об'єкт джерела даних на основному сервері деактивується.

Реалізація алгоритму здійснена у вигляді автоматизованих скриптів, інтегрованих із системою моніторингу Zabbix, що дозволяє виконувати всі етапи процедури без участі оператора та забезпечує швидку реакцію на зміну стану системи.

Для перевірки працездатності запропонованого методу було реалізовано сценарій контрольованого перевантаження обчислювального вузла. У рамках експерименту до Main Server підключався пристрій Raspberry Pi, який виступав джерелом поточних даних. Після встановлення з'єднання автоматично запускалася ресурсоємна задача емуляції обчислювального навантаження.

У процесі роботи система моніторингу Zabbix здійснювала збір показників використання CPU, RAM та дискового простору, на основі яких обчислювався інтегральний показник навантаження P_i .

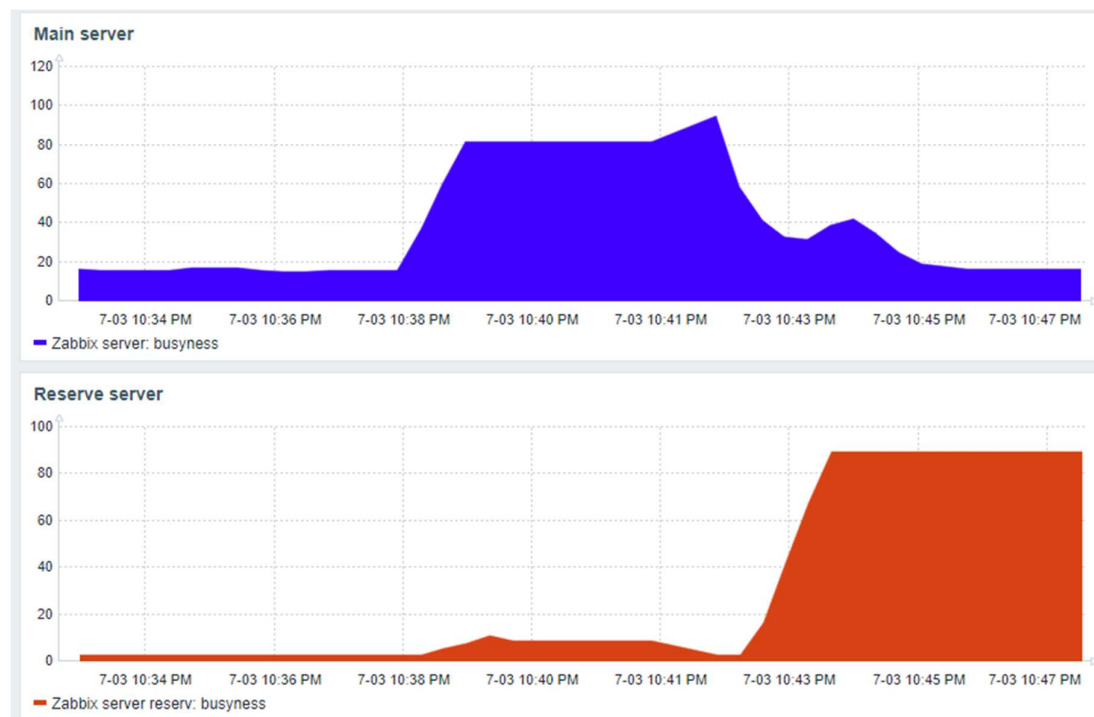


Рис. 4.13. Динаміка зміни інтегрального показника навантаження серверів

Графічний аналіз (рис. 4.13) показує, що після ініціалізації задачі емуляції навантаження відбулося поступове зростання значення інтегрального показника

для Main Server. Через дискретність вимірювання (20 секунд) на графіку спостерігається незначне зміщення моменту фіксації подій відносно реального часу запуску задачі.

Приблизно через три хвилини після початку експерименту інтегральний показник навантаження перевищив встановлений поріг P_{max} це зумовлено тим що спрацювання тригера було налаштовано на останні 9 вимірювань, для уникнення тимчасових піків навантаження. У цей момент система автоматично активувала механізм реконфігурації.

У результаті було виконано процедуру перенесення джерела даних із Main Server на Reserve Server. На початковому етапі міграції спостерігалось короткочасне додаткове зростання навантаження на основному сервері, що пояснюється виконанням конфігураційних операцій, пов'язаних зі створенням нового джерела даних та оновленням параметрів підключення.

Після завершення процесу міграції пристрій Raspberry Pi був автоматично перепідключений до Reserve Server, що призвело до перерозподілу обчислювального навантаження між вузлами системи.

Подальший аналіз отриманих графіків показав:

- суттєве зниження інтегрального показника навантаження Main Server після перенесення задачі;
- зростання навантаження Reserve Server, що підтверджує перенесення обчислювального процесу;
- стабілізацію роботи обох вузлів у допустимих межах навантаження.

Отримані результати підтверджують ефективність запропонованого методу динамічної реконфігурації обчислювальної інфраструктури. Використання інтегрального показника навантаження дозволяє своєчасно виявляти перевантаження вузлів та автоматично перерозподіляти обчислювальні задачі, що забезпечує підвищення стабільності функціонування розподіленої системи IoT та більш ефективне використання доступних обчислювальних ресурсів.

4.4. Перевірка методу автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT

У попередніх розділах було сформовано теоретичні засади методу автоматичного прийняття рішень у безпекових сценаріях у розподілених системах IoT та визначено архітектуру модуля перебудови мережі, який реалізує керуючі дії на мережевому рівні. Практичним продовженням цих положень є експериментальна перевірка здатності системи не лише виявляти підозрілі стани, а й автоматично змінювати логічну топологію мережі у відповідь на інциденти безпеки.

У межах запропонованого підходу під зміною топології розуміється не фізична перебудова мережі, а контрольована зміна логічних маршрутів, правил доступу, мережевої ізоляції та доступності окремих вузлів або зовнішніх адрес. Така зміна реалізується шляхом автоматичного оновлення правил міжмережевого екрана, політик маршрутизації, псевдонімів, механізмів сегментації та, за потреби, зміни IP-конфігурації кінцевого вузла. На відміну від пасивного моніторингу, запропонований механізм дозволяє перейти до активної фази реагування, за якої система самостійно локалізує загрозу, зменшує площину потенційної атаки та обмежує ризик поширення інциденту на суміжні вузли тришарової архітектури IoT.

Експериментальна реалізація базувалася на поєднанні трьох компонентів: системи моніторингу Zabbix, яка виконувала безперервний збір та первинний аналіз телеметрії; зовнішніх скриптів, що розширювали стандартний функціонал агента та забезпечували збір даних про активні процеси й мережеві з'єднання; а також шлюзу pfSense, через який виконувалося автоматизоване оновлення правил фільтрації, блокування трафіку, зміна маршрутів та ізоляція вузлів. У такій конфігурації Zabbix виступає джерелом фактів та подій, модуль перебудови мережі - аналітичним та оркестраційним ядром, а pfSense - виконавчим рівнем застосування рішень.

Практична перевірка безпосередньо спирається на метод, описаний у підрозділі 2.4, згідно з яким процес реагування має уніфіковану структуру: збір даних → аналіз і зіставлення з політиками → прийняття рішення → виконання

керуючого впливу → аудит і валідація. У межах розглянутого експерименту ця схема була реалізована для двох базових класів інцидентів:

1. встановлення з'єднання з зовнішньою IP-адресою, що не входить до списку дозволених адрес;
2. виявлення процесу відсутнього у білому списку із нетиповим ресурсним профілем.

Водночас у межах архітектури модуля перебудови мережі, описаної в підрозділі 3.5, практична реалізація охоплює такі логічні підсистеми:

- підсистему збору фактів на основі Zabbix Agent та *UserParameter*;
- підсистему політик довіри, що містить список дозволених IP адрес, білий список процесів і правила винятків;
- підсистему кореляції та оцінки ризику;
- підсистему прийняття рішень та вибору сценарію перебудови;
- підсистему виконання дій через pfSense API та локальні скрипти;
- журналювання, аудит і постконтроль результатів.

Таким чином, експериментальна частина не є окремим емпіричним сценарієм, а виступає практичним підтвердженням працездатності розробленого теоретичного методу і відповідного програмно-архітектурного модуля.

На етапі підготовки експериментального середовища для кожного контрольованого вузла формувалися базові політики довіри. До них належали:

- глобальний список дозволених IP адрес, що містив шлюзи, DNS/NTP, сервер моніторингу, дозволені зовнішні сервіси та інфраструктурні вузли;
- ролевий список дозволених IP адрес, який визначав допустимі мережеві напрямки залежно від типу вузла (граничний вузол, сервер збору даних, клієнтський Windows-хост, вузол керування);
- вузловий список дозволених IP адрес, що містив точні дозволи для конкретних вузлів;
- білий список програмного забезпечення, який включав перелік процесів, дозволених до виконання на певному вузлі, з можливістю уточнення

імені процесу, шляху, параметрів запуску та допустимого ресурсного профілю;

- винятки та спеціальні режими, зокрема технічне обслуговування, тимчасові дозволи та допустимі аномалії для сервісних операцій.

Усі політики зберігалися у вигляді окремих конфігураційних файлів, що забезпечувало можливість версіонування, аудиту змін та швидкого коригування без модифікації логіки реагування.

Для зменшення кількості хибних спрацювань у системі використовувався ризиковий підхід до класифікації подій. Результатом кореляції був індекс ризику (R), що визначав клас події та глибину втручання:

- (R_0) - нормальний стан, дії не виконуються;
- (R_1) - підозра, виконується логування та посилений моніторинг;
- (R_2) - загроза, застосовується точкове блокування IP, порту або напряму трафіку;
- (R_3) - критичний стан, виконується ізоляція вузла, карантин або повна перебудова мережевої взаємодії.

У практичній частині це дозволило відрізнити локальні мережеві відхилення від комплексних аномалій, пов'язаних із потенційною компрометацією вузла.

Підсистема збору фактів була реалізована за допомогою стандартних метрик Zabbix та розширена зовнішніми механізмами збору даних. Для оцінювання загального стану вузлів використовувалися показники завантаження CPU, пам'яті, файлової системи, *swap*, а також дані щодо окремих процесів. Для аналізу активних з'єднань та кореляції мережевої активності з конкретними процесами використовувалися спеціальні *UserParameter* і PowerShell-скрипти.

У Windows-середовищі PowerShell-скрипт виконував такі дії:

- збирав список активних TCP-з'єднань;
- фільтрував підключення за станом *Established*;
- виділяв віддалені IP-адреси, порти та протоколи;
- за можливості визначав процес-ініціатор та PID;
- формував структуру даних для передачі до Zabbix.

Для контролю процесів окремим скриптом формувався список активних задач із показниками CPU та RAM. Далі у Zabbix ці дані використовувалися в залежних елементах даних та правилах виявлення низького рівня. Це дозволяло не лише збирати факти, а й динамічно створювати об'єкти моніторингу для нових IP-адрес і процесів.

Механізм кореляції був побудований так, щоб рішення не приймалося за одним окремим симптомом. Наприклад, з'єднання з невідомою IP-адресою могло бути класифіковане як (R_1) або (R_2), тоді як одночасне виявлення невідомого процесу, перевищення CPU та нетипового вихідного трафіку підвищувало ризик до (R_3). Саме така багатофакторна кореляція дала можливість перейти від простого порогового виявлення до політико-орієнтованого безпекового керування.

Сценарій 1. Реакція на підключення до зовнішньої IP-адреси поза списком дозволених адрес

Перший сценарій експерименту відповідає алгоритму 1, визначеному в теоретичному підрозділі 2.4, і спрямований на автоматичне виявлення та блокування мережевих з'єднань із зовнішніми адресами, що не відповідають політикам довіри.

У системі було попередньо визначено:

- список дозволених IP адрес для конкретного вузла;
- період опитування активних з'єднань;
- шаблон правила виявлення низького рівня для автоматичного формування елементів даних;
- тригер високої важливості на випадок появи IP відстунього у списку дозволених IP адрес;
- сценарій автоматичної реакції через pfSense API.

Білий список і список дозволених IP адрес, були сформовані таким чином, щоб у штатному режимі вся легітимна мережна активність класифікувалася як *Allowed*. Нова адреса, що не входила до визначених переліків, класифікувалася як *Unknown* або *Denied/Suspicious* залежно від контексту.

У ході експерименту на контрольованому хості ініціювалося TCP-з'єднання з віддаленою IP-адресою, відсутньою у білому списку. Після переходу сесії в стан *Established* адреса з'являлася у таблиці активних мережевих підключень операційної системи. Під час наступного циклу опитування PowerShell-скрипт виявляв новий факт мережевої активності та передавав його у Zabbix.

Далі правило виявлення низького рівня автоматично створював окремий елементи даних для цієї зовнішньої IP-адреси, а також прив'язаний до нього тригер. Якщо адреса не знаходилася в список дозволених IP адрес, значення елемента даних інтерпретувалося як порушення політики доступу, що призводило до активації події.

На рис. 4.14 наведено приклад сформованих елементів даних у Zabbix, які відображають виявлення підозрілих зовнішніх IP-адрес.

☐	Name ▲	Triggers	Key
☐	... Discover suspicious IPs: Suspicious IP: 204.79.197.222	Triggers 1	net.bad.ip[204.79.197.222]

Рис. 4.14. Приклад автоматично сформованих елементів даних Zabbix для контролю зовнішніх IP-адрес, відсутніх у списку дозволених IP адрес.

Після спрацювання тригера система запускала сценарій точкового реагування, який відповідав класу (*R_2*). У межах цього сценарію:

- визначалась IP-адреса порушення;
- формувався запит до pfSense API;
- створювалося правило блокування для вхідного або вихідного трафіку;
- за потреби виконувався *apply changes*;
- факт реагування заносився до журналу аудиту.

Результатом була локальна зміна логічної топології мережі: хоча вузли фізично залишалися в тій самій інфраструктурі, комунікація із забороненою адресою ставала недоступною. Вигляд прикладних правил блокування у pfSense наведено на рис. 4.15.

☐		0/0 B	IPv4 TCP/UDP	204.79.197.222/32	*	*	*	*	none	Blocked by Zabbix auto				
☐		0/0 B	IPv4 TCP/UDP	*	*	204.79.197.222/32	*	*	none	Blocked by Zabbix auto				

Рис. 4.15. Приклад автоматично створених правил блокування в pfSense для зовнішніх IP-адрес, виявлених системою моніторингу.

Експеримент показав, що запропонований механізм коректно:

- виявляє нові підключення до IP відсутніх у списку дозволених IP адрес;
- автоматично створює відповідні об'єкти контролю в Zabbix;
- ініціює виконання сценарію блокування без участі адміністратора;
- забезпечує швидке припинення небажаної мережевої взаємодії;
- зберігає легітимну роботу інших сервісів.

Отже, у межах сценарію 1 було підтверджено ефективність точкового керування логічною топологією мережі через автоматичне обмеження підозрілих напрямків трафіку.

Сценарій 2. Реакція на аномальну поведінку ПЗ відсутнього у білому списку та ізоляція вузла

Другий сценарій відповідає алгоритму 2 і частково алгоритму 3 з підрозділу 2.4. Його метою є виявлення програмної активності, що не відповідає політиці довіри, та автоматичне переведення вузла в ізольований мережевий стан.

Для вузла було визначено:

- перелік дозволених процесів;
- допустимі пороги навантаження CPU та RAM;
- перелік «важких, але дозволених» процесів;
- сценарій реакції у разі виявлення процесу відсутнього у білому списку;
- ізоляційний IP-діапазон або окремий сегмент мережі;
- правила маршрутизації, які переводять вузол у карантинну зону.

Окремо було підготовлено службовий сценарій зміни IP-адреси та оновлення маршрутів, що відповідає алгоритму 3 – зміна IP-адреси ОС.

У межах експерименту на вузлі запускався процес, який:

- був відсутній у білому списку;

- споживав ресурси вище заданого порогу;
- міг ініціювати зовнішню мережеву активність.

PowerShell-скрипт передавав інформацію про процес до Zabbix, після чого за допомогою залежних елементів даних виконувалося:

- зіставлення імені процесу з білим списком;
- оцінка рівня CPU;
- перевірка додаткових ознак, зокрема мережевих з'єднань.

У разі одночасного виконання умов «процес відсутній у білому списку» та «аномальне ресурсне споживання» подія класифікувалася як критична та переводилася до класу (*R_3*). Якщо додатково виявлялися зовнішні з'єднання яке не входить у список дозволених IP адрес, інцидент розглядався як високопріоритетний безпековий сценарій.

На відміну від попереднього сценарію, тут точкового блокування зовнішньої адреси було недостатньо, оскільки загроза розглядалася як така, що походить від самого вузла. Тому система ініціювала глибшу зміну логічної топології, а саме:

- генерувала нову IP-адресу з ізоляційного сегмента;
- створювала або активувала відповідні правила у pfSense;
- змінювала мережеву конфігурацію вузла;
- оновлювала дані в системі моніторингу;
- після успішного переведення вимикала застарілі правила маршрутизації.

У межах цього сценарію вузол втрачав прямий доступ до основного робочого сегмента, але залишався доступним для Zabbix, адміністративного контролю та сервісів аналізу. Таким чином забезпечувалася не повна недоступність, а контрольована ізоляція.

Після переведення вузла до ізольованої зони:

- блокувався доступ до критичних серверів;
- припинялися або обмежувалися підозрілі мережеві взаємодії;
- зберігався канал керування для подальшого аналізу;
- мінімізувався ризик поширення інциденту на інші вузли.

Отримані результати підтвердили, що динамічна зміна IP-адресації та правил маршрутизації є ефективною формою локалізації загроз у розподіленій IoT-інфраструктурі, особливо коли джерело підозрілої активності знаходиться на самому вузлі. Час реакції системи наведено в таблиці 4.4

Таблиця 4.4.

Час реагування розробленого прототипа

Параметри	Зміна IP	Блокування підключення
Час виявлення підозрілої події (Невідоме підключення)	-	5 с
Час виявлення підозрілої події (Нетипова поведінка ПЗ)	15 с	-
Час реагування після виявлення підключення	-	9 с
Час реагування на нетипову поведінку програмного забезпечення	15 с	-
Загальний час реагування на виявлення IP	-	14 с
Загальний час реагування на виявлення нетипових дій ПЗ	30 с	-
Час відновлення доступу	16 с	-

Критично важливою особливістю реалізації стало те, що модуль перебудови мережі не виконував роль «сліпого виконавця». Після кожної автоматичної дії система здійснювала постконтроль результату, зокрема:

- перевіряла, чи правило в pfSense було реально застосоване;
- оцінювала, чи припинився небажаний трафік;
- контролювала, чи вузол після ізоляції залишається доступним для моніторингу;
- фіксувала відповідь API, час застосування та статус операції.

Усі інциденти логувалися із зазначенням:

- часу спрацювання;
- джерела події;

- версії застосованих політик;
- сценарію реагування;
- змінених правил доступу;
- успішності або помилки виконання;
- необхідності відкату.

Такий підхід відповідає вимогам відтворюваності, аудиту та валідації прийнятих рішень, що є особливо важливим у критичних системах IoT.

Для підвищення надійності експериментального механізму та недопущення циклічного блокування/розблокування в систему були закладені спеціальні обмеження. Зокрема використовувалися:

- гістерезис порогів, коли умови входу в аварійний стан і виходу з нього відрізнялися;
- таймаути реакцій, що обмежували тривалість тимчасового блокування;
- ліміти на кількість змін конфігурації за визначений інтервал часу;
- пріоритети подій, що враховували роль вузла в загальній архітектурі;
- спеціальні режими функціонування, зокрема *normal*, *alert*, *incident* і *maintenance*.

Завдяки цьому система не реагувала на одиничні нестійкі коливання та не створювала надмірного навантаження на маршрутизатор або систему моніторингу.

Проведена експериментальна перевірка підтвердила працездатність запропонованого методу автоматичного прийняття рішень у безпекових сценаріях та архітектури модуля перебудови мережі. У ході дослідження було показано, що інтеграція Zabbix, зовнішніх скриптів і pfSense дозволяє реалізувати повний цикл безпекового реагування: від виявлення інциденту до контрольованої зміни логічної топології мережі.

У сценарії з підключенням до зовнішньої адреси за межами списку дозволених IP адрес, система забезпечила автоматичне точкове блокування небажаного трафіку без впливу на штатну роботу інших компонентів. У сценарії з аномальною поведінкою процесу відсутнього у білому списку було підтверджено можливість переведення вузла до ізольованого сегмента зі зміною його IP-

конфігурації та маршрутизації. Це дозволило локалізувати потенційно небезпечний вузол, не припиняючи повністю його контрольованої доступності.

Таким чином, практичні результати узгоджуються з положеннями підрозділу 2.4 і підтверджують, що запропонований метод забезпечує адаптивне, політико-орієнтоване та автоматизоване реагування на інциденти безпеки в розподілених системах IoT. На відміну від традиційного пасивного моніторингу, даний підхід дозволяє не лише фіксувати аномалії, а й змінювати стан мережевої інфраструктури таким чином, щоб зменшити наслідки інциденту та зберегти працездатність системи в цілому.

4.5. Експериментальне дослідження апаратної реалізації нейромережевого обчислювача на ПЛІС

Експериментальна частина роботи передбачала не лише перевірку апаратної реалізації нейронного модуля, але й дослідження алгоритмічних способів реконфігурації обчислювальної структури ПЛІС в умовах змінного режиму обробки даних. На відміну від статичної архітектури, у запропонованому підході обчислювальна логіка ПЛІС змінюється залежно від етапу обробки: спочатку виконується апаратна фільтрація сенсорних сигналів, після чого здійснюється перепрограмування ПЛІС та запуск нейромережевого модуля прогнозування.

Таким чином, експеримент мав дворівневу структуру:

1. Етап апаратної фільтрації сигналів (режим стабілізації даних).
2. Етап нейромережевої інтерпретації (режим інтелектуальної оцінки стану).

Перемикання між цими режимами реалізовано шляхом часткової або повної реконфігурації ПЛІС під керуванням операційної системи Linux.

Експериментальна перевірка виконувалась на платформі DE10-Standard, побудованій на базі SoC Cyclone V, що поєднує двоядерний процесор ARM Cortex-A9 (HPS) та програмовану логіку ПЛІС в єдиному кристалі.

Архітектурно система складалася з таких підсистем:

1. Підсистема збору та попередньої обробки даних, реалізована на стороні Linux (HPS).

2. Апаратне ядро нейронної мережі, інтегроване у ПЛІС як ІР-блок із інтерфейсом Avalon-MM.
3. Механізм взаємодії HPS–ПЛІС, реалізований через пам'яттєво відображений інтерфейс із доступом через */dev/mem* та *mmap*.

Такий підхід дозволив організувати прямий керований обмін без розробки окремого драйвера ядра, що є доцільним на етапі дослідної апробації та швидких ітерацій архітектури.

Було підготовлено два незалежні варіанти конфігурації ПЛІС:

- Конфігурація F_1 - апаратний модуль багаторівневої фільтрації.
- Конфігурація F_2 - нейромережевий обчислювач.

Кожна конфігурація реалізована у вигляді окремого *rbf*-файлу та оптимізована з урахуванням ресурсів ПЛІС.

Експеримент проводився за таким сценарієм:

1. Завантаження конфігурації F_1 .
2. Апаратна фільтрація та стабілізація сенсорних даних.
3. Завершення циклу збору та накопичення вибірки.
4. Автоматична ініціація реконфігурації.
5. Завантаження конфігурації F_2 .
6. Виконання нейромережевого прогнозування на сформованих векторах ознак.

Таким чином, реконфігурація виступала невід'ємною частиною експериментальної процедури.

Алгоритм перемикання режимів базувався на запропонованих алгоритмічних способах реконфігурації обчислювальної структури та включав такі етапи.

Етап 1. Підготовка прошивок

Було сформовано два *rbf*-файли:

- *filter.rbf* - реалізація фільтраційного ядра;
- *nn_module.rbf* - реалізація нейронного модуля.

Файли розміщувалися у файлової системі Linux у визначеному каталозі.

Етап 2. Ініціація реконфігурації

Ініціація здійснювалася:

- автоматично за сценарієм експерименту;
- або через тригер системи моніторингу (Zabbix);
- або за таймером (cron).

Перед початком реконфігурації виконувалося коректне завершення поточних обчислень ПЛІС.

Етап 3. Деактивація мостів HPS–ПЛІС

На цьому етапі тимчасово вимикались канали обміну між Linux та ПЛІС для уникнення конфліктів доступу до пам'яті під час завантаження нової конфігурації.

Етап 4. Завантаження *rbf*-файлу

Обраний *rbf*-файл записувався у пристрій конфігурації ПЛІС через стандартний механізм Linux. Процес займав обмежений час і не вимагав перезавантаження HPS.

Етап 5. Відновлення взаємодії та тестування

Після завершення прошивки:

- відновлювався доступ до мостів;
- перевірялась доступність регістрів;
- виконувалися базові функціональні тести (читання/запис, контроль статусу).

Таблиця 4.5.

Параметри переконфігурації ПЛІС

Параметр	Модуль фільтрації	Нейромережевий обчислювач
Час розгортання модуля та перевірка коректності його роботи	199.92 мс	199.789 мс
Розмір модуля	2 310 324 байт	2 272 155 байт

Час розгортання модулів з тестуванням та їх об'єм наведено в таблиці 4.5 та об'єм використаних ресурсів ПЛІС для модулів наведені в таблиці 4.6

Таблиця 4.6

Використані апаратні ресурси для різних модулів

Параметри	Конфігурація апаратний модуль фільтрації	Конфігурація нейромережевого обчислювача
Логічні елементи	3288	3957
Кількість регістрів	4020	3655
Блоки цифрової обробки сигналів	0	87

Наступна частина експерименту базувалась на запропонованому способі поетапного формування інформаційного простору ознак, який узгоджується з особливостями апаратної реалізації на ПЛІС.

1. Формування первинного простору спостережень
2. Апаратний рівень фільтрації
3. Агрегування у часові вектори

Кожен навчальний приклад формувався як 11-вимірний вектор ознак, що описував усереднений стан середовища за певний часовий інтервал. Вектор включав як абсолютні параметри, так і похідні характеристики та відхилення від комфортних значень.

Таким чином, простір ознак набував семантичної інтерпретації - він відображав не миттєвий стан системи, а інтегральну характеристику її функціонування.

Цільова змінна формувалась як інтегральна оцінка якості середовища, що розглядалася як латентний параметр системи. Вона не може бути безпосередньо виміряна, але може бути апроксимована на основі множини сенсорних параметрів.

Цільова змінна для кожного прикладу формувалася у вигляді нормалізованого показника в інтервалі $[0;1]$, який відображав ступінь відповідності сукупного стану середовища встановленим критеріям. Значення, близькі до одиниці, відповідали станам, що перебувають у межах допустимого режиму, тоді

як значення, близькі до нуля, характеризували істотне відхилення від еталонних меж. Такий спосіб задання міток дозволив перейти від дискретного опису станів до неперервної моделі оцінювання, що краще узгоджується з фізичною природою змін параметрів середовища.

Навчання нейронної мережі здійснювалось у середовищі PyTorch [121, 122] із використанням архітектури:

Input(11) → Linear(8) → ReLU → Linear(4) → ReLU → Linear(1) → Sigmoid.

Процес навчання включав:

- нормалізацію ознак до заданого діапазону;
- мінімізацію функції втрат;
- контроль перенавчання на валідаційній вибірці;
- збереження параметрів моделі після досягнення стабілізації функції втрат.

Отримані вагові коефіцієнти та зміщення були експортовані у зовнішній формат для подальшого квантування.

Апаратна реалізація вимагала переходу від арифметики з плаваючою комою до арифметики з фіксованою комою представлення. Процедура включала:

1. Масштабування вагових коефіцієнтів.
2. Квантування до узгодженого формату.
3. Перевірку збереження числової стабільності.
4. Імпорт квантизованих параметрів у ROM-блоки ПЛІС.

Особлива увага приділялась узгодженню масштабування вхідних ознак і ваг, що забезпечувало мінімізацію похибки апаратної апроксимації.

Для оцінювання точності було сформовано тестовий набір векторів, які не використовувались під час навчання.

Для кожного тестового прикладу виконувались:

1. Обчислення еталонного значення у програмному середовищі.
2. Передача того самого вектора до ПЛІС-модуля.
3. Зчитування результату обробки.
4. Оцінка похибки обчислень при переході від плаваючої до фіксованої коми.

Оцінювались такі метрики:

- середня абсолютна похибка;
- максимальне відхилення;
- стабільність знаку та монотонності результату.

Отримані результати засвідчили збереження функціональної відповідності моделі після квантування.

Час обробки одного вхідного вектора визначався як інтервал між моментом завершення завантаження ознак та встановленням прапорця готовності. Результати вимірів наведені в таблиці 4.7 та розподіл часу обробки лише на центральному процесорі наведений на рис. 4.16.

Таблиця 4.7.

Результати тестування швидкості обробки вхідного вектора

Параметр	ПЛС(De1-SoC)	ЦП(i5-3550)
Середній час обробки 20 тисяч векторів	0.88 с	2.7336 с
Час обробки одного вектора	0. 044 мс	0.13668 мс
Кількість оброблених векторів за секунду	22727 шт	7316 шт

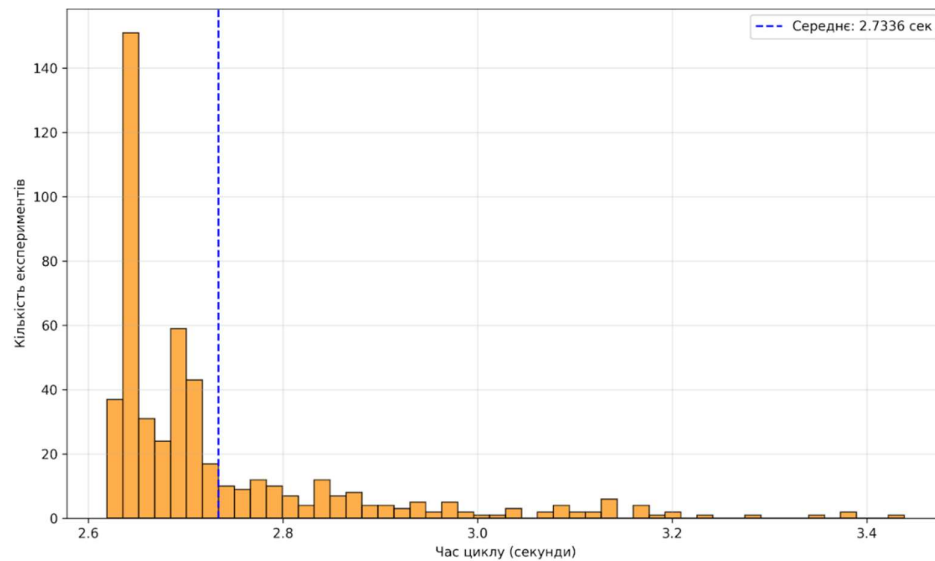


Рис. 4.16 Гістограма розподілу часу обробки 20 тисяч векторів на центральному процесорі.

Було встановлено, що:

- час виконання є детермінованим;
- відсутня залежність від завантаження процесора;
- обробка кожного вектора відбувається за однакову кількість тактів.

Це є критично важливим для систем реального часу, де прогнозованість латентності має принципове значення.

Фінальний етап обчислення реалізовано через LUT-апроксимацію нелінійності. Результати показали, що табличний метод забезпечує достатню точність для задач оцінювання стану середовища при мінімальних ресурсних витратах.

Отриманий вихідний показник у діапазоні (0;1) використовувався як інтегральний індикатор у загальній IoT-архітектурі як індикатор відповідності робочого середовища визначеним нормам.

Отже, експериментальне дослідження підтвердило не лише працездатність апаратної реалізації нейронного модуля, але й коректність усієї процедури формування датасету, починаючи від етапу стабілізації сенсорних вимірювань і завершуючи побудовою нормалізованої цільової оцінки. Поєднання попередньої апаратної фільтрації, нормативно-орієнтованого формування ознак, програмного навчання та подальшого перенесення моделі у ПЛІС забезпечило можливість реалізації детермінованого модуля оцінювання стану середовища, придатного для використання в системах моніторингу та автоматизованого керування.

Таким чином, нейромережевий обчислювач функціонує як детермінований апаратний оператор прийняття рішень на граничному рівні.

ВИСНОВОК ДО РОЗДІЛУ 4

У четвертому розділі наведено результати експериментальної перевірки розроблених у попередніх розділах моделей, методів та архітектурних рішень для інтелектуальної обробки даних у розподілених системах IoT. Проведені дослідження підтвердили працездатність запропонованої гібридної тришарової архітектури, у якій поєднуються апаратна обробка на SoC/ПЛІС, програмні механізми моніторингу, адаптивне прийняття рішень і засоби автоматизованого керування станом обчислювальної та мережевої інфраструктури.

Було сформовано методику експериментальних досліджень, що охоплює послідовну перевірку як окремих компонентів системи, так і їх спільної інтегрованої роботи. Показано, що побудована експериментальна платформа відповідає логіці замкненого контуру керування *«збір даних → аналіз і зіставлення з політиками → прийняття рішення → виконання керуючого впливу → інформування та аудит»*, а обрані критерії оцінювання дозволяють кількісно аналізувати продуктивність, енергоефективність, стабільність та адаптивність системи.

Підтверджено ефективність запропонованого способу багаторівневої фільтрації сенсорних даних. Верифікація у синтетичному середовищі ModelSim показала коректність послідовної роботи медіанного фільтра, ковзного середнього та експоненціального згладжування. Подальші експерименти на реальних даних із датчиків температури та вологості продемонстрували, що застосування тривірневої фільтрації дозволяє суттєво зменшити варіативність сигналів, приглушити імпульсні викиди, знизити амплітуду випадкових коливань та зберегти довгострокові тренди. Найбільший позитивний ефект зафіксовано для сенсорів із підвищеним рівнем нестабільності, зокрема DHT11, для якого стандартне відхилення температурного сигналу зменшилося більш ніж на 53%, а також для сигналів вологості з вираженою шумовою складовою. Для більш стабільних сенсорів фільтрація не змінювала середнього значення, але покращувала інтерпретацію часових рядів та знижувала імовірність хибних тригерних подій.

Окремий блок досліджень був присвячений оцінюванню обчислювальної ефективності апаратної реалізації фільтрації на ПЛІС. Отримані результати показали, що апаратний модуль забезпечує детерміновану часову поведінку, конвеєрний режим обробки та вищу пропускну здатність порівняно з програмною реалізацією на HPS-процесорі. За результатами обробки тестового набору обсягом 5 млн вимірів використання ПЛІС дозволило скоротити час обробки та зменшити енергетичні витрати, а інтегральний коефіцієнт ефективності виявився вищим, ніж для реалізації на CPU. Це підтверджує доцільність перенесення етапу первинної обробки даних на ПЛІС в умовах граничних вузлів із обмеженими ресурсами та потребою у гарантованій швидкодії.

Експериментально досліджено механізм балансування навантаження у тришаровій архітектурі IoT. Показано, що використання інтегрального показника завантаженості вузла, який враховує стан CPU, оперативної пам'яті та дискового простору, дозволяє своєчасно виявляти перевантаження та автоматично ініціювати перенесення джерела даних на менш завантажений вузол. У сценарії контрольованого перевантаження Main Server система коректно зафіксувала перевищення порогового значення, після чого виконала міграцію задачі на Reserve Server. Отримані графіки підтвердили зниження навантаження на основному вузлі, відповідне зростання навантаження на резервному та подальшу стабілізацію обох серверів у допустимих межах. Це доводить ефективність розробленого підходу для підтримання працездатності розподіленої системи IoT в умовах нерівномірного навантаження.

Підтверджено працездатність методу динамічної зміни логічної топології мережі на основі подій моніторингу безпеки. Практична реалізація на базі Zabbix, зовнішніх *PowerShell*- і *shell*-скриптів та шлюзу pfSense показала можливість автоматизованого переходу від виявлення підозрілої активності до активного керуючого впливу на мережеву інфраструктуру. У сценарії виявлення з'єднання з IP-адресою яка не входить у список дозволених IP адрес, система забезпечила автоматичне створення елементів контролю, формування тригера та блокування небажаного трафіку через pfSense. У сценарії виявлення процесу відстунього у

білому списку з аномальним ресурсним профілем було реалізовано більш глибоку реакцію у вигляді ізоляції вузла, зміни IP-конфігурації та перебудови правил маршрутизації. Отримані результати показали, що запропонований механізм дозволяє локалізувати інцидент без повного виведення вузла з експлуатації, зберігаючи його доступність для моніторингу та аналізу. Таким чином, експерименти підтвердили можливість практичної реалізації політико-орієнтованого автоматизованого реагування на загрози у розподілених IoT-системах.

Досліджено апаратну реалізацію нейронного модуля на SoC/ПЛІС та експериментально перевірено спосіб динамічної реконфігурації ПЛІС. Показано, що одна й та сама апаратна платформа може послідовно використовуватися для двох різних режимів: апаратної фільтрації сигналів і нейромережевого модуля обробки даних. Запропонований алгоритм реконфігурації, який включає підготовку *rbf*-файлів, ініціацію перемикавання, деактивацію мостів HPS–ПЛІС, завантаження нової конфігурації та подальше тестування працездатності, забезпечив коректне та безпечне перемикавання між режимами без перезавантаження всієї системи. Дослідження нейронного модуля підтвердило можливість використання одинадцяти-вимірний простору ознак для апаратної інтегральної оцінки стану середовища, а також показало, що після квантування вагових коефіцієнтів і переходу до реалізації з фіксованою комою зберігається функціональна відповідність моделі. Апаратний модуль прогнозування продемонстрував детермінований час обробки вхідних векторів, незалежність від поточного завантаження процесора та придатність до роботи в системах реального часу.

Загалом результати розділу підтверджують, що запропоновані методи та засоби інтелектуальної обробки даних в мережах IoT забезпечують комплексне підвищення ефективності функціонування системи. Використання багаторівневої апаратної фільтрації підвищує якість первинних даних, ПЛІС прискорення зменшує навантаження на процесор і підвищує енергоефективність, інтегральний показник завантаження дозволяє реалізувати адаптивне балансування ресурсів, а

механізми автоматизованої перебудови мережі підвищують стійкість системи до інцидентів безпеки. Додатково динамічна реконфігурація ПЛІС та апаратний нейромережевий обчислювач розширюють функціональні можливості граничного рівня, забезпечуючи не лише локальну обробку, а й детерміноване прийняття рішень безпосередньо на периферії мережі.

Отже, експериментальні дослідження повністю підтвердили працездатність, узгодженість і практичну цінність запропонованої архітектури та довели, що її використання дозволяє реалізувати масштабовану, енергоефективну, детерміновану та стійку до збоїв граничну платформу для мереж Інтернету речей.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-практичну задачу підвищення ефективності обробки потоків даних та функціонування інфраструктури на граничному рівні розподілених систем Інтернету речей (IoT), що функціонують в режимі реального часу та за умов надходження великої кількості різнорідних сенсорних даних, багатопоточності обробки та масштабування інфраструктури. Основні результати дослідження полягають у такому.

1. Виконано аналіз сучасних архітектур систем IoT, методів обробки поточкових сенсорних даних, підходів до підвищення ефективності функціонування інфраструктури. Встановлено, що традиційні підходи до обробки даних є недостатньо ефективними для функціонування систем такого класу в режимі реального часу, що обумовило необхідність розробки системного підходу до адаптивної інтелектуальної обробки даних на граничному рівні розподілених систем IoT, що ґрунтується на формуванні адаптивно-інтелектуально керованої гібридної інфраструктури IoT, у межах якої процеси обробки потоків даних, моніторингу даних та стану ресурсів, інтелектуальної аналітики та забезпечення ефективності інфраструктури інтегровані в єдиний керуючий контур.

2. Запропоновано новий комплексний метод адаптивної інтелектуальної обробки даних у розподілених системах IoT, який відрізняється від відомих тим, що інтегрує сукупність взаємопов'язаних часткових методів і засобів в єдиний керуючий механізм, поєднуючи процеси багаторівневої фільтрації сенсорних потоків, моніторингу стану обчислювальних ресурсів, балансування навантаження та інтелектуальної аналітики. Запропонований метод забезпечив можливість динамічного керування потоками обробки, реконфігурацію обчислювальних вузлів, спрощення масштабування граничної інфраструктури системи IoT, що в цілому дозволило підвищити якість сенсорних вимірювань, забезпечити оперативність керуючих впливів, підвищити ефективність використання обчислювальних ресурсів, безпеку функціонування інфраструктури.

3. Удосконалено спосіб адаптивного налаштування порогів контролю у моніторингових системах, який на відміну від відомих підходів із фіксованими значеннями, поєднує механізми вирівнювання, корекції сигналів та виділення тренду з адаптивним самоналаштуванням порогів контролю. Це дозволило забезпечити їх автоматичну корекцію залежно від змін стану середовища та режимів функціонування системи, за рахунок чого в цілому було підвищено чутливість до аномалій, зокрема зменшено кількість хибних спрацювань.

Експериментально підтверджено ефективність використання апаратного обчислювача на гетерогенній платформі SoC/ПЛІС Cyclone V для обробки кількох потоків сенсорних даних отриманих із реальних джерел (температура, вологість, освітленість, CO₂). Результати експериментів показали ефективність очищення та стабілізації сенсорних даних в межах від 0.32% до 53.06%.

4. Запропоновано спосіб динамічної реконфігурації обчислювальних вузлів на гетерогенних платформах SoC/ПЛІС, який на відміну від відомих, дозволив забезпечити ефективну інтеграцію реконфігуровної логіки з вбудованим ядром ARM, з точки зору ефективного узгодження обчислювальних задач між апаратною і програмною складовими системи та динамічного перепрошивання ПЛІС, залежно від навантаження та класу вирішуваних задач. Це дозволило підвищити швидкодію обробки даних та ефективність використання апаратних ресурсів, зокрема отримано збільшення продуктивності обчислень на **13,01%** та зменшення енергоспоживання на **12,5 %** порівняно з використанням процесорів загального призначення без апаратного прискорення.

5. Отримав подальшого розвитку метод автоматичного прийняття рішень у безпекових сценаріях розподілених системах IoT, який відрізняється від відомих інтеграцією трьох компонентів: безперервного моніторингу стану системи та мережевої взаємодії з виконанням формалізованих алгоритмів аналізу аномальних подій і автоматизованим виконанням керуючих впливів. Реалізація запропонованого методу дозволила підвищити стійкість і кібербезпеку інфраструктури IoT та зменшити залежність від централізованих сервісів. За результатами експериментів час виявлення та реакції на подію становить від 14 до

30 секунд, що є важливим результатом порівняно з ручним опрацюванням подій, яке, за приблизними експертними оцінками, становить 1–3 хвилини.

6. Запропоновано спосіб інтелектуальної нейромережевої обробки сенсорних даних на ПЛІС, який відрізняється від відомих поєднанням апаратної стабілізації даних із нейромережовим прийняттям рішень у межах єдиного обчислювального контуру. Реалізація нейронного модуля на ПЛІС забезпечила можливість виконання аналізу сенсорних потоків у режимі реального часу при обмежених обчислювальних ресурсах граничних вузлів. Кількісна оцінка продуктивності на підставі виконаних експериментів показала, що обробка 20 тис. вхідних векторів нейромережовим обчислювачем забезпечила зменшення часу обробки приблизно на **67,8 %** порівняно з процесорною реалізацією.

7. Практичне значення отриманих результатів полягає у створенні програмно-апаратної платформи адаптивної інтелектуальної обробки сенсорних даних у системах Інтернету речей, яка включає комплекс взаємопов'язаних компонентів: апаратний багаторівневий модуль фільтрації на базі ПЛІС для попередньої обробки, аналізу та формування трендів даних; нейромережовий модуль для інтелектуального аналізу сенсорних сигналів і підтримки прийняття рішень; підсистему моніторингу та керування інфраструктурою на основі системи Zabbix, що забезпечує контроль стану ресурсів, балансування обчислювального навантаження та автоматичну перебудову мережі.

Розроблена платформа орієнтована на використання у промислових і високотехнологічних системах, включаючи підприємства, виробництва, критичні технологічні об'єкти, а також у системи промислового IoT та розумних містах, що потребують обробки великих потоків даних у режимі реального часу.

**Заява про використання генеративного ШІ та технологій на основі ШІ в
процесі написання тексту статті**

Під час написання цього матеріалу автор використовував модель генеративного штучного інтелекту ChatGPT для перевірки правопису та оформлення списку літератури. Після використання ChatGPT автор переглянув та відредагував зміст і взяв на себе повну відповідальність за зміст роботи.

ЖИТЕПАТҮПА

1. Hasan B. T., Idrees A. K. Edge Computing for IoT. Learning Techniques for the Internet of Things / ed. by P. K. Donta, A. Hazra, L. Lovén. Cham : Springer, 2024. URL: https://doi.org/10.1007/978-3-031-50514-0_1 (date of access: 10.03.2026)
2. Munir A., Kansakar P., Khan S. U. IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things. IEEE Consumer Electronics Magazine. 2017. Vol. 6, no. 3. P. 74–82. URL: <https://doi.org/10.1109/MCE.2017.2684981> (date of access: 10.03.2026).
3. Hashem I. A. T., Yaqoob I., Anuar N. B., Mokhtar S., Gani A., Khan S. U. The rise of “big data” on cloud computing: Review and open research issues. Information Systems. 2015. Vol. 47. P. 98–115. URL: <https://doi.org/10.1016/j.is.2014.07.006> (date of access: 10.03.2026).
4. Al-Fuqaha A., Guizani M., Mohammadi M., Aledhari M., Ayyash M. Internet of Things: A survey on enabling technologies, protocols, and applications. IEEE Communications Surveys & Tutorials. 2015. vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015, URL: <https://doi.org/10.1109/COMST.2015.2444095> (date of access: 10.03.2026).
5. Shi W., Dustdar S. The promise of edge computing. Computer. 2016. vol. 49, no. 5, pp. 78-81 URL: <https://doi.org/10.1109/MC.2016.145> (date of access: 10.03.2026).
6. Yousefpour A., Fung C., Nguyen T., Kadiyala K. et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture. 2019. Vol. 98. P. 289–330. URL: <https://doi.org/10.1016/j.sysarc.2019.02.009> (date of access: 10.03.2026)
7. Satyanarayanan M. The emergence of edge computing. Computer. 2017. Vol. 50, no. 1. P. 30–39. URL: <https://doi.org/10.1109/MC.2017.9> (date of access: 10.03.2026).
8. Saleem K., Bajwa I. S., Sarwar N., Anwar W., Ashraf A. IoT Healthcare: Design of Smart and Cost-Effective Sleep Quality Monitoring System. Journal of Sensors. 2020. Vol. 2020. Article ID 8882378. URL: <https://doi.org/10.1155/2020/8882378> (date of access: 10.03.2026).

9. Binh N. T., Kieu-Do B., Hoang T.-T., Cong-Kha P., Pham-Quoc C. FPGA-Based Secured and Efficient Lightweight IoT Edge Devices with Customized RISC-V. 2023 RIVF International Conference on Computing and Communication Technologies (RIVF), Hanoi, Vietnam, 2023. P. 31–36. URL: <https://doi.org/10.1109/RIVF60135.2023.10471777> (date of access: 10.03.2026).
10. Nguyen L. A., Kiet P. T., Lee S., Yeo H., Son Y. Comprehensive Survey of Sensor Data Verification in Internet of Things. IEEE Access. 2023. Vol. 11. P. 50560–50577. URL: <https://doi.org/10.1109/ACCESS.2023.3277545> (date of access: 10.03.2026).
11. Buelvas J., Múnera D., Tobón V. D. P. et al. Data Quality in IoT-Based Air Quality Monitoring Systems: a Systematic Mapping Study. Water, Air, & Soil Pollution. 2023. Vol. 234. P. 248. URL: <https://doi.org/10.1007/s11270-023-06127-9> (date of access: 10.03.2026).
12. Zhang L., Jeong D., Lee S. Data Quality Management in the Internet of Things. Sensors. 2021. Vol. 21. P. 5834. URL: <https://doi.org/10.3390/s21175834> (date of access: 10.03.2026).
13. Kulanov V., Perepelitsyn A. Analysis of application of FPGA technologies in IoT. Aerospace Technic and Technology. 2024. No. 5. P. 85–94. URL: <https://doi.org/10.32620/aktt.2024.5.09> (date of access: 10.03.2026).
14. Magyari A., Chen Y. Review of state-of-the-art FPGA applications in IoT networks. Sensors. 2022. Vol. 22. Art. no. 7496. URL: <https://doi.org/10.3390/s22197496> (date of access: 10.03.2026).
15. Satpathy S., Trivedi M. C., Malhotra S., Chaudhary S., Sharma V. FPGA-enhanced IoT methods for disease pre-screening and prediction: An energy optimization approach. Measurement: Sensors. 2024. Vol. 33. Art. no. 101213. URL: <https://doi.org/10.1016/j.measen.2024.101213> (date of access: 10.03.2026).
16. Seng K. P., Lee P. J., Ang L. M. Embedded Intelligence on FPGA: Survey, Applications and Challenges. Electronics. 2021. Vol. 10. P. 895. URL: <https://doi.org/10.3390/electronics10080895> (date of access: 10.03.2026).

17. Hryshchuk O., Zagorodnyuk S. Managing energy consumption in FPGA-based edge computing systems with soft-core CPUs. *Journal of Edge Computing*. 2025. Vol. 4, no. 1. P. 57–72. URL: <https://doi.org/10.55056/jec.717> (date of access: 10.03.2026).
18. Gubbi J., Buyya R., Marusic S., Palaniswami M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*. 2013. Vol. 29, no. 7, P. 1645-1660.
URL: <https://doi.org/10.1016/j.future.2013.01.010> (date of access: 10.03.2026).
19. Atzori L., Iera A., Morabito G. The Internet of Things: A survey. *Computer Networks*. 2010. Vol. 54, no. 15, P. 2787-2805.
URL: <https://doi.org/10.1016/j.comnet.2010.05.010> (date of access: 10.03.2026).
20. Sethi P., Sarangi S. R. Internet of Things: Architectures, Protocols, and Applications. *Journal of Electrical and Computer Engineering*. 2017. Article ID 9324035. 25 p. URL: <https://doi.org/10.1155/2017/9324035> (date of access: 10.03.2026).
21. Bonomi F., Milito R., Zhu J., Addepalli S. Fog computing and its role in the Internet of Things. *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing (MCC)*. 2012. P. 13–16. URL: <https://doi.org/10.1145/2342509.2342513> (date of access: 10.03.2026).
22. Atlam H., Walters R., Wills G. Fog Computing and the Internet of Things: A Review. *Big Data and Cognitive Computing*. 2018. Vol. 2, no. 2. P. 10. URL: <https://doi.org/10.3390/bdcc2020010> (date of access: 10.03.2026).
23. Hong K., Lillethun D., Ramachandran U., Ottenwälder B., Koldehofe B. Mobile fog: a programming model for large-scale applications on the Internet of Things. *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC '13)*. New York : Association for Computing Machinery, 2013. P.15–20.
URL: <https://doi.org/10.1145/2491266.2491270> (date of access: 10.03.2026).
24. Abel E. Edje, M.S. Abd Latiff, Weng Howe Chan. IoT data analytic algorithms on edge-cloud infrastructure: A review. *Digital Communications and Networks*. 2023. vol. 9, no. 6, pp. 1486-1515 URL: <https://doi.org/10.1016/j.dcan.2023.10.002> (date of access: 10.03.2026).

25. PunithaIlayarani P., Dominic M. Anatomization of Fog Computing and Edge Computing. 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, 20–22 February 2019. 2019. URL: <https://doi.org/10.1109/ICECCT.2019.8869125> (date of access: 10.03.2026)
26. Hamdan S., Ayyash M., Almajali S. Edge-Computing Architectures for Internet of Things Applications: A Survey. *Sensors (Basel)*. 2020. No. 20(22). URL: <https://doi.org/10.3390/s20226441> (date of access: 10.03.2026)
27. Ogino T., Kitagami S., Shiratori N. A Multi-agent Based Flexible IoT Edge Computing Architecture and Application to ITS. *Journal of Communications*. 2019. P. 47–52. URL: <https://doi.org/10.12720/jcm.14.1.47-52> (date of access: 10.03.2026)
28. Dauda A., Flauzac O., Nolot F. A Survey on IoT Application Architectures. *Sensors*. 2024. Vol. 24. P. 5320. URL: <https://doi.org/10.3390/s24165320> (date of access: 10.03.2026)
29. Jin W., Xu R., You T., Hong Y.-G., Kim D. Secure edge computing management based on independent microservices providers for gateway-centric IoT networks. *IEEE Access*. 2020. Vol. 8. P. 187975–187990. URL: <https://doi.org/10.1109/ACCESS.2020.3030297> (date of access: 10.03.2026)
30. Shi W., Cao J., Zhang Q., Li Y., Xu L. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*. 2016. Vol. 3, no. 5. P. 637–646. URL: <https://doi.org/10.1109/IIOT.2016.2579198> (date of access: 10.03.2026)
31. Varghese B., Buyya R. Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*. 2017. Vol. 79, no. 3. P. 849–861. URL: <https://doi.org/10.1016/j.future.2017.09.020> (date of access: 10.03.2026)
32. Gupta A. Network Monitoring: Network Monitoring Basics. [Електронний ресурс] <https://www.motadata.com/blog/network-monitoring-basics/> (date of access: 10.03.2026)
33. Каплунов А., Гайдай А., Гер В., Нікольський С. “Засоби моніторингу мережі в іот інфраструктурі з гібридною архітектурою.” *Телекомунікаційні та*

- інформаційні технології № 2, с. 22-32, 2023 URL: <https://doi.org/10.31673/2412-4338.2022.022332> (date of access: 10.03.2026)
34. Mishra S. Cloud of Things (CoT): Security, Privacy & Adoption. International Journal of Security and Its Applications. 2020. Vol. 14, no. 3. P. 1–14. URL: https://www.researchgate.net/publication/344476971_CLOUD_OF_THINGS_COT_SECURITY_PRIVACY_ADOPTION (date of access: 10.03.2026)
 35. Andriulo, F.C.; Fiore, M.; Mongiello, M.; Traversa, E.; Zizzo, V. Edge Computing and Cloud Computing for Internet of Things: A Review. Informatics 2024, 11, 71. URL: <https://doi.org/10.3390/informatics11040071> (date of access: 10.03.2026)
 36. Wilson M. Best Monitoring Software & Tools for Bandwidth & Traffic Analysis [Електронний ресурс] – PCWDL.com, 2023 <https://www.pcwld.com/router-monitoring-software/#wbounce-modal> (date of access: 10.03.2026)
 37. Ramachandra G., Iftikhar M., Khan F. A. A Comprehensive Survey on Security in Cloud Computing. Procedia Computer Science. 2017. Vol. 3, issue 11. P. 465–472. URL: <https://doi.org/10.1016/j.procs.2017.06.124> (date of access: 10.03.2026).
 38. Kaur C. The Cloud Computing and Internet of Things (IoT). International Journal of Scientific Research in Science, Engineering and Technology. 2020. Vol. 7, issue 1. URL: <https://doi.org/10.32628/IJSRSET196657> (date of access: 10.03.2026).
 39. Rogier B. How to measure network performance metrics via passive traffic analysis? [Електронний ресурс]. accedian.com. 2023. URL: <https://accedian.com/blog/measure-network-performance-metrics-passive-traffic-analysis/>
 40. Lamberti A. Network Metrics: How to Measure Network Performance. [Електронний ресурс]. obkio.com. 2023. URL: <https://obkio.com/blog/how-to-measure-network-performance-metrics/>
 41. Zhou Z., Chen X., Li E., Zeng L., Luo K., Zhang J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE. 2019. Vol. 107, no. 8. P. 1738–1762. URL: <https://doi.org/10.48550/arXiv.1905.10083> (date of access: 10.03.2026).

- 42.Chen M., Mao S., Liu Y. Big data: A survey. *Mobile Networks and Applications*. 2014. Vol. 19, no. 2. P.171–209. URL: <https://doi.org/10.1007/s11036-013-0489-0> (date of access: 10.03.2026).
- 43.Roman R., Lopez J., Mambo M. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*. 2018. Vol. 78. P. 680–698. URL: <https://doi.org/10.1016/j.future.2016.11.009> (date of access: 10.03.2026).
- 44.Kitchin R. Big Data, new epistemologies and paradigm shifts. *Big Data & Society*. 2014. Vol. 1, no. 1. URL: <https://doi.org/10.1177/2053951714528481> (date of access: 10.03.2026).
- 45.Compton M. et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*. 2012. Vol. 17. P. 25–32. URL: <https://doi.org/10.1016/j.websem.2012.05.003> (date of access: 10.03.2026).
- 46.Perera C., Zaslavsky A., Christen P., Georgakopoulos D. Context aware computing for the Internet of Things: A survey. *IEEE Communications Surveys & Tutorials*. 2014. Vol. 16, no. 1. P. 414–454. URL: <https://doi.org/10.1109/SURV.2013.042313.00197> (date of access: 10.03.2026).
- 47.Gavriilidis N. O., Halkidis S. T., Petridou S. Empirical Evaluation of TLS-Enhanced MQTT on IoT Devices for V2X Use Cases. *Applied Sciences*. 2025. Vol. 15. P. 8398. URL: <https://doi.org/10.3390/app15158398> (date of access: 10.03.2026).
- 48.Solving the complexity of communicating between IoT devices and the enterprise. [Електронний ресурс]. URL: <https://content.u-blox.com/sites/default/files/documents/low-power-mqtt-sn-white-paper.pdf> (date of access: 10.03.2026).
- 49.Haidai A., Klymenko I. “Method of load balancing in distributed three-layer IoT architecture.” *Information, Computing and Intelligent systems* 4, 60-68, 2024 DOI: URL: <https://doi.org/10.20535/2786-8729.4.2024.311595>(date of access: 10.03.2026).
- 50.Iftikhar S., Gill S. S., Song C., Xu M., Aslanpour M. S., Toosi A. N., Du J., Wu H., Ghosh S., Chowdhury D., Golec M., Kumar M., Abdelmoniem A. M., Cuadrado F.,

- Varghese B., Rana O., Dustdar S., Uhlig S. AI-based fog and edge computing: A systematic review, taxonomy and future directions. *Internet of Things*. 2023. Vol. 21. 100674. ISSN 2542-6605. URL: <https://doi.org/10.1016/j.iot.2022.100674> (date of access: 10.03.2026).
51. Tahmasebi-Pouya N., Sarram M. A., Mostafavi S. A reinforcement learning-based load balancing algorithm for fog computing. *Telecommunication Systems*. 2023. Vol. 84. P. 321–339. URL: <https://doi.org/10.1007/s11235-023-01049-7> (date of access: 10.03.2026).
52. Taherizadeh S., Stankovski V., Grobelsnik M. A Capillary Computing Architecture for Dynamic Internet of Things: Orchestration of Microservices from Edge Devices to Fog and Cloud Providers. *Sensors*. 2018. Vol. 18. P. 2938. URL: <https://doi.org/10.3390/s18092938> (date of access: 10.03.2026).
53. Ahmad M. O., Khan R. Z. Load Balancing Tools and Techniques in Cloud Computing: A Systematic Review. *Advances in Computer and Computational Sciences / ed. by S. Bhatia, K. Mishra, S. Tiwari, V. Singh*. Singapore : Springer, 2018. (Advances in Intelligent Systems and Computing ; vol. 554). URL: https://doi.org/10.1007/978-981-10-3773-3_18 (date of access: 10.03.2026).
54. Randles M., Lamb D., Taleb-Bendiab A. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing. 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, Perth, WA, Australia, 2010. P. 551–556. URL: <https://doi.org/10.1109/WAINA.2010.85> (date of access: 10.03.2026).
55. Shuaib M., Bhatia S., Alam S., Masih R. K., Alqahtani N., Basheer S., Alam M. S. An Optimized, Dynamic, and Efficient Load-Balancing Framework for Resource Management in the Internet of Things (IoT) Environment. *Electronics*. 2023. Vol. 12, no. 5. P. 1104. URL: <https://doi.org/10.3390/electronics12051104> (date of access: 10.03.2026).
56. Shivaraman N., Fittler J., Ramanathan S., Easwaran A., Steinhorst S. WiP Abstract: Mobility-based Load Balancing for IoT-enabled Devices in Smart Grids. 2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS),

- Sydney, NSW, Australia, 2020. P. 192–193. URL: <https://doi.org/10.1109/ICCPS48487.2020.00029> (date of access: 10.03.2026).
57. Tseng H. Multipath Load Balancing Routing for Internet of Things. *Journal of Sensors*. 2016. Vol. 2016. P. 1–8. URL: <https://doi.org/10.1155/2016/4250746> (date of access: 10.03.2026).
58. Kanellopoulos D., Sharma V. K. Dynamic Load Balancing Techniques in the IoT: A Review. *Symmetry*. 2022. Vol. 14, no. 12. P. 2554. URL: <https://doi.org/10.3390/sym14122554> (date of access: 10.03.2026).
59. Devi K., Sumathi D., Venkataraman V., Chunduru A., Kataraki K., Balakrishnan S. CLOUD load balancing for storing the internet of things using deep load balancer with enhanced security. *Measurement: Sensors*. 2023. Vol. 28. P. 100818. URL: <https://doi.org/10.1016/j.measen.2023.100818> (date of access: 10.03.2026).
60. Eghbali Z., Zolfy Lighvan M. A hierarchical approach for accelerating IoT data management process based on SDN principles. *Journal of Network and Computer Applications*. 2021. Vol. 181. P. 103027. URL: <https://doi.org/10.1016/j.jnca.2021.103027> (date of access: 10.03.2026).
61. Hajian E., Khayyambashi M. R., Movahhedinia N. A Mechanism for Load Balancing Routing and Virtualization Based on SDWSN for IoT Applications. *IEEE Access*. 2022. Vol. 10. P. 37457–37476. URL: <https://doi.org/10.1109/ACCESS.2022.3164693> (date of access: 10.03.2026).
62. Belgaum M. R., Musa S., Alam M. M., Su'ud M. M. A Systematic Review of Load Balancing Techniques in Software-Defined Networking. *IEEE Access*. 2020. Vol. 8. P. 98612–98636. URL: <https://doi.org/10.1109/ACCESS.2020.2995849> (date of access: 10.03.2026).
63. Alhilali A., Montazerolghaem A. Artificial intelligence based load balancing in SDN: A comprehensive survey. [Электронный ресурс]. 2023. URL: <https://doi.org/10.48550/arXiv.2308.02149> (date of access: 10.03.2026).
64. Gupta E., Deshpande V. A Technique Based on Ant Colony Optimization for Load Balancing in Cloud Data Center. 2014 International Conference on Information

- Technology, Bhubaneswar, India, 2014. P. 12–17. URL: <https://doi.org/10.1109/ICIT.2014.54> (date of access: 10.03.2026).
65. Lin W. C., Zhang L. C. The Load Balancing Research of SDN based on Ant Colony Algorithm with Job Classification. [Електронний ресурс]. URL: <https://doi.org/10.2991/wartia-16.2016.95> (date of access: 10.03.2026).
66. Hassan M. D. A., Hamzah N., Hafizan M. N. Z. M. An Evaluation Of Ant Colony Optimization Based Techniques for Network Load Balancing. 2023 IEEE 13th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2023. P. 162–167. URL: <https://doi.org/10.1109/ISCAIE57739.2023.10165180> (date of access: 10.03.2026).
67. D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proceedings of the IEEE. 2015. vol. 103, no. 1, pp. 14-76, URL: <https://doi.org/10.1109/JPROC.2014.2371999> (date of access: 10.03.2026).
68. O. Salman, I. H. Elhajj, A. Kayssi and A. Chehab, "SDN controllers: A comparative study," 2016 18th Mediterranean Electrotechnical Conference (MELECON), Lemesos, Cyprus, 2016, pp. 1-6, URL: <https://doi.org/10.1109/MELCON.2016.7495430> (date of access: 10.03.2026).
69. Nunes B., Mendonca M., Nguyen X., Obraczka K., Turletti T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Communications Surveys & Tutorials. 2014. URL: <https://doi.org/10.1109/SURV.2014.012214.00180> (date of access: 10.03.2026).
70. Mao Y., Zhang J., Letaief K. Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. IEEE Journal on Selected Areas in Communications. 2016. URL: <https://doi.org/10.1109/JSAC.2016.2611964> (date of access: 10.03.2026).
71. Musaddiq, A.; Olsson, T.; Ahlgren, F. Reinforcement-Learning-Based Routing and Resource Management for Internet of Things Environments: Theoretical Perspective and Challenges. Sensors 2023, 23, 8263. URL: <https://doi.org/10.3390/s23198263> (date of access: 10.03.2026).

72. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. vol. 521, no. 7553. URL: <https://doi.org/10.1038/nature14539> (date of access: 10.03.2026).
73. Sathupadi K., Achar S., Bhaskaran S. V., Faruqui N., Abdullah-Al-Wadud M., Uddin J. Edge-Cloud Synergy for AI-Enhanced Sensor Network Data: A Real-Time Predictive Maintenance Framework. *Sensors*. 2024. Vol. 24. P. 7918. URL: <https://doi.org/10.3390/s24247918> (date of access: 10.03.2026).
74. DeMedeiros K., Hendawi A., Alvarez M. A Survey of AI-Based Anomaly Detection in IoT and Sensor Networks. *Sensors*. 2023. Vol. 23. P. 1352. URL: <https://doi.org/10.3390/s23031352> (date of access: 10.03.2026).
75. Ficili I., Giacobbe M., Tricomi G., Puliafito A. From Sensors to Data Intelligence: Leveraging IoT, Cloud, and Edge Computing with AI. *Sensors*. 2025. Vol. 25. P. 1763. URL: <https://doi.org/10.3390/s25061763> (date of access: 10.03.2026).
76. Klymenko I., Gaidai A., Nikolskyi S., Tkachenko V. The Architectural Concept Of The Monitoring System On The Basis On A Neuron Module IoT Data Analytics. - Vol. 2, No. 41 - 2022. – pp. 111-123. URL: <https://doi.org/10.20535/1560-8956.41.2022.271355> (date of access: 10.03.2026)
77. Lane N. D., Bhattacharya S., Mathur A. et al. Squeezing deep learning into mobile and embedded devices. *IEEE Pervasive Computing*. 2017. URL: <https://doi.org/10.1109/MPRV.2017.2940968> (date of access: 10.03.2026).
78. Tariq M. U., Saqib S. M., Mazhar T., Khan M. A., Shahzad T., Hamam H. Edge-enabled smart agriculture framework: Integrating IoT, lightweight deep learning, and agentic AI for context-aware farming. *Results in Engineering*. 2025. Vol. 28. 107342. ISSN 2590-1230. URL: <https://doi.org/10.1016/j.rineng.2025.107342>. (date of access: 10.03.2026).
79. Alajlan N. N., Ibrahim D. M. TinyML: Enabling of Inference Deep Learning Models on Ultra-Low-Power IoT Edge Devices for AI Applications. *Micromachines*. 2022. Vol. 13. P. 851. URL: <https://doi.org/10.3390/mi13060851> (date of access: 10.03.2026).

80. Abadade Y., Temouden A., Bamoumen H., Benamar N., Chtouki Y., Hafid A. S. A Comprehensive Survey on TinyML. *IEEE Access*. 2023. Vol. 11. P. 96892–96922. URL: <https://doi.org/10.1109/ACCESS.2023.3294111> (date of access: 10.03.2026).
81. Warden P., Situnayake D. TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. 2019. [Электронный ресурс]. URL: <https://api.semanticscholar.org/CorpusID:226814321> (date of access: 10.03.2026).
82. Banbury, C., Reddi, V. J., Torelli, P., Holleman, J., Jeffries, N., Kiraly, C., ... & Xuesong, X. Mlperf tiny benchmark.. 2021. URL: <https://doi.org/10.48550/arXiv.2106.07597> (date of access: 10.03.2026).
83. Heydari S., Mahmoud Q. H. Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions. *Sensors*. 2025. Vol. 25. P. 3191. URL: <https://doi.org/10.3390/s25103191> (date of access: 10.03.2026).
84. Zhang C., Patras P., Haddadi H. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials*. 2019. URL: <https://doi.org/10.1109/COMST.2019.2904897> (date of access: 10.03.2026).
85. Stephansen J. B., Olesen A. N., Olsen M. et al. Neural network analysis of sleep stages enables efficient diagnosis of narcolepsy. *Nature Communications*. 2018. Vol. 9. P. 5229. URL: <https://doi.org/10.1038/s41467-018-07229-3> (date of access: 10.03.2026).
86. Dhruva A. R., Alam K. N., Khan M. S., Bourouis S., Khan M. M. Development of an IoT-Based Sleep Apnea Monitoring System for Healthcare Applications. *Computational and Mathematical Methods in Medicine*. 2021. URL: <https://doi.org/10.1155/2021/7152576> (date of access: 10.03.2026).
87. Zabbix case studies. [Электронный ресурс]. URL: https://www.zabbix.com/case_studies (date of access: 10.03.2026).
88. Guo F., Chen C., Li K. Research on Zabbix monitoring system for large-scale smart campus network from a distributed perspective. *Journal of Physics: Conference Series*. 2024. Vol. 20(10s). URL: <https://journal.esrgroups.org/jes/article/view/5153> (date of access: 10.03.2026).
89. Mardiyono A., Sholihah W., Hakim F. Mobile-based Network Monitoring System Using Zabbix and Telegram. 2020 3rd International Conference on Computer and

- Informatics Engineering (IC2IE), Yogyakarta, Indonesia, 2020. P. 473–477. URL: <https://doi.org/10.24191/jcrinn.v7i1.282> (date of access: 10.03.2026).
90. Haidai A., Klymenko I. “Automatic network reconfiguration method with dynamic ip address management.” *Information, Computing and Intelligent systems* 6, 217–229, 2025 URL: <https://doi.org/10.20535/2786-8729.6.2025.334139> (date of access: 10.03.2026).
 91. Suparman A., Akhmad E. P. A., Dinata B. M. Leveraging Artificial Intelligence for Enhancing Cybersecurity: A Deep Learning Approach to Real-Time Threat Detection. *The Journal of Academic Science*. 2024. Vol. 1, no. 7. P. 835–842. URL: <https://doi.org/10.59613/0yv79c49> (date of access: 10.03.2026).
 92. Gupta S., Gupta A., Shankar G. Cloud Computing: Services, Deployment Models and Security Challenges. 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021. P. 414–418. DOI: 10.1109/ICOSEC51865.2021.9591794
 93. Alotaibi A., Barnawi A. Securing massive IoT in 6G: Recent solutions, architectures, future directions. *Internet of Things*. 2023. Vol. 22. 100715. ISSN 2542-6605. URL: <https://doi.org/10.1016/j.iot.2023.100715> (date of access: 10.03.2026).
 94. Kumar P., Gupta G. P., Tripathi R. A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks. *Journal of Ambient Intelligence and Humanized Computing*. 2021. Vol. 12. P. 9555–9572. URL: <https://doi.org/10.1007/s12652-020-02696-3> (date of access: 10.03.2026).
 95. Ahmed N., Hossain M. E., Hossain Z., Kabir F. AI-Enabled System for Efficient Cyber Incident Detection and Response in Cloud Environments: Safeguarding Against Systematic Attacks. *Indonesian Journal of Educational Science and Technology*. 2024. URL: <https://doi.org/10.55927/nurture.v3i4.16> (date of access: 10.03.2026).
 96. Arafune M. et al. Design and Development of Automated Threat Hunting in Industrial Control Systems. 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Pisa, Italy, 2022. P. 618–623. URL:

- <https://doi.org/10.1109/PerComWorkshops53856.2022.9767375> (date of access: 10.03.2026).
97. Angin P., Mert M. B., Mete O., Ramazanli A., Sarica K., Gungoren B. A Blockchain-Based Decentralized Security Architecture for IoT. *Internet of Things – ICIOT 2018* / ed. by D. Georgakopoulos, L. J. Zhang. Cham : Springer, 2018. URL: https://doi.org/10.1007/978-3-319-94370-1_1 (date of access: 10.03.2026).
 98. Huber B., Kandah F. Zero Trust+: A Trusted-based Zero Trust architecture for IoT at Scale. 2024 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2024. P. 1–6. URL: <https://doi.org/10.1109/ICCE59016.2024.10444321> (date of access: 10.03.2026).
 99. Katiyar N., Tripathi S., Kumar P., Verma S. AI and Cyber-Security: Enhancing threat detection and response with machine learning. *Educational Administration Theory and Practice Journal*. 2024. Vol. 30, no. 4. URL: <https://doi.org/10.53555/kuey.v30i4.2377> (date of access: 10.03.2026).
 100. Zhang M., Li X., Wang L. An Adaptive Outlier Detection and Processing Approach Towards Time Series Sensor Data. *IEEE Access*. 2019. Vol. 7. P. 175192–175212. URL: <https://doi.org/10.1109/ACCESS.2019.2957602> (date of access: 10.03.2026).
 101. Haidai A., Klymenko I. “Methodology of adaptive data processing in IoT monitoring systems with multilevel sensor data filtering and self-tuning.” *Information, Computing and Intelligent systems* 7, 110-126, 2025 URL: <https://doi.org/10.20535/2786-8729.7.2025.341409> (date of access: 10.03.2026).
 102. Tey K. S., Leong W. C., Rahman T. A. Z., Choong K. N., Kunjunni B. Data Analysis of Sensors Value Using Various Moving Average Methods. 2024 IEEE 7th International Conference on Electrical, Electronics and System Engineering (ICEESE), Kanazawa, Japan, 2024. P. 1–5. URL: <https://doi.org/10.1109/ICEESE62315.2024.10828550> (date of access: 10.03.2026).
 103. Kusumo L. Aji. Implementation of Median Filter in Data Processing of Temperature and Humidity Monitoring System with DHT 11 and DHT 22 Sensors. *Computer, Control System, and Networking Journal*. 2025. Vol. 4, no. 2. P. 103–110. URL: <https://doi.org/10.58982/krisnadana.v4i2.723> (date of access: 10.03.2026).

104. Armando E. J., Hanyurwimfura D., Gatera O. et al. Evaluation of advanced Kalman filter on real-time agricultural soil parameters through an IoT resources-constrained device. *Scientific Reports*. 2025. Vol. 15. P. 20829. URL: <https://doi.org/10.1038/s41598-025-05427-w> (date of access: 10.03.2026).
105. Kunsina S. P., Iswanto, Maarif A., Pratama G. N. P. Optimized Kalman Filter using Genetic Algorithm for IoT Sensor. 2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), Palembang, Indonesia, 2023. P. 499–503. URL: <https://doi.org/10.1109/EECSI59885.2023.10295867> (date of access: 10.03.2026).
106. Brito T., Azevedo B. F., Mendes J., Zorawski M., Fernandes F. P., Pereira A. I., Rufino J., Lima J., Costa P. Data acquisition filtering focused on optimizing transmission in a LoRaWAN network applied to the WSN forest monitoring system. *Sensors*. 2023. Vol. 23, no. 3. P. 1282. URL: <https://doi.org/10.3390/s23031282> (date of access: 10.03.2026).
107. Liu J., Gao Z., Li Y., Lv S., Liu J., Yang C. Ranging Offset Calibration and Moving Average Filter Enhanced Reliable UWB Positioning in Classic User Environments. *Remote Sensing*. 2024. Vol. 16. P. 2511. URL: <https://doi.org/10.3390/rs16142511> (date of access: 10.03.2026).
108. Baskoro F., Buditjahjanto A., Rohman M., D. F., Nurdiansyah A. Impact of sample size variation on moving average filter performance for stability and accuracy in ultrasonic sensor measurements. *TEM Journal*. 2025. Vol. 14, no. 2. P. 1681–1688. URL: <https://doi.org/10.18421/TEM142-65> (date of access: 10.03.2026).
109. Faghih B., Timoney J. Smart-Median: A new real-time algorithm for smoothing singing pitch contours. *Applied Sciences*. 2022. Vol. 12. P. 7026. URL: <https://doi.org/10.3390/app12147026> (date of access: 10.03.2026).
110. Daru A., Susanto S., Adhiwibowo W., Hirzan A. M. Internet of things based seasonal auto regression integrated moving average model for hydroponic water quality prediction. *International Journal of Advances in Applied Sciences*. 2025. Vol. 14. P. 123–131. URL: <https://doi.org/10.11591/ijaas.v14.i1.pp123-131> (date of access: 10.03.2026).

111. Гайдай А., Клименко І. “Високопродуктивна апаратно-програмна система для граничної обробки сенсорних потоків у системах IoT на платформі FPGA-SOC” Технічні науки та технології 4(42), 195-207, 2025 URL: [https://doi.org/10.25140/2411-5363-2025-4\(42\)-195-207](https://doi.org/10.25140/2411-5363-2025-4(42)-195-207) (date of access: 10.03.2026).
112. Abdulsamad A. A., Répás S. R. Application of FPGA Devices in Network Security: A Survey. Electronics. 2025. Vol. 14. P. 3894. URL: <https://doi.org/10.3390/electronics14193894> (date of access: 10.03.2026).
113. Jumaa N. Survey: Internet of Thing Using FPGA. Iraqi Journal for Electrical and Electronic Engineering. 2017. Vol. 13, no. 1. P. 38–45. URL: <https://doi.org/10.37917/ijeee.13.1.5> (date of access: 10.03.2026).
114. Ahmed S. Z., Sassatelli G., Torres L., Rougé L. Survey of New Trends in Industry for Programmable Hardware: FPGAs, MPPAs, MPSoCs, Structured ASICs, eFPGAs and New Wave of Innovation in FPGAs. 2010 International Conference on Field Programmable Logic and Applications, Milan, Italy, 2010. P. 291–297. URL: <https://doi.org/10.1109/FPL.2010.66> (date of access: 10.03.2026).
115. Ooi C.-P., Tan W.-H., Cheong S.-N., Lee Y.-L., Baskaran V. M., Low Y.-L. FPGA-based embedded architecture for IoT home automation application. Indonesian Journal of Electrical Engineering and Computer Science. 2019. Vol. 14, No. 2. P. 646–652. URL: <https://doi.org/10.11591/ijeecs.v14.i2.pp646-652>. (date of access: 10.03.2026).
116. Kong L., Tan J., Huang J., Chen G., Wang S., Jin X., Zeng P., Khan M., Das S. K. Edge-computing-driven Internet of Things: A survey. ACM Computing Surveys. 2023. Vol. 55, no. 8. Art. no. 174. P. 1–41. URL: <https://doi.org/10.1145/3555308> (date of access: 10.03.2026).
117. Draz H. H., Elashker N. E., Mahmoud M. M. A. Optimized Algorithms and Hardware Implementation of Median Filter for Image Processing. Circuits, Systems, and Signal Processing. 2023. Vol. 42. P. 5545–5558. URL: <https://doi.org/10.1007/s00034-023-02370-x> (date of access: 10.03.2026).
118. Erkan U., Enginoğlu S., Thanh D. N. H., Hieu L. M. Adaptive frequency median filter for the salt and pepper denoising problem. IET Image Processing. 2020. Vol. 14.

- P. 1291–1302. URL: <https://doi.org/10.1049/iet-ipr.2019.0398> (date of access: 10.03.2026).
119. Shejul K., Harikrishnan R., Gupta H. The improved integrated Exponential Smoothing based CNN-LSTM algorithm to forecast the day ahead electricity price. *MethodsX*. 2024. Vol. 13. Article 102923. URL: <https://doi.org/10.1016/j.mex.2024.102923> (date of access: 10.03.2026).
120. Tanenbaum A. S., Wetherall D. J. *Computer networks*. 5th ed. Boston : Pearson, 2011. 960 p.
121. PyTorch documentation. [Электронный ресурс]. URL: <https://docs.pytorch.org/docs/stable/index.html> (date of access: 10.03.2026).
122. Paszke A. et al. Automatic differentiation in PyTorch. 2017. [Электронный ресурс]. URL: <https://api.semanticscholar.org/CorpusID:40027675> (date of access: 10.03.2026).

ДОДАТОК А

Filter.v

```

module filter (
    input  wire      clk,
    input  wire      reset,      // active high

    // Avalon-MM slave
    input  wire [2:0] address,
    input  wire      write,
    input  wire [31:0] writedata,
    input  wire      read,
    output reg  [31:0] readdata
);

// ----- Параметри -----
localparam integer MED_WIN = 5;
localparam integer MA_WIN  = 4;

// alpha = 0.4 => 4/10
localparam integer ALPHA_NUM      = 4;
localparam integer ONE_MINUS_NUM = 6;
localparam integer ALPHA_DEN      = 10;

reg [31:0]      ctrl;
reg [31:0]      status;

reg  signed [31:0] data_in_reg;
reg  signed [31:0] data_out_reg;

reg          ready_out;
reg          busy;

reg  signed [31:0] med_buf [0:MED_WIN-1];
reg  [2:0]         med_idx;      // індекс для запису в med_buf
reg  signed [31:0] stage1_out;   // результат медіани

```

```

reg signed [31:0] tmp_med_comb [0:MED_WIN-1];
reg signed [31:0] median_comb;

integer a, b;
reg signed [31:0] tmp_swap;

reg signed [31:0] ma_buf [0:MA_WIN-1];
reg [1:0] ma_idx;
reg signed [47:0] ma_sum;
reg signed [31:0] stage2_out;

reg signed [47:0] new_sum;
reg signed [31:0] old_ma_member;

reg exp_init;
reg signed [31:0] exp_prev;
reg signed [31:0] stage3_out;
reg signed [47:0] acc;

localparam [1:0]
    S_IDLE = 2'd0,
    S_MED  = 2'd1,
    S_MA   = 2'd2,
    S_EXP  = 2'd3;

reg [1:0] state;

always @* begin
    for (a = 0; a < MED_WIN; a = a + 1) begin
        tmp_med_comb[a] = med_buf[a];
    end

    for (a = 0; a < MED_WIN-1; a = a + 1) begin
        for (b = 0; b < MED_WIN-1-a; b = b + 1) begin
            if (tmp_med_comb[b] > tmp_med_comb[b+1]) begin
                tmp_swap = tmp_med_comb[b];

```

```

        tmp_med_comb[b]    = tmp_med_comb[b+1];
        tmp_med_comb[b+1] = tmp_swap;
    end
end
end

median_comb = tmp_med_comb[MED_WIN/2];
end

always @(posedge clk) begin
    if (reset) begin

        ctrl          <= 32'd0;
        data_in_reg <= 32'sd0;
        data_out_reg <= 32'sd0;

        ready_out     <= 1'b0;
        busy          <= 1'b0;

        med_idx       <= 3'd0;
        stage1_out    <= 32'sd0;
        for (a = 0; a < MED_WIN; a = a + 1) med_buf[a] <= 32'sd0;

        ma_idx        <= 2'd0;
        ma_sum        <= 48'sd0;
        stage2_out    <= 32'sd0;
        for (a = 0; a < MA_WIN; a = a + 1) ma_buf[a] <= 32'sd0;

        exp_init      <= 1'b0;
        exp_prev      <= 32'sd0;
        stage3_out    <= 32'sd0;
        acc           <= 48'sd0;

        state         <= S_IDLE;
    end else begin

```

```

    if (write) begin
        case (address)
            3'd0: begin
                ctrl <= writedata;
            end
            3'd4: begin

                data_in_reg <= writedata;

                med_buf[med_idx] <= writedata;
                med_idx <= (med_idx == MED_WIN-1) ? 3'd0 :
med_idx + 3'd1;

                busy      <= 1'b1;
                ready_out <= 1'b0;
                state      <= S_MED;
            end
            default: begin
            end
        endcase
    end

    case (state)
        S_IDLE: begin
        end

        S_MED: begin
            stage1_out <= median_comb;
            state      <= S_MA;
        end

        S_MA: begin
            old_ma_member = ma_buf[ma_idx];
            new_sum       = ma_sum +

```

```

                                {{16{stage1_out[31]}}},
stage1_out} -
                                {{16{old_ma_member[31]}}},
old_ma_member};

    ma_sum      <= new_sum;
    ma_buf[ma_idx] <= stage1_out;
    ma_idx      <= (ma_idx == MA_WIN-1) ? 2'd0 :
ma_idx + 2'd1;

    stage2_out   <= new_sum >>> 2;

    state        <= S_EXP;
end

S_EXP: begin
    if (!exp_init) begin
        exp_prev   <= stage2_out;
        stage3_out <= stage2_out;
        exp_init    <= 1'b1;
        acc         <= {{16{stage2_out[31]}}},
stage2_out} * ALPHA_DEN;
    end else begin
        acc <= ALPHA_NUM      * {{16{stage2_out[31]}}},
stage2_out} +
                                ONE_MINUS_NUM      *
{{16{exp_prev[31]}}}, exp_prev};
        exp_prev   <= acc / ALPHA_DEN;
        stage3_out <= acc / ALPHA_DEN;
    end

    data_out_reg <= (!exp_init) ? stage2_out : (acc /
ALPHA_DEN);

    busy         <= 1'b0;
    ready_out    <= 1'b1;

```



```

        state          <= S_IDLE;
    end

    default: begin
        state <= S_IDLE;
    end
endcase
end
end

always @* begin
    status      = 32'b0;
    status[0] = ready_out;
    status[1] = busy;
end

always @* begin
    case(address)
        3'd0: readdata = ctrl;
        3'd1: readdata = status;
        3'd4: readdata = data_in_reg;
        3'd5: readdata = data_out_reg;
        default: readdata = 32'hERROR;
    endcase
end

endmodule

```

analytic.v

```

module analytic(
    input  wire      clk,
    input  wire      reset,
    input  wire [2:0] address,
    input  wire      write,

```

```

input  wire [31:0] writedata,
input  wire      read,
output reg  [31:0] readdata
);

localparam integer FRAC = 12;  // Q4.12

reg [31:0] ctrl;
reg [31:0] status;

reg [31:0] data_in_reg;
reg [31:0] data_out_reg;

reg ready_out;
reg busy;

reg signed [15:0] x [0:10];
reg [3:0] x_cnt;          // 0..11

reg signed [15:0] W1 [0:7][0:10];
reg signed [15:0] b1 [0:7];

reg signed [15:0] W2 [0:3][0:7];
reg signed [15:0] b2 [0:3];

reg signed [15:0] W3 [0:0][0:3];
reg signed [15:0] b3 [0:0];

integer ii, jj;

initial begin
    // init to 0
    for (ii=0; ii<8; ii=ii+1) begin
        for (jj=0; jj<11; jj=jj+1) W1[ii][jj] = 16'sd0;
        b1[ii] = 16'sd0;
    end
end

```

```

for (ii=0; ii<4; ii=ii+1) begin
    for (jj=0; jj<8; jj=jj+1) W2[ii][jj] = 16'sd0;
    b2[ii] = 16'sd0;
end
for (jj=0; jj<4; jj=jj+1) W3[0][jj] = 16'sd0;
b3[0] = 16'sd0;

W1[0][0] = 16'sd1375; W1[0][1] = 16'sd1018; W1[0][2] = -
16'sd728;
W1[0][3] = 16'sd619; W1[0][4] = -16'sd271; W1[0][5] =
16'sd249;
W1[0][6] = -16'sd990; W1[0][7] = 16'sd467; W1[0][8] = -
16'sd962;
W1[0][9] = -16'sd1341; W1[0][10] = 16'sd483;

W1[1][0] = -16'sd385; W1[1][1] = 16'sd505; W1[1][2] =
16'sd607;
W1[1][3] = 16'sd1280; W1[1][4] = -16'sd174; W1[1][5] =
16'sd952;
W1[1][6] = 16'sd677; W1[1][7] = -16'sd405; W1[1][8] =
16'sd2780;
W1[1][9] = 16'sd115; W1[1][10] = 16'sd752;

W1[2][0] = -16'sd502; W1[2][1] = 16'sd819; W1[2][2] = -
16'sd975;
W1[2][3] = -16'sd569; W1[2][4] = -16'sd349; W1[2][5] = -
16'sd743;
W1[2][6] = 16'sd117; W1[2][7] = -16'sd1220; W1[2][8] =
16'sd1115;
W1[2][9] = -16'sd1049; W1[2][10] = 16'sd953;

W1[3][0] = -16'sd329; W1[3][1] = -16'sd721; W1[3][2] =
16'sd1202;
W1[3][3] = 16'sd856; W1[3][4] = 16'sd998; W1[3][5] =
16'sd135;

```

W1[3][6] = 16'sd296; W1[3][7] = 16'sd600; W1[3][8] =
16'sd2089;

W1[3][9] = 16'sd1088; W1[3][10] = 16'sd1832;

W1[4][0] = 16'sd383; W1[4][1] = -16'sd751; W1[4][2] =
16'sd1071;

W1[4][3] = 16'sd770; W1[4][4] = 16'sd627; W1[4][5] = -
16'sd753;

W1[4][6] = -16'sd315; W1[4][7] = -16'sd212; W1[4][8] =
16'sd959;

W1[4][9] = 16'sd973; W1[4][10] = 16'sd811;

W1[5][0] = -16'sd304; W1[5][1] = -16'sd379; W1[5][2] =
16'sd151;

W1[5][3] = 16'sd1353; W1[5][4] = -16'sd877; W1[5][5] =
16'sd78;

W1[5][6] = -16'sd678; W1[5][7] = 16'sd57; W1[5][8] =
16'sd2394;

W1[5][9] = 16'sd956; W1[5][10] = 16'sd501;

W1[6][0] = 16'sd1024; W1[6][1] = -16'sd732; W1[6][2] = -
16'sd737;

W1[6][3] = -16'sd737; W1[6][4] = 16'sd1111; W1[6][5] =
16'sd412;

W1[6][6] = 16'sd1188; W1[6][7] = -16'sd1019; W1[6][8] = -
16'sd1225;

W1[6][9] = -16'sd966; W1[6][10] = -16'sd831;

W1[7][0] = 16'sd696; W1[7][1] = 16'sd302; W1[7][2] =
16'sd1424;

W1[7][3] = -16'sd683; W1[7][4] = -16'sd842; W1[7][5] =
16'sd655;

W1[7][6] = -16'sd839; W1[7][7] = 16'sd1386; W1[7][8] = -
16'sd293;

W1[7][9] = 16'sd1396; W1[7][10] = -16'sd1454;

```

        b1[0] = -16'sd818; b1[1] = 16'sd168;  b1[2] = 16'sd261;  b1[3]
= -16'sd450;
        b1[4] = 16'sd712;  b1[5] = 16'sd270;  b1[6] = -16'sd896; b1[7]
= -16'sd842;

        W2[0][0]=16'sd1415;          W2[0][1]=-16'sd808;          W2[0][2]=-
16'sd513;  W2[0][3]=-16'sd2411;
        W2[0][4]=-16'sd1041;W2[0][5]=16'sd24;          W2[0][6]=-
16'sd191;  W2[0][7]=-16'sd2519;

        W2[1][0]=16'sd34;          W2[1][1]=-16'sd989;  W2[1][2]=-16'sd1229;
W2[1][3]=-16'sd797;
        W2[1][4]=-16'sd1267;W2[1][5]=-16'sd922;
W2[1][6]=16'sd1448;  W2[1][7]=16'sd274;

        W2[2][0]=16'sd783;          W2[2][1]=-16'sd1560;W2[2][2]=-
16'sd951;  W2[2][3]=-16'sd1730;
        W2[2][4]=16'sd148;          W2[2][5]=-16'sd1558;W2[2][6]=-
16'sd624;  W2[2][7]=-16'sd2683;

        W2[3][0]=16'sd4;          W2[3][1]=-16'sd539;          W2[3][2]=-
16'sd100;  W2[3][3]=-16'sd981;
        W2[3][4]=-16'sd994;          W2[3][5]=-16'sd845;          W2[3][6]=-
16'sd496;  W2[3][7]=-16'sd1143;

        b2[0]=16'sd1259;          b2[1]=-16'sd287;          b2[2]=16'sd1588;
b2[3]=16'sd451;

        W3[0][0]=-16'sd2802; W3[0][1]=16'sd1417; W3[0][2]=-16'sd3335;
W3[0][3]=-16'sd785;

        // b3 [1]
        b3[0]=-16'sd426;
end

reg [15:0] sigmoid_lut [0:1023];

```

```

initial begin
    $readmemh("sigmoid_lut_q16.hex", sigmoid_lut);
end

reg [3:0] nn_state;
localparam NN_IDLE=0, NN_L1=1, NN_L2=2, NN_L3=3, NN_SIG=4,
NN_DONE=5;

integer i, j;
reg signed [31:0] acc;

reg signed [15:0] h1 [0:7];
reg signed [15:0] h2 [0:3];
reg signed [31:0] z_q12_32;    // keep wider than 16!

reg [15:0] y_q16;

function [9:0] z_to_idx;
    input signed [31:0] z; // Q4.12
    reg signed [31:0] zc;
    reg signed [31:0] num;
    begin
        // clamp to [-8..+8] in Q4.12
        if (z < -32'sd32768) zc = -32'sd32768;           // -8*4096
        else if (z > 32'sd32767) zc = 32'sd32767;       // ~ +8*4096
        (safe)
        else zc = z;

        num = (zc + 32'sd32768) * 32'sd1023;
        z_to_idx = num[31:16]; // divide by 65536 (>>16)
    end
endfunction

always @(posedge clk) begin
    if (reset) begin
        ctrl      <= 32'd0;

```

```

data_in_reg<= 32'd0;
data_out_reg<=32'd0;
ready_out  <= 1'b0;
busy       <= 1'b0;
x_cnt      <= 4'd0;
nn_state   <= NN_IDLE;
y_q16      <= 16'd0;
end else begin
    if (write) begin
        case (address)
            3'd0: begin
                ctrl <= writedata;
            end

            3'd4: begin
                data_in_reg <= writedata;

                // push next feature (Q4.12 in low 16 bits)
                if (!busy) begin
                    x[x_cnt] <= $signed(writedata[15:0]);
                    if (x_cnt == 4'd10) begin
                        x_cnt      <= 4'd0;
                        ready_out <= 1'b0;
                        busy       <= 1'b1;
                        nn_state   <= NN_L1; // auto start
                    end else begin
                        x_cnt <= x_cnt + 4'd1;
                    end
                end
            end
        endcase
    end

    case (nn_state)
        NN_IDLE: begin
            // nothing

```

```

end

NN_L1: begin
    for (i=0; i<8; i=i+1) begin
        acc = {{16{b1[i][15]}}}, b1[i]];
        for (j=0; j<11; j=j+1) begin
            acc = acc + (($signed(W1[i][j]) *
$signed(x[j])) >>> FRAC);
        end
        h1[i] <= (acc[31] ? 16'sd0 : acc[15:0]);
    end
    nn_state <= NN_L2;
end

NN_L2: begin
    for (i=0; i<4; i=i+1) begin
        acc = {{16{b2[i][15]}}}, b2[i]];
        for (j=0; j<8; j=j+1) begin
            acc = acc + (($signed(W2[i][j]) *
$signed(h1[j])) >>> FRAC);
        end
        h2[i] <= (acc[31] ? 16'sd0 : acc[15:0]);
    end
    nn_state <= NN_L3;
end

NN_L3: begin
    acc = {{16{b3[0][15]}}}, b3[0]];
    for (j=0; j<4; j=j+1) begin
        acc = acc + (($signed(W3[0][j]) *
$signed(h2[j])) >>> FRAC);
    end
    z_q12_32 <= acc;      // keep full
    nn_state <= NN_SIG;
end

```



```

        NN_SIG: begin
            y_q16      <= sigmoid_lut[z_to_idx(z_q12_32)];
            nn_state <= NN_DONE;
        end

        NN_DONE: begin
            data_out_reg <= {16'd0, y_q16}; // Q0.16 in low
16
            busy        <= 1'b0;
            ready_out <= 1'b1; // sticky until new run
            nn_state    <= NN_IDLE;
        end
    endcase
end
end

always @* begin
    status = 32'd0;
    status[0]    = ready_out;
    status[1]    = busy;
    status[15:8] = {4'd0, x_cnt};
end

always @* begin
    case(address)
        3'd0: readdata = ctrl;
        3'd1: readdata = status;
        3'd4: readdata = data_in_reg;
        3'd5: readdata = data_out_reg;
        default: readdata = 32'hERROR;
    endcase
end

endmodule

```

ДОДАТОК Б

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці, в яких опубліковано основні наукові результати дисертації:

1. Haidai A., Klymenko I. “Automatic network reconfiguration method with dynamic ip address management.” *Information, Computing and Intelligent systems* 6, 217–229, 2025 DOI: <https://doi.org/10.20535/2786-8729.6.2025.334139> (Фахове наукове видання України категорії Б) *Здобувач провів комплексний аналіз сучасних викликів інформаційної безпеки та спроможностей штучного інтелекту в протидії кіберзагрозам, розробив та практично реалізував метод автоматичної реконфігурації мережі шляхом інтеграції системи моніторингу Zabbix із мережевим шлюзом pfSense, налаштував механізми оперативного реагування, включаючи автоматичну зміну IP-адрес, ізоляцію підозрілих вузлів та адаптацію правил фаєрволу, що дозволило автоматизувати управління інцидентами безпеки. Клименко І. здійснила загальне керівництво, обґрунтувала необхідність переходу до динамічних систем захисту, розробила концепцію керуючого інформаційного середовища та визначила ключові політики безпеки для детектування аномалій у поведінці мережевих вузлів.*

2. Haidai A., Klymenko I. “Methodology of adaptive data processing in IoT monitoring systems with multilevel sensor data filtering and self-tuning.” *Information, Computing and Intelligent systems* 7, 110-126, 2025 DOI: <https://doi.org/10.20535/2786-8729.7.2025.341409> (Фахове наукове видання України категорії Б) *Здобувач розробив алгоритм адаптивного коригування порогових значень на основі середнього значення та стандартного відхилення сигналу. Реалізував інтеграцію адаптивних тригерів у систему Zabbix та провів серію експериментів, які підтвердили підвищення точності вимірювань (зокрема, для датчиків DHT11) та суттєве зменшення кількості хибних спрацювань тригерів. Клименко І. здійснила загальне керівництво, обґрунтувала та розробила трирівневу методику фільтрації потокових даних, що комбінує медіанне, ковзне середнє та експоненційне*

згладжування, забезпечила баланс між придушенням шумів та чутливістю системи, мінімізувавши затримки обробки для роботи в режимі реального часу.

3. Гайдай А., Клименко І. “Високопродуктивна апаратно-програмна система для граничної обробки сенсорних потоків у системах IoT на платформі FPGA-SOC” *Технічні науки та технології* 4(42), 195-207, 2025 DOI: [https://doi.org/10.25140/2411-5363-2025-4\(42\)-195-207](https://doi.org/10.25140/2411-5363-2025-4(42)-195-207) (Фахове наукове видання України категорії Б) *Здобувач розробив концепцію та архітектуру трирівневого фільтраційного модуля на базі FPGA. Реалізував обчислювальний конвеєр, що поєднує медіанний фільтр, ковзне середнє та експоненційне згладжування. Провів комплексні експериментальні дослідження, що підтвердили зменшення затримок на 26% порівняно з програмними аналогами, а також довів ефективність системи в усуненні імпульсних шумів та стабілізації тренду сенсорних сигналів. Клименко І. Розробила програмно-апаратний механізм взаємодії між процесором ARM та логікою FPGA через інтерфейс Avalon-MM. Обґрунтувала використання фіксованого формату обчислень для мінімізації витрат логічних ресурсів та забезпечення паралельної обробки даних у режимі реального часу.*

4. Haidai A., Klymenko I. “Method of load balancing in distributed three-layer IoT architecture.” *Information, Computing and Intelligent systems* 4, 60-68, 2024 DOI: <https://doi.org/10.20535/2786-8729.4.2024.311595> (Фахове наукове видання України категорії Б) *Здобувач розробив математичну модель управління даними та граничними обчислювальними вузлами. Обґрунтував метод динамічного розподілу навантаження залежно від типу даних та поточних показників продуктивності системи, що дозволило оптимізувати використання системних ресурсів і дешевших каналів зв'язку. Клименко І. спроектувала багаторівневу структуру системи та створила її діючий прототип. Провела серію експериментів для перевірки стійкості системи до відмов і мережесвих збоїв, підтвердивши ефективність механізмів тимчасового зберігання та кешування результатів на проміжних вузлах.*

5. Каплунов А., Гайдай А., Гер В., Нікольський С. “Засоби моніторингу мережі в іот інфраструктурі з гібридною архітектурою.” *Телекомунікаційні та інформаційні технології* № 2, с. 22-32, 2023 DOI: <https://doi.org/10.31673/2412->

4338.2022.022332 (Фахове наукове видання України категорії Б) *Здобувач розробив архітектуру системи моніторингу на базі технологій граничних обчислень (Edge Computing). Запропонував концепцію гібридної IoT-інфраструктури, що дозволяє поєднувати локальний моніторинг із хмарними сервісами для застосунків, чутливих до затримок часу. Гер В. розробив методику інтеграції сучасних інструментів моніторингу в мережеву інфраструктуру. Нікольський С: розробив спосіб аналізу продуктивності комп'ютерної мережі на основі зваженого показника. Впровадив використання вагових коефіцієнтів для параметрів мережі, що дозволило підвищити точність оцінки загального стану інфраструктури. Каплунов А. реалізував механізми запобігання критичним станам мережі через балансування навантаження. Довів ефективність запропонованих засобів для моніторингу потоків даних та планування ресурсів у складних IoT-системах.*

6. Клименко І., Гайдай А., Нікольський С., Ткаченко В. “Архітектурна концепція моніторингової системи на основі нейронного модуля IoT аналітики даних.” Адаптивні системи автоматичного управління Том 2 № 41, с. 111-123, 2022 DOI: <https://doi.org/10.20535/1560-8956.41.2022.271355> *Здобувач розробив прототип системи, інтеграцію нейронної мережі для аналітики даних IoT. Забезпечив поєднання доступної переносної електроніки з недорогим апаратним забезпеченням. Нікольський С. розробив теоретичну концепцію системи на базі парадигми периферійних обчислень. Обґрунтував перенесення критичних за часом обчислень із хмарних сервісів на рівень локальної мережі користувача для підвищення швидкості прийняття рішень. Клименко І. провела дослідження швидкості відгуку системи. Виявила та проаналізував виклики, пов'язані з безпекою даних у розподілених мережах та обмеженістю ресурсів пристроїв. Ткаченко В. Організувала тестування прототипу в контексті «розумного будинку» та медичної діагностики. Оцінила вплив системи на енергоефективність та масштабованість мережі, а також сформулювала напрямки для подальших досліджень.*

7. Гайдай А., Клименко І. “Прогресивна архітектура моніторингу сенсорних даних на граничному рівні у розподілених системах IoT” IX International Scientific and Practical Conference «SoftTech-2025» Kyiv, November 26-28, 2025

Здобувач презентував результати алгоритму адаптивного коригування порогових значень на основі середнього значення та стандартного відхилення сигналу. Климєнко І. здійснила загальне керівництво, обґрунтувала та розробила трирівневу методику фільтрації поточкових даних, що комбінує медіанне, ковзне середнє та експоненційне згладжування, забезпечила баланс між придушенням шумів та чутливістю системи, мінімізувавши затримки обробки для роботи в режимі реального часу.

8. Haidai A., Klymenko I. “The method of the functional parameters estimation of the sleep monitoring system based on neural network“ The International Conference on Security, Fault Tolerance, Intelligence, Kyiv, 2022 *Здобувач презентував прототип системи, інтеграцію нейронної мережі для аналітики даних IoT. Забезпечив поєднання доступної переносної електроніки з недорогим апаратним забезпеченням. Кліменко І. провела дослідження швидкості відгуку системи. Виявила та проаналізував виклики, пов'язані з безпекою даних у розподілених мережах та обмеженістю ресурсів пристроїв.*