Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Виявлення аномальної активності у PLC-керованому обладнанні на основі адаптації методів поведінкового аналізу і машинного навчання з урахуванням специфіки циклічних виробничих процесів

Виконав: студент 2 курсу,
гр.1КІТС-24м
Спеціальності 125 – Кібербезпека та захист інформації
Освітня програма – Кібербезпека інформаційних технологій та систем
Дармороз Дмитро Олегович
(прізвище та ініціали)

Керівник к.т.н., доц, доцент каф. МБІС:
Шиян Анатолій Антонович
(прізвище та ініціали)
«09» грудня 2025 р.

Опонент: к.т.н., доц., проф. каф. ОТ
Захарченко Сергій Михайлович
(прізвище та ініціали)
«09» грудня 2025 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
Юрій ЯРЕМЧУК
«09» грудня 2025 р.

Вінниця ВНТУ – 2025 рік
Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека та захист інформації
Освітньо-професійна програма - Кібербезпека інформаційних технологій та систем

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС
Юрій ЯРЕМЧУК
"24" вересня 2025 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Дарморoз Дмитро Олегович
(прізвище, ім'я, по-батькові)

1. Тема роботи:
«Виявлення аномальної активності у PLC-керованому обладнанні на основі адаптації методів поведінкового аналізу і машинного навчання з урахуванням специфіки циклічних виробничих процесів»
2. Керівник роботи: Шиян Анатолій Антонович, к.т.н., доцент
(прізвище, ім'я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "24" вересня 2025 року № 313
3. Строк подання студентом роботи: За тиждень до захисту
4. Вихідні дані до роботи:
Стандарти, електронні джерела, підручники та наукові статті по темі, які стосуються теми магістерської кваліфікаційної роботи
5. Зміст текстової частини:
У Вступі викладено актуальність теми дослідження та сформульовано мету роботи, яка зосереджена на розробці та адаптації методів поведінкового аналізу і машинного навчання для виявлення аномальної активності у PLC-керованому обладнанні. У першому розділі розглянуто теоретичні основи захисту PLC-керованих виробничих систем, проаналізовано архітектуру PLC, охарактеризовано існуючі методи захисту та відповідні програмні засоби. У другому розділі проведено удосконалення методу виявлення аномальної активності на основі

адаптації поведінкового аналізу з урахуванням циклічних виробничих проце
включаючи розробку ER-моделі та відповідного алгоритму. У третьому розді
здійснено програмну реалізацію алгоритму з використанням LSTM-автоенко
та проведено його експериментальну перевірку через інтеграцію з поведінко
аналізатором. Четвертий розділ містить економічне обґрунтування розробки,
включаючи оцінювання комерційного потенціалу, прогнозування витрат і
розрахунок комерційних ефектів від впровадження запропонованої системи
виявлення аномалій.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов`язкових креслень)

Рисунки: 16; Таблиці: 10; Лістинги: 10

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина			
I	Шиян Анатолій Антонович, к.т.н., доцент		
II	Шиян Анатолій Антонович, к.т.н., доцент		
III	Шиян Анатолій Антонович, к.т.н., доцент		
Економічна частина			
IV	Ратушняк Ольга Георгіївна, доцент кафедри ЕПВМ, к.т.н.		

7.Дата видачі завдання 24 вересня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи		Примітка
1	Визначення напрямку магістерської роботи, формулювання теми	24.09.2025	26.09.2025	
2	Аналіз предметної області обраної теми	27.09.2025	30.09.2025	
3	Розробка роботи	01.10.2025	15.10.2025	
4	Написання магістерської роботи на основі розробленої теми	16.10.2025	10.11.2025	
5	Передзеахист магістерської кваліфікаційної роботи	21.11.2025	25.11.2025	
6	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	26.11.2025	05.12.2025	
7	Захист магістерської кваліфікаційної роботи	09.12.2025	13.12.2025	

Студент _____ Дарморoз Д.О
(підпис)

Керівник роботи _____ Шиян А.А
(підпис)

АНОТАЦІЯ

УДК

Дарморoз Дмитро Олегович. Удосконалення методів виявлення аномальної активності у PLC-керованих виробничих системах на основі поведінкового аналізу та машинного навчання з урахуванням циклічних виробничих процесів. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Кібербезпека інформаційних технологій та систем. Вінниця: ВНТУ, 2025. 105 с.

На укр. мові. Бібліогр.: 23 назви; рис.: 21; табл. 10, лістингів 10.

У магістерській кваліфікаційній роботі розроблено програмний засіб для виявлення аномальної активності у PLC-керованих виробничих системах на основі поведінкового аналізу та машинного навчання з урахуванням циклічної природи технологічних процесів. Розглянуто архітектуру PLC, принципи їх функціонування та основні методи захисту інформації. Проаналізовано існуючі методи виявлення аномалій та програмні засоби для захисту PLC.

У методологічній частині обґрунтовано удосконалення методу виявлення аномалій, застосування стандарту ISA-88, побудовано ER-модель взаємодії компонентів системи, сформовано алгоритм роботи з циклічним моніторингом параметрів та автоматичною зупинкою обладнання. У технологічній частині реалізовано систему на основі LSTM-автоенкодера для кожної фази роботи PLC, проведено навчання моделей на нормальних даних, визначено порогові значення реконструкційної помилки та виконано інтеграцію нейронного детектора з поведінковим аналізатором.

В економічній частині обґрунтовано доцільність розробки, розраховано витрати на виконання НДР, інвестиції, чисті прибутки, термін окупності та економічну ефективність впровадження системи у виробничі процеси.

Ключові слова: PLC-керовані системи, виявлення аномальної активності, поведінковий аналіз, машинне навчання, LSTM-автоенкодер, ISA-88, Python.

ABSTRACT

Darmoroz Dmytro Olehovich. Improvement of Methods for Anomaly Detection in PLC-Controlled Industrial Systems Based on Behavioral Analysis and Machine Learning Considering Cyclical Production Processes. Master's Thesis in the specialty 125 – Cybersecurity, Educational Program – Cybersecurity of Information Technologies and Systems. Vinnytsia: VNTU, 2025. 105 p.

In Ukrainian. Bibliography: 23 titles; figures: 21; tables: 10; listings: 10.

The master's thesis presents the development of a software tool for detecting anomalous activity in PLC-controlled industrial systems based on behavioral analysis and machine learning, taking into account the cyclical nature of technological processes. The thesis examines the architecture of PLCs, their operating principles, and main methods of information protection. Existing anomaly detection methods and software tools for PLC protection are analyzed.

In the methodological part, improvements to the anomaly detection method are justified, the ISA-88 standard is applied, an ER model of system component interactions is constructed, and an algorithm for cyclic monitoring of parameters with automatic equipment shutdown in case of deviations is developed. In the technological part, a system based on an LSTM autoencoder for each PLC operation phase is implemented, models are trained on normal data, reconstruction error thresholds are determined, and integration of the neural detector with the behavioral analyzer is carried out.

The economic part substantiates the feasibility of the development, calculates the costs of R&D, investments, net profits, payback period, and economic efficiency of implementing the system in production processes.

Keywords: PLC-controlled systems, anomaly detection, behavioral analysis, machine learning, LSTM autoencoder, ISA-

ЗМІСТ

ВСТУП

Актуальність. У промислових системах харчового виробництва програмовані логічні контролери (PLC) є основним елементом автоматизації. Вони забезпечують керування технологічними процесами, що відбуваються у циклічному режимі: дозування, змішування, миття, термічна обробка та інші операції. Циклічність таких процесів створює визначену послідовність дій обладнання, яка може бути використана для побудови моделей поведінки системи. Разом із тим, підключення PLC-керованих машин до корпоративних мереж та систем віддаленого керування підвищує рівень уразливості до кібератак, що може призвести до збоїв у роботі обладнання, втрати продукції та порушення виробничих планів.

Особливістю харчових підприємств є висока залежність від безперервності технологічних процесів. Збої у роботі обладнання чи втручання сторонніх осіб безпосередньо впливають не лише на економічні показники, а й на якість та безпечність продукції. Традиційні методи моніторингу промислових мереж базуються на аналізі трафіку, сигнатур або статистичних характеристик даних. Проте такі методи здебільшого не враховують специфіку циклічності виробничих операцій, яка притаманна саме харчовим технологіям. Через це підвищується ймовірність як хибних спрацювань, так і пропуску дійсних загроз.

Потреба у дослідженні зумовлена необхідністю розробки та адаптації методів поведінкового аналізу та машинного навчання для виявлення аномалій у PLC-керованих машинах. На відміну від універсальних рішень для промислових мереж, у даній роботі пропонується поєднання поведінкового аналізу та машинного навчання із врахуванням циклічних закономірностей роботи обладнання. Це дозволяє створювати моделі нормального функціонування виробничих процесів та виявляти відхилення, що можуть свідчити про несправності, помилки персоналу чи несанкціоновані дії.

Метою роботи є удосконалення методу виявлення аномальної активності у PLC-керованому обладнанні шляхом інтеграції поведінкового аналізу та

машинного навчання з урахуванням специфіки циклічних виробничих процесів. Досягнення цієї мети передбачає як формування теоретичних основ, так і практичну реалізацію у вигляді прототипу програмного забезпечення.

Для досягнення поставленої мети необхідно виконати наступні **завдання**:

1. Здійснити аналіз існуючих методів виявлення аномалій в PLC-керованих системах на основі поведінкового аналізу і машинного навчання з урахуванням специфіки циклічних виробничих процесів.
2. Удосконалити метод виявлення аномальної активності у PLC-керованому обладнанні шляхом адаптації поведінкового аналізу та машинного навчання до циклічних виробничих процесів.
3. Розробити програмний прототип для реалізації удосконаленого методу виявлення аномальної активності у PLC-керованому обладнанні.

Об'єктом дослідження є процеси захисту PLC-керованих виробничих систем від інформаційних загроз, що виникають унаслідок деструктивних впливів на промислові мережі та контрольні програми.

Предметом дослідження є методи поведінкового аналізу та алгоритми машинного навчання, адаптовані до циклічних виробничих процесів для виявлення аномальної активності.

Методи дослідження. У роботі застосовано системний аналіз для вивчення архітектури PLC-керованих систем, методи математичного моделювання для опису циклічних процесів, алгоритми машинного навчання для побудови моделей поведінки обладнання та виявлення аномалій. Для перевірки результатів використано експериментальне моделювання з залученням даних, що відтворюють типові режими роботи харчового обладнання.

Наукова новизна роботи полягає в адаптації методів поведінкового аналізу та машинного навчання до умов циклічних виробничих процесів, що дозволяє підвищити точність виявлення відхилень у роботі PLC-керованого обладнання. На відміну від існуючих підходів, які здебільшого орієнтовані на загальний аналіз промислових систем, у дослідженні враховано специфіку саме харчових технологій із типовою послідовністю повторюваних операцій.

Практичне значення полягає у можливості використання результатів дослідження для створення програмних засобів, здатних здійснювати моніторинг промислових процесів та виявляти аномальну активність у режимі реального часу. Такі засоби можуть бути інтегровані у системи керування харчових підприємств для підвищення стійкості виробництва до інформаційних загроз та зменшення ризиків втрати продуктивності.

Особистий внесок магістранта полягає у проведенні огляду літературних джерел, аналізі особливостей функціонування PLC-керованих систем, розробці алгоритму удосконаленого методу, реалізації програмного прототипу та оцінці його функціональних характеристик.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ЗАХИСТУ PLC-КЕРОВАНИХ ВИРОБНИЧИХ СИСТЕМ

1.1 Поняття та архітектура PLC

Програмований логічний контролер (PLC, Programmable Logic Controller) є мікропроцесорним пристроєм для автоматизації керування технологічними процесами.

Історія розвитку PLC починається з кінця 1960-х років, коли автомобільна промисловість США потребувала заміни складних релейних схем більш гнучким рішенням. Перші комерційні контролери з'явилися у 1969 році. У 1970-х вони почали підтримувати аналогові сигнали та комунікаційні функції. У 1980-х роках було зроблено спроби стандартизувати промислові мережі (Manufacturing Automation Protocol, MAP). У 1990-х роках розроблено міжнародний стандарт IEC 1131 (нині IEC 61131-3), що уніфікував мови програмування. Сьогодні десятки компаній виробляють PLC, серед них Siemens, Rockwell Automation (Allen-Bradley), Mitsubishi, Schneider Electric, ABB, Beckhoff та Omron [1].

PLC містить програмовану пам'ять, де зберігаються інструкції для виконання операцій логіки, арифметики, таймінгу, підрахунку та послідовного керування [2]. За визначенням Національної асоціації виробників електрообладнання (NEMA), PLC – це цифровий електронний пристрій, який використовує програмовану пам'ять для зберігання команд, що реалізують керуючу логіку, синхронізацію, підрахунок і обчислення з використанням аналогових та дискретних модулів вводу/виводу [3, с. 97-99]. Загальна архітектура PLC наведено на рисунку 1.1.

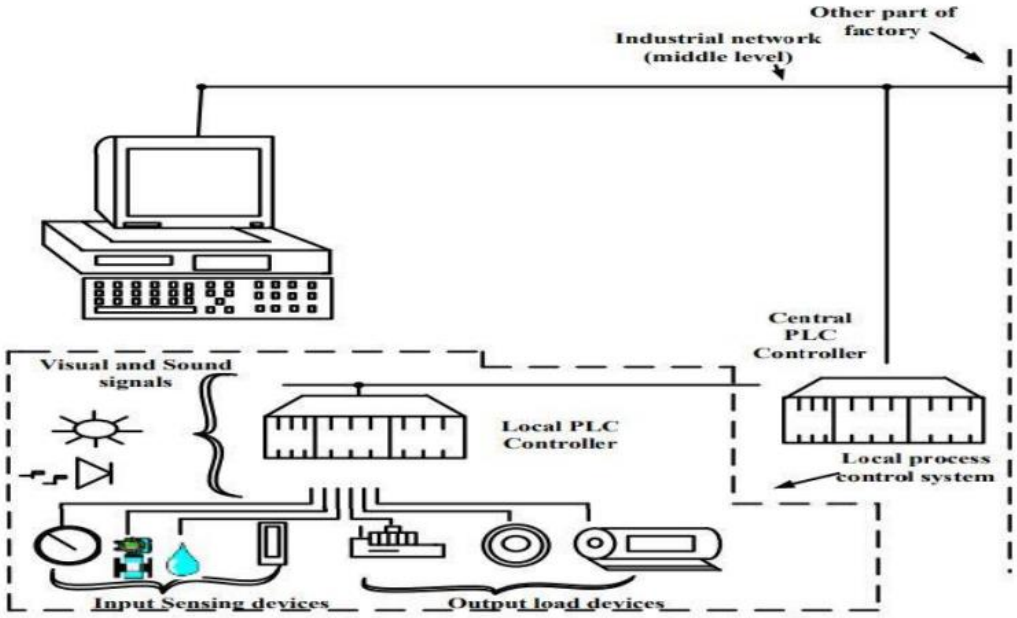


Рисунок 1.1 – Загальна структура PLC [2]

Центральний процесор (CPU) PLC виконує користувацьку програму і визначає реакцію системи на сигнали датчиків. Його функціональні можливості можуть реалізовуватися як на універсальних мікроконтролерах, так і на спеціалізованих процесорах, призначених для логічних операцій. Пам'ять PLC поділяється на системну, де зберігається прошивка, та користувацьку, що містить програму керування. Блок живлення забезпечує роботу пристрою від мережі 24 VDC або 220 VAC.

Модулі вводу/виводу забезпечують взаємодію з реальним обладнанням. Дискретні входи призначені для реєстрації станів «увімкнено/вимкнено», тоді як аналогові дозволяють зчитувати значення температури, тиску, витрати чи рівня. Після оцифрування дані передаються на центральний процесор для подальшої обробки. Вихідні модулі, у свою чергу, керують виконавчими механізмами, замикаючи чи розмикаючи релейні контакти або формуючи аналогові сигнали для регулюючих пристроїв. Структурна схема PLC пристроїв наведено на рисунку 1.2.

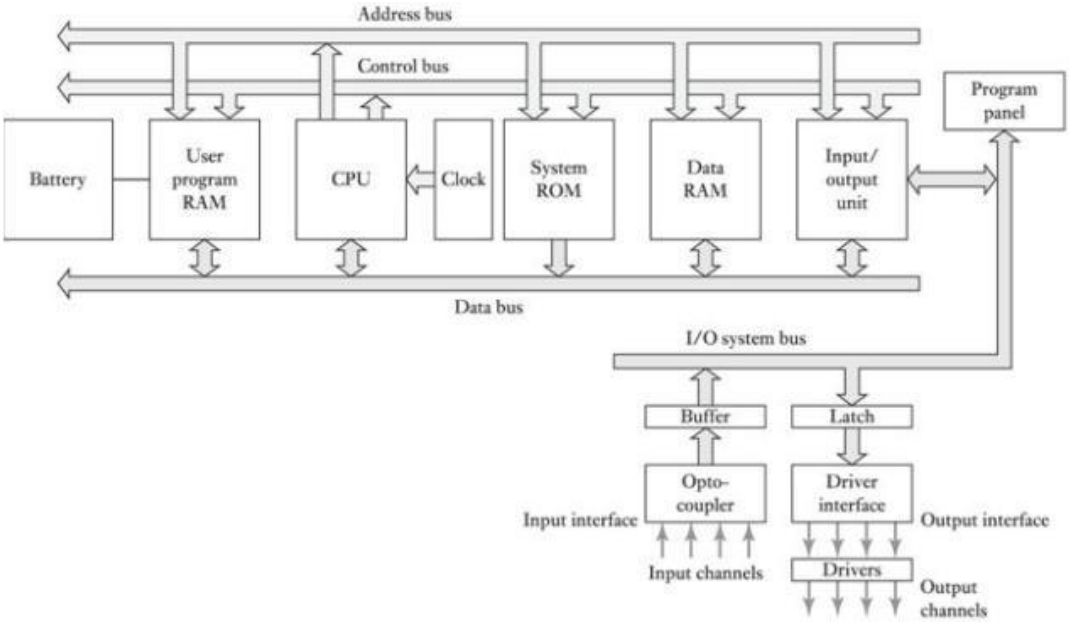


Рисунок 1.2 – Блок-схема PLC із CPU, модулями вводу/виводу та живлення [4]

Програмування PLC здійснюється за допомогою спеціалізованого програмного забезпечення. Стандарт IEC 61131-3 визначає п’ять мов програмування: релейно-контактні схеми (Ladder Diagram), функціональні блоки (Function Block Diagram), структурований текст (Structured Text), інструкційні списки (Instruction List) та послідовні функціональні діаграми (Sequential Function Chart) [5, с. 35238-35239]. Це дозволяє інженерам обирати метод, найбільш зручний для конкретного завдання, що робить PLC доступними як для спеціалістів з електроніки, так і для програмістів.

У промислових системах PLC зазвичай інтегруються у більш широкі автоматизовані комплекси. Через комунікаційні інтерфейси вони взаємодіють із панелями оператора (HMI), системами диспетчерського керування (SCADA) та корпоративними інформаційними ресурсами. Найпоширенішими протоколами є Modbus, PROFINET, EtherNet/IP та OPC UA, що забезпечують як обмін даними в реальному часі, так і інтеграцію з аналітичними платформами [6].

Таким чином, архітектура PLC поєднує апаратні та програмні компоненти, які забезпечують циклічне виконання програм, стійкість до впливу промислового середовища та можливість масштабування від простих систем керування до

складних мережевих рішень. Ці властивості визначають значення PLC як базового елемента сучасної промислової автоматизації.

1.2 Методи захисту інформації та виявлення аномальної активності в PLC-керованих виробничих системах

Одним із підходів до захисту PLC-керованих систем є моніторинг часової поведінки завдань для виявлення атак типу «відмова в обслуговуванні» (DoS). Цей метод базується на тому факті, що PLC-завдання виконуються з заздалегідь визначеною частотою, тобто періодично з фіксованим інтервалом [7].

DoS-атака на PLC порушує нормальну обробку контролера, відключаючи його здатність спілкуватися з іншими компонентами ICS або виконувати програми керуючої логіки. Неконтрольована передача трафікових даних, типова для DoS-атак, впливає на доступність системи. Це може призводити до втрати пакетів, збільшення затримок у мережевому трафіку системи керування, пропуску завдань та загального порушення роботи системи.

Алгоритм виявлення базується на моніторингу використання процесора (CPU) завданнями PLC. Виміряний час виконання завдань відстежується статистичним завданням. Кожне PLC-завдання має свій період, тому використання CPU обчислюється за формулою:

$$K_{CPU} = T_{execution}/T_{period} \quad (1.1)$$

де K_{CPU} – коефіцієнт використання процесора;

$T_{execution}$ – час виконання завдання, с;

T_{period} – період завдання, с.

Час виконання завдання можна виміряти, зчитуючи апаратний таймер при кожному перемиканні контексту. Якщо перемикання відбувається через

переривання, враховуються накладні витрати на його обробку. На рисунку 1.3 показано розрахунок часу виконання з урахуванням накладних витрат. В області даних користувача завдання зберігаються: кількість перемикань контексту, час виконання завдання в поточному періоді та загальний час виконання завдання.

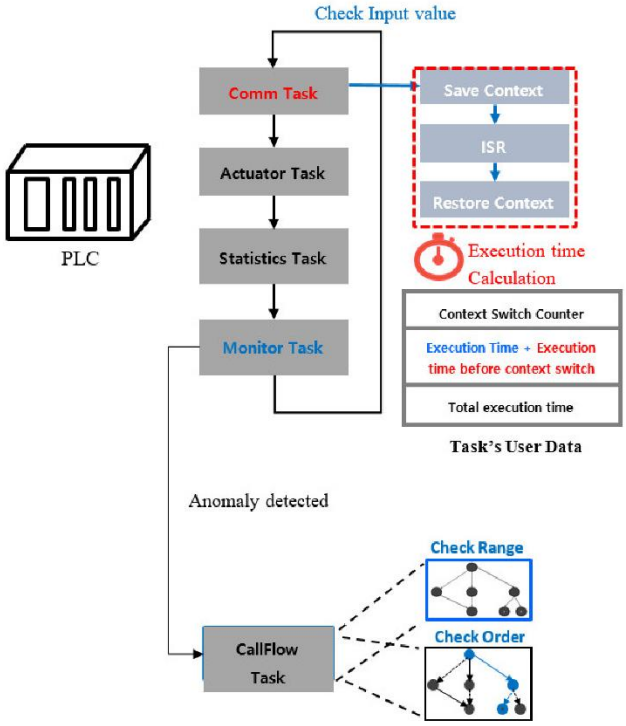


Рисунок 1.3 – Вимірювання часу виконання завдань та виявлення аномалій у потоці управління [8]

Для розрізнення між DoS-атаками та атаками ін'єкції коду використовується метод аналізу потоку керування. Він базується на формуванні графа викликів, що відображає всі можливі потоки керування завдання під час компіляції. Додатково застосовується тіньовий стек, у якому зберігаються адреси повернення кожного виклику функції.

Графи викликів генеруються під час компіляції на основі скомпільованих машинних кодів із використанням даних про адресу функції та її розмір. Кожна функція отримує індекс, який стає вузлом у графі. Ребро графа відповідає виклику від однієї функції до іншої. Структура зберігання графа викликів у пам'яті PLC показана на рисунку 1.4.

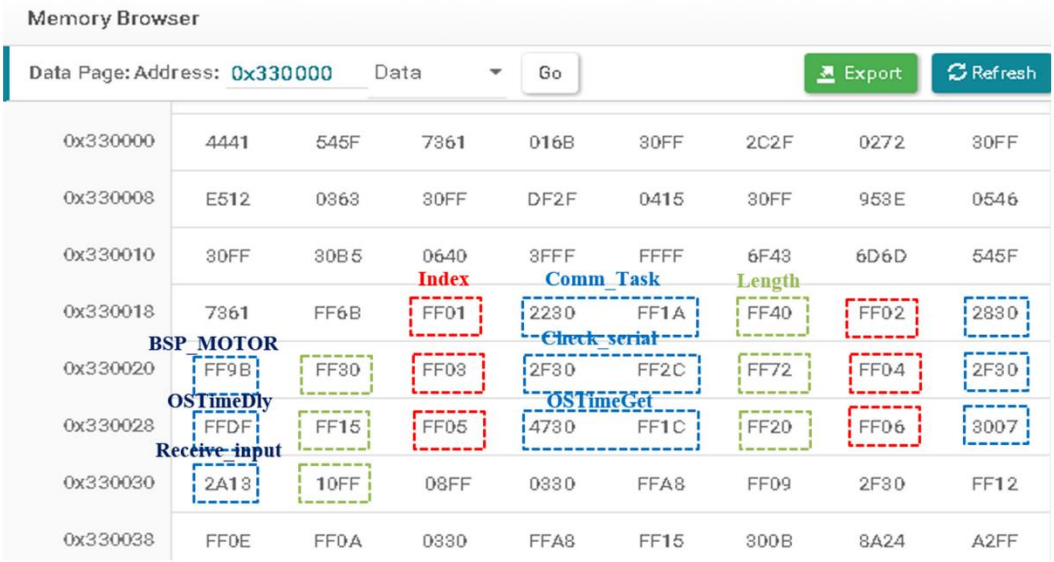


Рисунок 1.4 – Структура зберігання графа викликів функцій у пам'яті PLC [8]

Як показано на рисунку 1.4, граф викликів зберігається у вигляді масиву у флеш-пам'яті PLC. Кожна функція описується індексом, початковою адресою (наприклад, BSP_MOTOR з адресою FF9B), довжиною коду та зв’язками з іншими функціями. Це дає змогу контролю системи відстежувати послідовність викликів і виявляти аномалії у потоці керування.

Тіньовий стек слугує додатковим захисним механізмом для адрес повернення, що знижує ризик атак переповнення буфера. Виклики функцій зберігаються як у стандартному стеці завдань, так і у теневому стеці. При виявленні відхилення активується завдання CallFlow для перевірки відповідності послідовності викликів функцій сформованому графу [9].

Наступним етапом розвитку методів виявлення аномальної активності стало застосування алгоритмів машинного навчання. Серед них варто виділити однокласові методи, які дозволяють формувати модель нормальної поведінки без потреби у великих обсягах мічених даних.

Математичні основи однокласової машини опорних векторів (OCSVM) базуються на адаптації звичайних SVM. Для навчального набору даних:

$$\{X_i | i = 1, 2, \dots, n\}, X \in R^d$$

OCSVM формулює задачу оптимізації, спрямовану на максимізацію відстані між початком координат та гіперплощиною, що розділяє дані в просторі ознак:

$$min_{w,b} \frac{1}{2}|w|^2 - \frac{1}{\nu N}\sum_{i=1}^{N} \quad \xi_i - b \tag{1.2}$$

за умов:

$$\langle w, \Phi(X_i)\rangle \geq b - \xi_i, \xi_i \geq 0$$

де:

ξ_i – змінна послаблення, яка дозволяє алгоритму враховувати точки даних, що дещо відрізняються від нормальних;

ν (0, 1) – параметр, що контролює верхню межу частки помилок;

$\Phi(X_i)$ – функція перетворення даних для того, щоб їх легше було розділити алгоритмом.

Функція рішення для нових даних X_n визначається як:

$$f(X) = w^T \Phi(X_n) - b \tag{1.3}$$

де:

w – нормальний вектор гіперплощини;

b [0,1] – параметр, що контролює ширину межі розділення;

$\Phi(X_n)$ – перетворені дані, що допомагають алгоритму правильно відокремити нормальні точки від аномальних.

Для порівняння розглядається метод Isolation Forest (Ізоляційний ліс, IF). Цей підхід представлено в [10], де зазначено, що IF у складі ансамблевих моделей демонструє вищі результати, ніж OCSVM та однокласові нейронні мережі (OCNN), при виявленні аномалій у PLC-системах.

Концепція IF базується на використанні ансамблю випадкових дерев. Кожна точка даних ізолюється шляхом послідовних розділень за випадково вибраними ознаками та порогами. Для аномальних точок характерна менша кількість необхідних розділень, що дозволяє алгоритму відносити їх до відхилень у поведінці системи.

Метод застосовується до багатовимірних сенсорних даних та не потребує попередньої розмітки. Крім того, IF може інтегруватися в ансамблеві підходи для підвищення точності виявлення аномалій.

1.3 Програмні засоби для захисту інформації та виявлення аномальної активності в PLC-керованих виробничих системах

Застосування методів виявлення аномалій у PLC-керованих системах потребує використання спеціалізованих програмних засобів. Вони реалізують як традиційні підходи до контролю мережевого трафіку, так і алгоритми машинного навчання для аналізу поведінки пристроїв.

Одним із поширених рішень є системи виявлення та запобігання вторгненням Snort та Suricata. Ці інструменти спочатку розроблені для загальних комп'ютерних мереж, проте адаптовані для роботи з промисловими протоколами (Modbus, DNP3, Profinet). Засоби ґрунтуються на сигнатурному аналізі: виявлення атаки відбувається шляхом порівняння трафіку з відомими шаблонами шкідливої активності. Додатково можлива інтеграція з поведінковими модулями та розширення баз правил під конкретні умови виробництва [11, 12].

Іншу групу складають комерційні платформи моніторингу промислових мереж, зокрема SCADAfence, Indegy та Nozomi Networks. Вони спеціально орієнтовані на роботу з ICS/PLC, забезпечують збір і аналіз даних з контролерів, побудову профілів їхньої поведінки та виявлення відхилень у роботі. У таких системах застосовуються комбіновані методи: сигнатурний аналіз, контроль цілісності логіки PLC-програм і поведінковий аналіз трафіку. Вони підтримують

інтеграцію зі SCADA-рівнем та централізованими системами управління інформаційною безпекою [13].

Окремо виділяються відкриті дослідницькі платформи, що застосовуються для моделювання атак і перевірки алгоритмів виявлення. Прикладом є Conpot – honeypot для ICS, який імітує роботу PLC-контролерів і протоколів зв'язку, дозволяючи збирати зразки аномальної активності та тестувати механізми захисту [14 – 15].

Порівняння основних програмних засобів подано у таблиці 1.1.

Таблиця 1.1 – Порівняння програмних засобів захисту та виявлення аномалій у PLC-системах

Засіб/платформа	**Тип підходу**	**Методи**	**Особливості застосування**	**Обмеження**
Snort / Suricata	IDS/IPS	Сигнатурний аналіз, підтримка ICS-протоколів	Поширені рішення, можливість розширення правил	Обмежена здатність виявляти невідомі атаки
SCADAfence / Indegy	Моніторинг PLC і SCADA	Сигнатурний + поведінковий аналіз, контроль логіки	Інтеграція з PLC і SCADA, робота у реальному часі	Переважно комерційні, потребують ліцензії
Conpot	Honeypot для ICS	Емуляція PLC і протоколів	Збір зразків атак, тестування алгоритмів	Не використовується у виробництві напряму

Сформовано на основі [11 – 13]

Таким чином, наявні програмні засоби охоплюють широкий спектр завдань захисту, від сигнатурного аналізу мережевого трафіку до комплексного моніторингу поведінки контролерів. Вибір конкретного рішення залежить від

вимог до глибини аналізу, можливості інтеграції з промисловими системами та бюджетних обмежень.

1.4 Висновки до розділу та постановка задачі

Розглянуто архітектуру PLC, принципи їх функціонування та основні методи захисту інформації у промислових системах. PLC поєднують апаратні та програмні компоненти, що забезпечують циклічне виконання керуючих програм та взаємодію з сенсорами й виконавчими механізмами. Аналіз методів виявлення аномальної активності показав наявність як традиційних підходів – моніторинг часу виконання завдань, аналіз графа викликів і використання теневого стеку – так і методів на основі машинного навчання, серед яких виділяються алгоритми класичного машинного навчання (OCSVM, Isolation Forest) та нейромережеві підходи. Серед класичних алгоритмів Isolation Forest демонструє вищі результати завдяки ефективності роботи з багатовимірними даними та низькій чутливості до шуму. Водночас, нейромережеві підходи здатні моделювати складні часові залежності у послідовностях даних, що робить їх перспективними для аналізу циклічних процесів.

Програмні засоби для захисту PLC-систем реалізують як сигнатурні та поведінкові методи, так і алгоритми машинного навчання для виявлення відхилень у роботі контролерів. Приклади включають Snort та Suricata для сигнатурного аналізу, SCADAfence та Indegy для моніторингу PLC/SCADA і дослідницькі платформи типу Conpot для моделювання атак та тестування алгоритмів. Порівняння основних засобів наведено у таблиці 1.1, що дозволяє оцінити їхні можливості та обмеження для конкретних умов виробництва. Проте наявні рішення переважно не враховують послідовність повторюваних операцій, характерну для харчових виробництв.

Необхідність удосконалення методу виявлення аномальної активності зумовлена тим, що циклічні процеси формують стабільні патерни поведінки

обладнання, які можуть бути використані для побудови більш точних моделей нормального функціонування. Це дозволяє знизити кількість хибних спрацювань та підвищити чутливість системи до реальних відхилень, що можуть свідчити про кібератаки, технічні несправності або помилки персоналу. Для реалізації поставленої мети у другому розділі необхідно вирішити наступні задачі:

1. Удосконалити метод виявлення аномальної активності у PLC-керованому обладнанні за рахунок інтеграції поведінкового аналізу і машинного навчання з урахуванням специфіки циклічних виробничих процесів.

2. Сформувати алгоритм роботи удосконаленого методу, що забезпечує виявлення відхилень у режимі реального часу на основі аналізу послідовності операцій технологічного процесу.

РОЗДІЛ 2. УДОСКОНАЛЕННЯ МЕТОДУ ВИЯВЛЕННЯ АНОМАЛЬНОЇ АКТИВНОСТІ У PLC-КЕРОВАНОМУ ОБЛАДНАННІ НА ОСНОВІ АДАПТАЦІЇ МЕТОДІВ ПОВЕДІНКОВОГО АНАЛІЗУ І МАШИННОГО НАВЧАННЯ З УРАХУВАННЯМ СПЕЦИФІКИ ЦИКЛІЧНИХ ВИРОБНИЧИХ ПРОЦЕСІВ

2.1 Удосконалення методу виявлення аномальної активності у PLC-керованому обладнанні за рахунок поведінкового аналізу і машинного навчання та специфіки циклічних виробничих процесів

Виробничі процеси, що керуються програмованими логічними контролерами, характеризуються високою регулярністю виконання операцій. На відміну від традиційних IT-систем, де поведінка користувачів і процесів варіативна, у промисловому середовищі переважають циклічні сценарії. Технологічні параметри змінюються за передбачуваними траєкторіями: температури піднімаються і знижуються у визначених межах, рівень рідин у резервуарах проходить повторювані фази заповнення та спорожнення, тиск та витрата коливаються відповідно до заданої програми керування.

Така специфіка відкриває можливості для удосконалення методів виявлення аномальної активності, оскільки відхилення від циклічної динаміки з високою ймовірністю вказує на несправність обладнання, неправильне керування або зловмисне втручання. Традиційні методи моніторингу трафіку, орієнтовані на сигнатури чи статистику, не завжди враховують цей аспект, що обмежує їхню застосовність у PLC-керованих системах [16].

Для формалізації циклічних виробничих процесів використовується стандарт ISA-88 (Batch Control Standard), який описує модульний підхід до управління партійними процесами [17]. Цей стандарт встановлює ієрархічну структуру, що поєднує три взаємопов'язані моделі:

- процедурну модель, яка визначає послідовність дій;

- фізичну модель, що описує обладнання;
- модель процесу, яка відображає технологічний зміст операцій.

Структура стандарту ISA-88, представлена на рисунку 2.1, демонструє взаємозв'язки між цими трьома моделями.

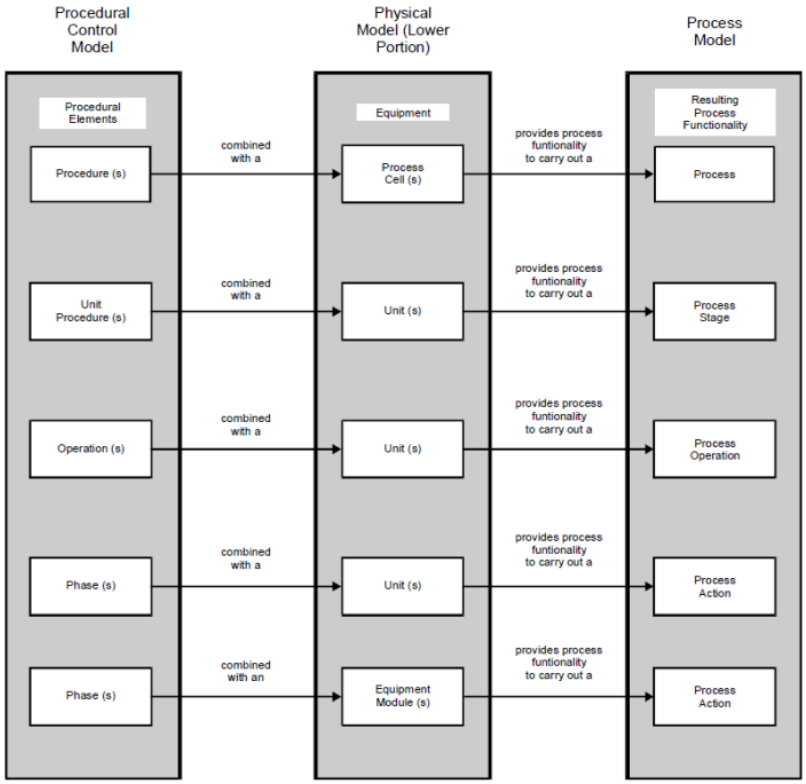


Рисунок 2.1 – Структура стандарту ISA-88 для управління партійними процесами [18]

У процедурній моделі керування виділяються чотири рівні ієрархії:

- Procedure – повна процедура виробничого циклу;
- Unit Procedure – частина процедури для конкретного обладнання;
- Operation – окрема операція в межах процедури;
- Phase – елементарна фаза, що відповідає конкретним діям на обладнанні.

Фізична модель описує ієрархію обладнання від виробничої ділянки (Process Cell) через окремі установки (Unit) до модулів обладнання (Equipment Module).

Модель процесу відображає технологічний зміст від повного процесу до елементарних дій (Process Action). Така структуризація дозволяє точно ідентифікувати нормальну поведінку системи на кожному рівні ієрархії [19].

Застосування стандарту ISA-88 для поведінкового аналізу PLC-систем дозволяє формувати моделі нормальної поведінки з урахуванням фаз технологічного процесу. Кожна фаза характеризується специфічними значеннями параметрів і переходами між станами. У процесі Clean-In-Place виділяються такі фази:

- попереднє ополіскування;
- циркуляція миючого розчину;
- проміжне ополіскування;
- дезінфекція;
- фінальне ополіскування.

Кожна фаза має характерні значення температури, тиску, витрати та часові обмеження. Технологічна схема системи CIP представлена на рисунку 2.2, де показано PI-діаграму з позначенням контрольованих параметрів та виконавчих механізмів.

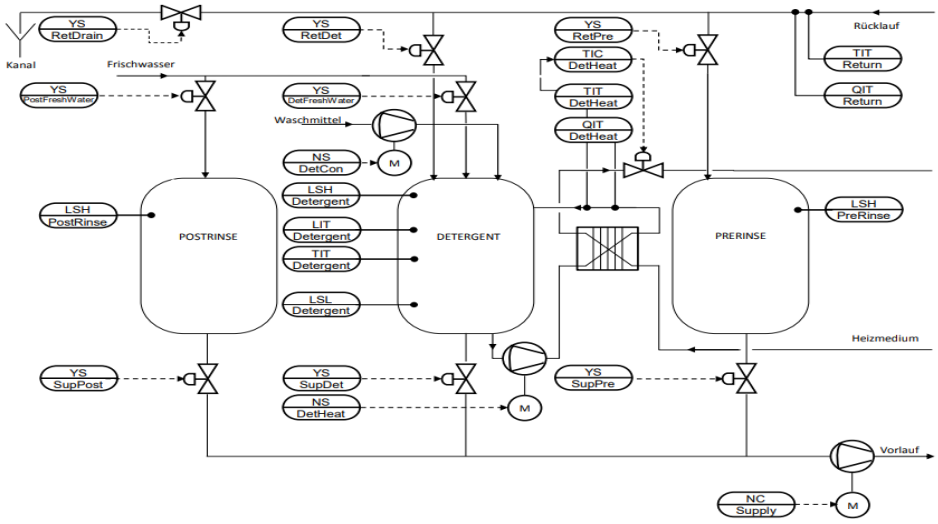


Рисунок 2.2 – PI-діаграма системи CIP з позначенням контрольованих параметрів та виконавчих механізмів [18]

На схемі видно три основні резервуари (POSTRINSE, DETERGENT, PRERINSE), систему клапанів типу YS для керування потоками, датчики рівня LSH/LSL, датчики температури TIT, датчики витрати QIT, насоси та теплообмінник. У нормальному режимі параметри, що знімаються з датчиків, повторюються з високою стабільністю протягом кожної фази процесу, формуючи характерні циклічні патерни [18].

Удосконалення методу виявлення аномалій полягає у поєднанні структурованого підходу ISA-88 з алгоритмами машинного навчання. Формування моделі нормальної поведінки відбувається на основі історичних даних з урахуванням фазової структури процесу. Для кожної фази створюється окрема модель, що враховує специфічні для неї параметри та їх динаміку.

Для побудови моделей нормальної поведінки використовуються такі алгоритми машинного навчання:

- One-Class SVM – формує гіперплощину, що відокремлює нормальні точки даних від потенційних аномалій;

- Isolation Forest – виконує випадковий поділ даних з ізоляцією аномалій за меншу кількість кроків;

- LSTM-мережі – враховують часову залежність параметрів і відтворюють характерні цикли процесу.

Порівняльні характеристики цих алгоритмів наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння алгоритмів машинного навчання для виявлення аномалій у PLC-системах

Алгоритм	**Тип даних**	**Складність навчання**	**Інтерпретованість результатів**	**Чутливість до шуму**
One-Class SVM	Статичні вектори ознак	$O(n^2)$ - $O(n^3)$	Низька	Середня
Isolation Forest	Багатовимірні дані	$O(n \log n)$	Середня	Низька
LSTM	Часові ряди	$O(n \times m^2)$	Низька	Висока

Створено на основі [20]

Особливістю запропонованого підходу є аналіз не окремих значень параметрів, а їх послідовностей у контексті поточної фази процесу. Система навчається розпізнавати нормальні переходи між фазами, характерні зміни параметрів всередині кожної фази та допустимі відхилення. Це дозволяє підвищити чутливість до справжніх аномалій при одночасному зменшенні кількості хибних спрацювань.

Інтеграція поведінкового аналізу з фазовою структурою ISA-88 створює багаторівневу систему виявлення аномалій. На нижньому рівні контролюються окремі параметри в межах кожної фази, на середньому – правильність переходів між фазами, на верхньому – відповідність загальній логіці технологічного процесу. Така архітектура забезпечує комплексний захист від різних типів порушень, від випадкових збоїв до цілеспрямованих атак на систему керування.

2.2 Бази даних необхідних характеристик та побудова ER (UML)-моделі удосконаленого методу

Для реалізації удосконаленого методу виявлення аномалій у PLC-керованому обладнанні необхідно створити структуру бази даних, яка дозволяє зберігати та обробляти всі ключові технологічні параметри процесу CIP (Clean-In-Place). CIP широко застосовується у харчовій та фармацевтичній промисловості для автоматизованої мийки обладнання без його демонтажу, що забезпечує високу ефективність та повторюваність процесів.

Структура даних для моделювання процесу CIP базується на реально існуючих тегах та блоках системи PCS7, описаних у документації Siemens PCS 7 Unit Template «CIP – Cleaning in Place» (support.industry.siemens.com). Основні характеристики процесу включають:

− стан обладнання: клапани, насоси, змішувачі;

− параметри рідин: рівень, температура, концентрація миючого розчину;

− часові показники циклу: тривалість фаз, затримки та порядок виконання операцій.

База даних для збереження характеристик процесу CIP логічно поділяється на кілька груп параметрів:

− Дискретні сигнали (BOOL) – відображають стан виконавчих механізмів: клапанів, насосів, датчиків рівня. Кожен сигнал може приймати два стани (0 – вимкнено, 1 – увімкнено).

− Аналогові параметри (REAL) – включають температуру, рівень рідини, концентрацію та витрату миючого розчину. Ці параметри змінюються протягом циклу відповідно до програми керування.

– Часові параметри (DINT / Timers) – відстежують тривалість циклу, окремих фаз та затримки виконання операцій.

Процес CIP складається з послідовності фаз, кожна з яких виконує специфічну функцію у технологічному циклі очищення. Послідовність та функціональне призначення фаз описано у документації Siemens [18, с. 7-9]. Часові характеристики кожної фази визначаються технологічними вимогами до тривалості обробки та ступенем забруднення обладнання. Згідно з літературними даними [21, с. 1; 22], типові інтервали для харчової промисловості наведено у таблиці 2.2.

Таблиця 2.2 – Фази процесу CIP та їх часові характеристики

Фаза	Функціональне призначення	Типова тривалість
Prerinse (Попереднє ополіскування)	Видалення основних залишків продукту водою перед подачею миючого розчину	5–10 хвилин
Detergent (Циркуляція миючого розчину)	Циркуляція лужного або кислотного миючого розчину для розчинення білкових та жирових забруднень	15–30 хвилин
Mixer (Змішувач)	Змішування води та концентрату миючого розчину для підтримки потрібної концентрації	Інтегровано у Detergent
Postrinse (Фінальне ополіскування)	Видалення слідів кислоти та детергенту чистою водою	5–10 хвилин

Створено на основі [18, 21 – 24]

Часові інтервали, наведені у таблиці 2.2, узгоджуються з вимогами міжнародних стандартів гігієнічного проектування обладнання [25–27] та галузевими рекомендаціями щодо забезпечення санітарно-гігієнічних норм у харчовій та фармацевтичній промисловості [28–29]. Тривалість циркуляції миючого розчину (Detergent) залежить від ступеня забруднення обладнання, температурного режиму та концентрації миючих засобів [22–23]. Попереднє ополіскування (Prerinse) забезпечує видалення до 90–95% залишків продукту, що знижує навантаження на наступні фази циклу та зменшує витрати миючих засобів.

Для кращого контролю та аналізу процесу всі параметри прив'язуються до конкретної фази CIP. Основні фази циклу: − Prerinse (Попереднє ополіскування) – етап промивання обладнання водою перед подачею миючого розчину. Контролюються рівні води в баку (LSH_PreRinse, LSL_PreRinse) та робота насоса (Pump_PreRinse). − Detergent (Циркуляція миючого розчину) – етап циркуляції миючого розчину для очищення обладнання. Відстежуються температура (TIC_DetHeat), рівень миючого розчину (LIT_Detergent) та стан клапанів подачі і повернення розчину (YS_DetFreshWater, YS_RetDetergent). − Mixer (Змішувач) – змішування води та миючого розчину для підтримки потрібної концентрації та об'єму. Контролюються рівні у змішувачі (LSH_Mixer, LSL_Mixer) та робота насоса змішувача (NS_Mixer). − Postrinse (Фінальне ополіскування) – завершальна промивка обладнання чистою водою. Контролюються рівні води в баку (LSH_Postrinse, LSL_Postrinse) та стан клапана подачі води (YS_FreshWaterPostrinse).

Додатково параметри, що змінюються у всіх фазах циклу, включають: рівень рідини (LEVEL), температуру (TEMP), концентрацію миючого розчину (FILL_HEAT_CONC) та витрату (QIT). Структура параметрів по фазах циклу CIP представлена у таблиці 2.3.

Таблиця 2.3 – Структура даних для моделювання процесу CIP

Фаза	Назва параметра	Тип даних	Опис	Діапазон значень / стан
Prerinse	LSH_PreRinse	BOOL	Максимальний рівень у баку попереднього ополіскування	0/1
Prerinse	LSL_PreRinse	BOOL	Мінімальний рівень у баку попереднього ополіскування	0/1
Prerinse	Pump_PreRinse	BOOL	Насос попереднього ополіскування	0/1
Detergent	TIC_DetHeat	REAL	Температура миючого розчину	0–95 °C
Detergent	LIT_Detergent	REAL	Поточний рівень у баку миючого розчину	0–100%
Detergent	YS_DetFreshWater	BOOL	Клапан подачі свіжої води у бак миючого розчину	0/1
Detergent	YS_RetDetergent	BOOL	Клапан повернення миючого розчину	0/1
Mixer	LSH_Mixer	BOOL	Максимальний рівень у змішувачі	0/1
Mixer	LSL_Mixer	BOOL	Мінімальний рівень у змішувачі	0/1
Mixer	NS_Mixer	BOOL	Насос змішувача (вкл/викл)	0/1
Postrinse	LSH_Postrinse	BOOL	Максимальний рівень у баку фінального ополіскування	0/1
Postrinse	LSL_Postrinse	BOOL	Мінімальний рівень у баку фінального ополіскування	0/1
Postrinse	YS_FreshWaterPostrinse	BOOL	Клапан подачі свіжої води у бак фінального ополіскування	0/1
Всі фази	LEVEL	REAL	Рівень рідини у баку	0–100%
Всі фази	TEMP	REAL	Температура рідини у баку	0–95 °C
Всі фази	FILL_HEAT_CONC	REAL	Концентрація миючого розчину	0–5%
Всі фази	QIT	REAL	Витрата рідини через систему	0–300 л/хв

Створено на основі [18]

Таким чином, описана структура параметрів процесу CIP дозволяє забезпечити повний збір та систематизацію ключових характеристик обладнання

на всіх фазах мийного циклу. Проте для реалізації удосконаленого методу виявлення аномалій недостатньо лише накопичення технологічних даних – необхідно формалізувати взаємозв'язки між компонентами системи та визначити шляхи обробки інформації. Наведена структура параметрів процесу CIP базується на технічній документації Siemens SIMATIC PCS 7 Unit Template «CIP – Cleaning in Place» версії 8.2 [18, с. 32-41]. Документація містить детальний опис блоків керування, процесних тегів та їх взаємозв'язків у системі автоматизованого мийного комплексу. Параметри таблиці 2.2 відповідають офіційним специфікаціям Siemens PCS 7, що підтверджує технічну коректність обраної моделі даних. Зокрема, параметри для резервуара миючого розчину (Detergent tank) детально описані в розділі 4.2.3 документації [18, с. 32-41], де наведено типи даних, діапазони значень та функціональне призначення кожного параметра.

Діапазони значень аналогових параметрів визначено з урахуванням технологічних вимог процесу CIP у харчовій промисловості [28]. Температурний режим 0–95°C відповідає типовим умовам термічної санітарної обробки обладнання, де підвищені температури забезпечують денатурацію білкових забруднень та руйнування мікробних клітин. Концентрація миючого розчину в діапазоні 0–5% визначається вимогами нормативних документів щодо застосування лужних та кислотних миючих засобів у контакті з харчовими продуктами [30]. Витрата рідини 0–300 л/хв обумовлена гідравлічними характеристиками трубопроводів та продуктивністю насосного обладнання, що забезпечує достатню швидкість потоку для механічного змиву забруднень [29].

Процес CIP регулюється міжнародними стандартами гігієнічного проектування обладнання. Європейська група гігієнічного інжинірингу та проектування (EHEDG) у своєму керівному документі Doc. 8 визначає вимоги до гігієнічного дизайну обладнання, включаючи можливість очищення на місці без демонтажу [25]. Аналогічно, стандарти 3-A Sanitary Standards, що застосовуються у Північній Америці, встановлюють санітарні вимоги до проектування, виготовлення та встановлення обладнання для харчової промисловості [26]. Ці стандарти забезпечують відповідність обладнання вимогам регуляторних органів,

таких як FDA (Food and Drug Administration) у США, та сприяють підвищенню рівня безпеки харчових продуктів [27].

Часові характеристики циклу CIP визначаються послідовністю фаз та технологічними вимогами до тривалості кожного етапу [18, с. 7-9]. Попереднє ополіскування (Prerinse) зазвичай триває 5–10 хвилин, циркуляція миючого розчину (Detergent) – 15–30 хвилин залежно від ступеня забруднення, фінальне ополіскування (Postrinse) – 10–15 хвилин. Ці інтервали узгоджуються з галузевими рекомендаціями щодо забезпечення санітарно-гігієнічних вимог до обладнання у харчовій та фармацевтичній промисловості [31].

Створення синтетичних даних для дослідження кібербезпеки промислових систем керування є усталеною практикою. Дослідження [32] демонструє застосування симульованої фабрики розливу пляшок з PLC-керованим обладнанням та протоколом Modbus для генерації навчальних даних. Симуляція промислових процесів дозволяє відтворити реалістичну поведінку системи без ризиків для діючого виробництва.

Для відтворення аномальної поведінки системи планується використати підхід, описаний у роботі [33], який передбачає створення аномальних даних шляхом систематичної модифікації нормальних даних. Цей метод забезпечує точне відтворення характеристик кібератак та дозволяє генерувати реалістичні сценарії порушення штатної роботи PLC-керованого обладнання.

Таким чином, описана структура параметрів процесу CIP дозволяє забезпечити повний збір та систематизацію характеристик обладнання на всіх фазах мийного циклу. Проте для реалізації удосконаленого методу виявлення аномалій недостатньо лише накопичення технологічних даних – необхідно формалізувати взаємозв'язки між компонентами системи та визначити шляхи обробки інформації.

З цією метою було розроблено ER-модель [34] удосконаленого методу (рис. 2.3), яка відображає послідовність обробки даних від моменту їх генерації контролером до виявлення аномальної активності. Модель складається з семи

основних сутностей, кожна з яких виконує специфічну функцію у загальному ланцюзі аналізу.

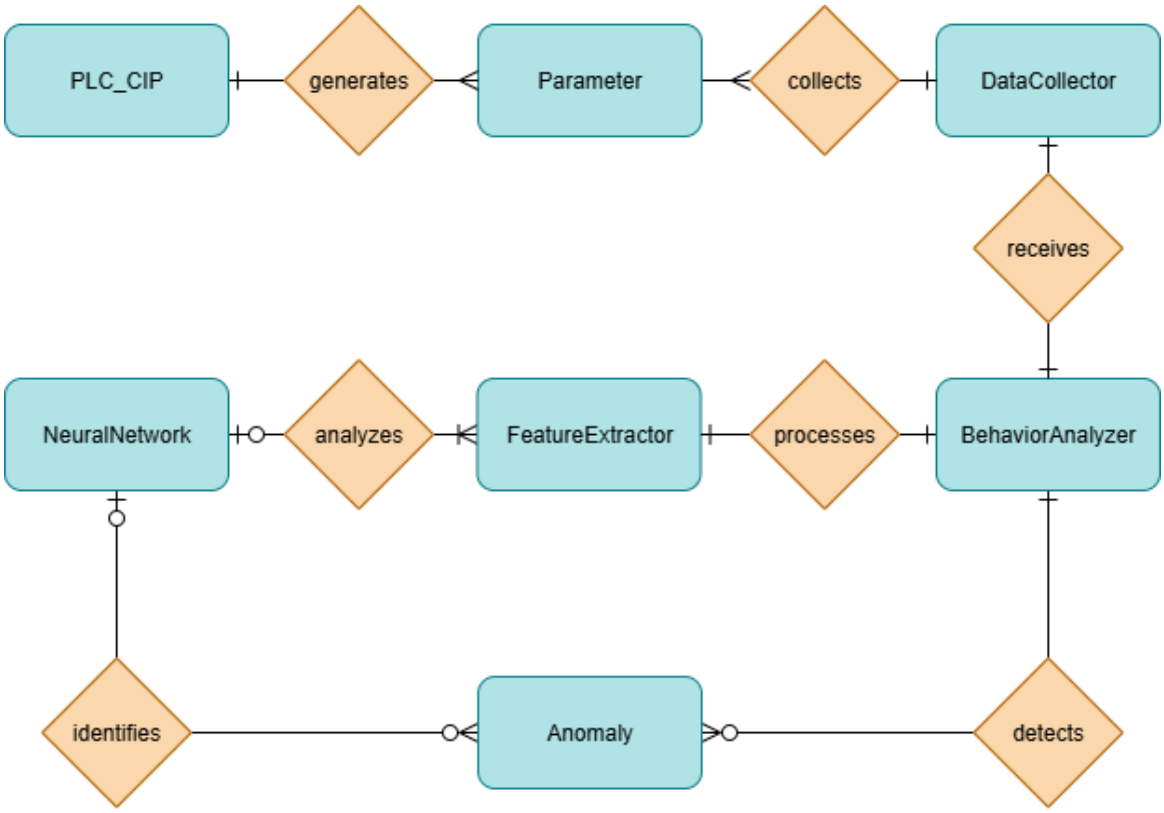


Рисунок 2.3 – ER-модель удосконаленого методу виявлення аномальної активності у PLC-керованому обладнанні

Сутність PLC_CIP представляє програмований логічний контролер, що керує процесом мийки та генерує технологічні параметри. Контролер пов'язаний із сутністю Parameter через зв'язок «generates» з кардинальністю 1:N, оскільки один контролер генерує множину параметрів різних типів протягом усього циклу роботи.

Сутність DataCollector відповідає за збір параметрів від контролера. Зв'язок між Parameter та DataCollector має кардинальність N:M, що відображає можливість багаторазового зчитування одного параметра різними екземплярами колектора, а також збір множини параметрів одним колектором. Після накопичення даних колектор передає їх модулю поведінкового аналізу через зв'язок «receives» з кардинальністю 1:1.

Сутність BehaviorAnalyzer виконує первинний аналіз зібраних даних на основі правил та статистичних характеристик нормальної поведінки системи. Ця сутність є критичною точкою розгалуження методу. Якщо аналізатор виявляє відхилення від типових сценаріїв роботи обладнання, він формує запис про аномалію через зв'язок «detects» з кардинальністю 1:N – один аналізатор може виявити декілька різних типів аномалій протягом одного циклу. У випадку, коли поведінковий аналіз не виявляє очевидних відхилень, дані передаються далі через зв'язок «processes» до сутності FeatureExtractor для поглибленого аналізу засобами машинного навчання.

Сутність FeatureExtractor формує вектор ознак із часових рядів параметрів, виконуючи нормалізацію, вибір релевантних характеристик та підготовку даних у форматі, придатному для подачі на вхід нейронної мережі. Зв'язок між FeatureExtractor та NeuralNetwork має кардинальність 1:1, що відповідає процесу передачі підготовленого вектора ознак єдиній моделі машинного навчання.

Сутність NeuralNetwork представляє навчену модель, яка аналізує підготовлені ознаки та ідентифікує приховані аномалії, які не були виявлені на етапі поведінкового аналізу. Нейронна мережа формує записи про виявлені аномалії через зв'язок «identifies» з кардинальністю 1:N.

Сутність Anomaly є результуючою і містить інформацію про виявлені відхилення від нормальної роботи системи. Кожна аномалія пов'язана з конкретними параметрами через зв'язок з кардинальністю N:M, оскільки одна аномалія може бути спричинена відхиленнями у декількох параметрах одночасно, а один параметр може брати участь у різних аномальних ситуаціях.

Послідовність взаємодії між компонентами [35] системи відображено на рисунку 2.4.

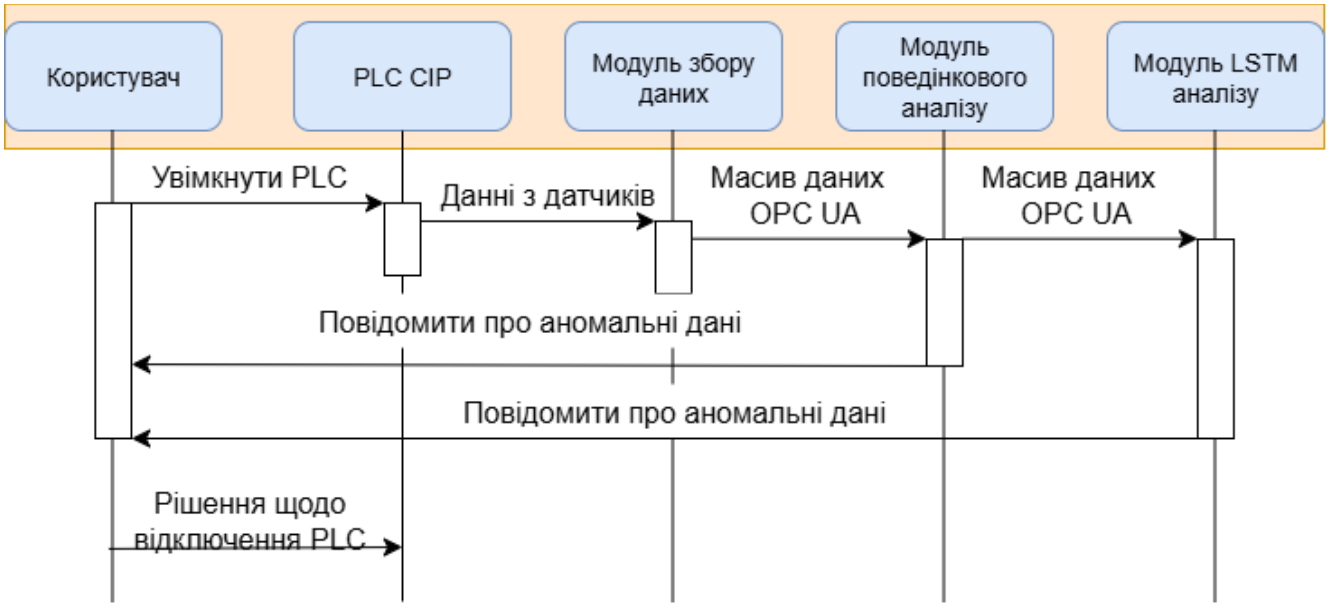


Рисунок 2.4 – Діаграма послідовності роботи удосконаленого методу виявлення аномалій у PLC-керованому обладнанні

Користувач ініціює роботу системи, активуючи PLC CIP, який розпочинає формування та передавання технологічних параметрів процесу через промисловий протокол обміну даними. Отримані дані надходять до модуля збору даних, що виконує перевірку цілісності показників і передає їх на подальший аналіз [36].

На наступному етапі модуль поведінкового аналізу здійснює оцінку отриманих параметрів на основі статистичних характеристик та встановлених нормальних профілів роботи. Якщо виявлено відхилення, система негайно формує повідомлення для користувача про наявність потенційної аномалії у функціонуванні PLC. У випадку відсутності явних порушень дані передаються до модуля LSTM-аналізу, який виконує глибоку перевірку часових рядів з використанням нейронної мережі типу Long Short-Term Memory.

Результати роботи LSTM-аналізу передаються користувачу у вигляді повідомлення про підтверджену або ймовірну аномалію. На основі отриманої інформації користувач приймає рішення щодо подальших дій – продовження роботи обладнання або примусового його відключення для усунення можливих несправностей чи загроз безпеці.

Запропонована структура забезпечує дворівневу систему виявлення аномалій: швидкий поведінковий аналіз для очевидних відхилень та глибокий аналіз нейронною мережею для складних патернів. Така архітектура дозволяє балансувати між швидкодією системи та точністю виявлення прихованих аномалій.

2.3 Формування алгоритму роботи удосконаленого методу

Удосконалений метод виявлення аномальної активності у PLC-керованому обладнанні функціонує як циклічний процес моніторингу технологічних параметрів з дворівневою системою перевірки. Алгоритм складається з послідовних етапів збору, обробки та аналізу даних, що повторюються протягом усього часу роботи обладнання до моменту виявлення аномалії або завершення технологічного циклу.

Загальна структура алгоритму представлена на рисунку 2.5.

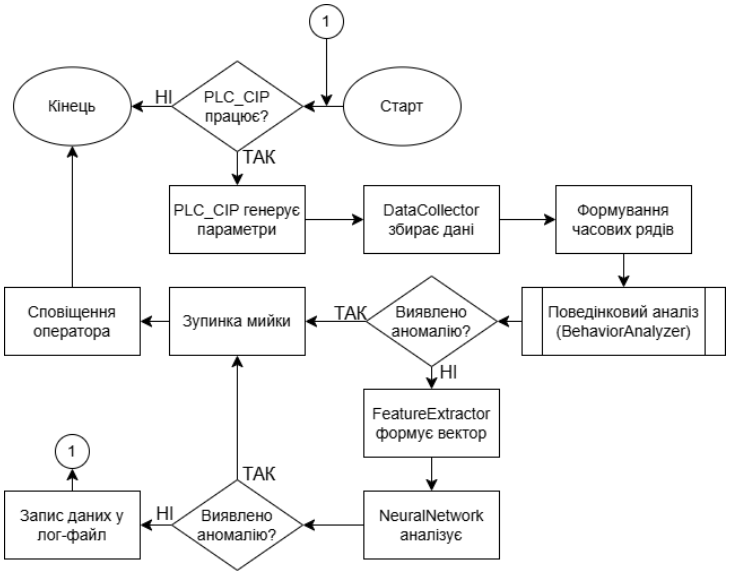


Рисунок 2.5 – Блок-схема алгоритму роботи удосконаленого методу виявлення аномалій

Робота методу розпочинається з перевірки стану контролера PLC_CIP. Якщо контролер не функціонує, алгоритм завершує роботу. У випадку активного стану контролера розпочинається основний цикл обробки даних.

На першому етапі циклу PLC_CIP генерує поточні значення технологічних параметрів процесу мийки. Сутність DataCollector отримує ці параметри та виконує їх накопичення у буфері з прив'язкою до часових міток. Після збору достатньої кількості даних формуються часові ряди для кожного параметра, що дозволяє відстежувати динаміку їх зміни.

Сформовані часові ряди передаються до підпрограми поведінкового аналізу (BehaviorAnalyzer), яка виконує першу стадію перевірки на основі правил та статистичних порогів. Результатом роботи цієї підпрограми є бінарне рішення: виявлено аномалію або ні.

У випадку виявлення аномалії на етапі поведінкового аналізу система переходить до блоку зупинки технологічного процесу. Виконується послідовність дій: зупинка мийки, відключення PLC_CIP та формування сповіщення для оператора з деталями виявленого відхилення. Після цього алгоритм завершує роботу.

Якщо поведінковий аналіз не виявив відхилень, дані передаються до модуля FeatureExtractor, який формує вектор ознак для аналізу нейронною мережею. Сутність NeuralNetwork обробляє підготовлений вектор та виконує другу стадію перевірки. При виявленні аномалії на цьому етапі система аналогічно виконує зупинку процесу, відключення обладнання та сповіщення оператора.

У випадку, коли обидва рівні аналізу не виявили аномалій, поточні параметри, результати перевірок та часові мітки записуються у лог-файл. Ця процедура забезпечує можливість ретроспективного аналізу випадків, коли оператор виявив нестандартну поведінку обладнання, але автоматична система не зафіксувала відхилення. Накопичені дані використовуються для коригування порогових значень та параметрів моделей машинного навчання. Після запису у лог алгоритм повертається до початку циклу для обробки наступної порції даних від контролера.

Підпрограма поведінкового аналізу виконує послідовну перевірку отриманих параметрів за чотирма групами критеріїв. Структура цієї підпрограми формалізована у вигляді псевдокоду (алгоритм 2.1).

Алгоритм 2.1 – Підпрограма поведінкового аналізу (BehaviorAnalyzer)

```
Input: time_series_data[], rule_base, thresholds, current_phase
Output: Anomaly OR pass_to_FeatureExtractor

// Етап 1: Перевірка фазових переходів
IF NOT validate_phase_sequence(current_phase, rule_base):
    RETURN Anomaly(type="phase_sequence_error")

IF NOT validate_phase_duration(current_phase, rule_base):
    RETURN Anomaly(type="phase_duration_error")

// Етап 2: Перевірка діапазонів параметрів
FOR each parameter IN time_series_data:
    IF parameter.type == REAL:
        IF parameter.value < thresholds[parameter.name].min:
            RETURN Anomaly(type="below_threshold",
param=parameter.name)
        IF parameter.value > thresholds[parameter.name].max:
            RETURN Anomaly(type="above_threshold",
param=parameter.name
    IF parameter.type == BOOL:
        expected = rule_base.get_expected_state(parameter.name,
current_phase)
        IF parameter.value != expected:
            RETURN Anomaly(type="incorrect_state",
param=parameter.name)

// Етап 3: Перевірка швидкості зміни
FOR each parameter IN time_series_data WHERE parameter.type ==
REAL:
    rate = (parameter.value - parameter.previous_value) /
time_delta
    IF ABS(rate) > thresholds[parameter.name].max_rate:
        RETURN Anomaly(type="abnormal_rate",
param=parameter.name)

// Етап 4: Перевірка кореляцій
FOR each rule IN rule_base.correlations:
    IF NOT check_correlation(rule, time_series_data):
        RETURN Anomaly(type="correlation_violation", rule=rule)

RETURN pass_to_FeatureExtractor
```

Підпрограма виконує послідовну перевірку за чотирма етапами. На першому етапі контролюється послідовність фазових переходів (Prerinse, Detergent, Mixer, Postrinse) та тривалість кожної фази. Виявлення порушень призводить до формування аномалії типу «phase_sequence_error» або «phase_duration_error».

Другий етап перевіряє відповідність значень параметрів встановленим межам. Аналогові параметри типу REAL порівнюються з мінімальними та максимальними порогами, що формує аномалії «below_threshold» або «above_threshold» при виході за межі. Дискретні параметри типу BOOL перевіряються на відповідність очікуваному стану для поточної фази.

Третій етап контролює швидкість зміни аналогових параметрів через обчислення відношення різниці послідовних значень до часового інтервалу. Перевищення допустимої швидкості вказує на порушення у роботі датчика або обладнання.

Четвертий етап аналізує кореляції між параметрами згідно з правилами взаємозв'язків. При увімкненому насосі попереднього ополіскування (Pump_PreRinse = 1) має спостерігатися зростання рівня води у баку. Порушення таких залежностей класифікується як «correlation_violation».

При виявленні відхилення на будь-якому етапі підпрограма формує запис про аномалію та завершує роботу. У разі успішного проходження всіх перевірок дані передаються до модуля машинного навчання.

Алгоритм працює в умовах, коли заздалегідь визначити фіксовані межі між нормальним та аномальним режимами неможливо. Циклічні технологічні процеси характеризуються тим, що їх робочі профілі залежать від конкретного обладнання, стану трубопроводів, гідравліки, миючих розчинів та інших чинників, що не мають універсальних нормативів. З цієї причини система не використовує жорстко задані рівні аномалій [37].

Поняття норми формується на основі періоду стабільної роботи PLC-керованого обладнання. У цей період процес працює без збоїв, тому всі часові ряди, зібрані у цей інтервал, розглядаються як зразок штатної поведінки. Поведінковий аналізатор та нейромережева модель налаштовуються на цих даних та формують

внутрішні представлення нормальних значень, допустимих коливань та типових залежностей між параметрами.

У цій моделі саме алгоритм формує нормативи. Інженер, оператор або програміст не визначають порогові значення вручну. У процесі навчання система аналізує змінність параметрів, тривалість фаз, статистичні характеристики аналогових сигналів та кореляційні залежності. На основі цих властивостей створюється математичне представлення норми, з яким порівнюються робочі дані в подальшому.

Після завершення навчання на нормальних даних система працює у моніторинговому режимі. Відхилення визначаються відносно профілів, створених на основі стабільного циклу. Порогових значень, заданих у відсотках або абсолютних одиницях, не існує. Відхилення визначається тим, що поточна динаміка параметрів виходить за межі поведінки, яку система спостерігала під час навчання.

Поведінковий аналізатор формує набір обмежень у вигляді фазових правил, діапазонів параметрів, швидкості їх зміни та кореляційних залежностей. Кожне з цих обмежень базується на навчальних даних. Порушення будь-якого правила інтерпретується як відхилення.

Нейромережева модель типу LSTM виконує реконструкцію часових рядів. Якщо відновлені значення систематично відрізняються від фактичних, обчислюється помилка реконструкції. Значення помилки, що виходять за межі статистичного розподілу, сформованого на нормальних даних, інтерпретуються як аномалія. Таким чином, відхилення визначається не за фіксованим рівнем, а як вихід траєкторії параметрів за межі поведінки, властивої стабільному циклу роботи [38].

Після виявлення відхилення система формує структурований запис із характеристиками події. Для зберігання результатів використовується формат JSON. Запис має таку структуру:

– «id» – ідентифікатор події;

– «timestamp» – час виявлення;

– «phase» – фаза CIP-процесу на момент виявлення;

– «parameter_name» – параметр або група параметрів, що спричинили спрацювання;

– «anomaly_type» – тип відхилення (phase_sequence_error, below_threshold, above_threshold, incorrect_state, abnormal_rate, correlation_violation, nn_reconstruction_error);

– «deviation_value» – числовий показник відхилення або помилки реконструкції;

– «source_level» – рівень аналізу, на якому виявлено аномалію: BehaviorAnalyzer або NeuralNetwork.

Записи можуть надалі використовуватися для перегляду роботи системи, коригування параметрів аналізу або розширення навчальної вибірки. Формування бази аномалій не є частиною технологічного процесу, а слугує допоміжною функцією у дослідницькому середовищі, тому вибір JSON як формату зберігання не впливає на метод і не потребує окремої інтеграційної інфраструктури.

2.4 Висновки до розділу

У другому розділі сформовано підхід до удосконалення методу виявлення аномальної активності у PLC-керованому обладнанні. Метод ґрунтується на поєднанні поведінкового аналізу та моделі машинного навчання з урахуванням циклічного характеру процесу CIP. Для формалізації структури процесу застосовано стандарт ISA-88, що дозволяє прив’язувати параметри до фаз технологічного циклу.

У роботі прийнято, що нормальними є дані, зібрані під час безперервного періоду штатної роботи процесу без відхилень. Такі дані сформовані на основі технічної документації Siemens PCS 7 CIP V8.2 та синтетично відтворених часових рядів, побудованих згідно з діапазонами параметрів, типами сигналів та послідовністю фаз, описаних у стандартних шаблонах PCS7. Ці дані

використовуються як база для налаштування поведінкового аналізатора та навчання нейромережевої моделі реконструкції.

Аномалія визначається як розбіжність між фактичними даними та моделями норми, сформованими під час навчання. Розроблена поведінкова підпрограма виконує чотири типи перевірок: відповідність фазовим переходам, належність аналогових сигналів до робочих діапазонів, швидкість їх зміни та дотримання кореляцій між параметрами. Нейронна мережа типу LSTM виконує реконструкцію часових рядів та фіксує відхилення за величиною помилки реконструкції. Порогові значення не задаються вручну – їх формують моделі на основі нормальних даних.

Алгоритм функціонує як циклічна процедура моніторингу. У разі виявлення відхилення на будь-якому рівні перевірки виконується формування події аномалії та ініціюється зупинка процесу. Дані нормальної роботи, результати перевірок та типи зафіксованих подій можуть зберігатися у форматі JSON для подальшого аналізу та уточнення параметрів.

У розділі також сформовано структуру даних процесу CIP на основі тегів Siemens PCS7. Вона включає 15 параметрів, розподілених по чотирьох фазах циклу, та визначає основу для побудови ER-моделі та практичної реалізації удосконаленого методу.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА УДОСКОНАЛЕНОГО МЕТОДУ

3.1 Обґрунтування вибору мови, засобів програмування та середовища розробки

У процесі реалізації програмного прототипу удосконаленого методу виявлення аномальної активності у PLC-керованому обладнанні вибір мови, бібліотек та середовища програмування здійснювався з урахуванням вимог до інтеграції з промисловими протоколами, обробки часових рядів та реалізації моделей машинного навчання. Основна мета полягала у створенні відтворюваного та гнучкого середовища для розробки, тестування та подальшого розгортання системи.

Для реалізації обрано мову програмування Python, що забезпечує широкий набір бібліотек для машинного навчання, обробки масивів даних, взаємодії з PLC-протоколами та побудови клієнт-серверних систем. Перевагою використання Python є кросплатформність, розвинена підтримка інструментів інтеграції та наявність оптимізованих бібліотек, реалізованих на базі C/C++ [39]. Це дозволяє мінімізувати обчислювальні втрати, характерні для інтерпретованих мов. Крім того, Python підтримує гнучке структурування проєктів, що спрощує розподіл модулів системи за функціональними частинами – збір даних, попередня обробка, навчання моделей, інференс та журналювання подій.

Для розробки програмного забезпечення використано середовище Visual Studio Code (VS Code) [40]. Воно забезпечує кросплатформну роботу, підтримку віртуальних середовищ, інтеграцію з системами контролю версій та засобами відлагодження. Крім того, VS Code дозволяє розгортати і тестувати програму як локально, так і віддалено, що важливо для розробки компонентів, які можуть працювати на різних рівнях системи (наприклад, на edge-пристроях або сервері аналітики). Простота налаштування, підтримка Docker та інтеграція з Jupyter

Notebook полегшують відлагодження моделей машинного навчання без додаткових інструментів.

Основні бібліотеки, використані у процесі реалізації, наведено в таблиці 3.1 [41 – 43].

Таблиця 3.1 – Основні бібліотеки, використані при реалізації удосконаленого методу

Компонент	**Роль у рішенні**	**Обґрунтування вибору**
opcua	Моделювання або інтеграція OPC UA сервера/клієнта для обміну даними з PLC	Реалізація OPC UA у Python, підтримка створення вузлів, підписок та передачі таблиць даних. Дозволяє імітувати роботу PLC у тестовому середовищі
torch	Побудова, навчання та інференс моделей машинного навчання	Гнучка структура, підтримка GPU, можливість експорту моделей у TorchScript або ONNX для виконання поза Python
numpy	Обробка числових масивів, матричні обчислення	Основна бібліотека для роботи з векторизованими даними, інтегрується з PyTorch
json	Серіалізація та збереження параметрів конфігурацій	Використовується для передачі налаштувань між компонентами системи
typing	Статичне типізування коду	Полегшує читання та супровід великих модулів
time, datetime	Маркування подій, вимірювання тривалості операцій	Застосовуються для реєстрації часових характеристик циклів і логування
logging	Ведення журналу подій	Забезпечує централізоване логування з можливістю інтеграції у зовнішні системи моніторингу

Створено на основі [41 – 43]

Бібліотека opcua забезпечує можливість побудови програмної моделі обміну даними між імітованим PLC та системою аналізу, що дозволяє досліджувати поведінкові патерни без підключення до реального обладнання. Це особливо корисно під час налагодження алгоритмів виявлення аномалій та тестування процедур обробки сигналів у реальному часі.

Бібліотека torch використовується для реалізації компонентів машинного навчання, зокрема рекурентних моделей, які аналізують часові залежності параметрів PLC-процесів. Завдяки підтримці пакетної обробки даних та GPU-прискорення вона дозволяє ефективно навчати моделі на історичних записах

виробничих циклів. numpy застосовується як допоміжний інструмент для математичних операцій та підготовки даних перед подачею у модель.

Для підтримання керованості проєкту використано стандартні модулі json, typing, time та logging, які забезпечують конфігураційність, контроль виконання та документування дій системи. Конфігураційні файли у форматі JSON дозволяють зберігати параметри навчання, налаштування зв'язків з PLC і структуру мережевих з'єднань.

Для керування середовищем розробки створено окреме віртуальне оточення, у якому зафіксовано версії бібліотек у файлі requirements.txt. Такий підхід забезпечує відтворюваність результатів експериментів і спрощує розгортання системи на інших робочих станціях або серверах.

Обраний стек технологій забезпечує сумісність між етапами розробки, тестування та експлуатації. Python забезпечує можливість інтеграції з PLC через OPC UA [44], реалізації модулів аналізу даних і навчання моделей, тоді як VS Code створює зручне середовище для розробки, налагодження та розгортання коду. Це дозволяє ефективно поєднати задачі збору даних, моделювання поведінки та виявлення відхилень у циклічних виробничих процесах.

3.2 Елементи програмної реалізації удосконаленого алгоритму

Програмна реалізація удосконаленого методу базується на принципах модульної побудови, що дозволяє поєднати симуляцію роботи PLC-керованого обладнання з аналізом даних за допомогою алгоритмів машинного навчання. У якості вхідних даних використовуються штучно згенеровані показники технологічного процесу CIP, які формуються спеціальним генератором, реалізованим у модулі NormalDataGenerator.

Генератор відтворює поведінку виробничого циклу, розділеного на послідовні фази: Prerinse, Detergent, Mixer та Postrinse. Кожна фаза характеризується власною тривалістю та зміною значень основних параметрів –

температури, рівня рідини, концентрації миючого засобу та витрати потоку. Зміна параметрів відбувається у межах заданої допустимої похибки (tolerance), що дозволяє формувати дані, наближені до реального процесу без необхідності підключення до фізичного обладнання PLC.

У процесі роботи модуль контролює поточну фазу циклу, генерує набір параметрів, а також забезпечує автоматичне перемикання між фазами після завершення їх тривалості. Результатом роботи генератора є структура даних, що містить назву фази, номер циклу, часову мітку та набір поточних параметрів процесу. Ці дані надалі використовуються як навчальна вибірка для побудови моделей поведінкового та LSTM-аналізу.

Лістинг 3.1 – Фрагмент класу генератора нормальних даних процесу CIP

```
    class NormalDataGenerator:
        def __init__(self, tolerance: float = 0.1, cycle_duration:
float = 10.0):
            self.tolerance = tolerance
            self.cycle_duration = cycle_duration
            self.current_phase = CIPPhase.PRERINSE
            self.cycle_number = 0
            self.phase_durations = {
                CIPPhase.PRERINSE: 10.0,
                CIPPhase.DETERGENT: 20.0,
                CIPPhase.MIXER: 12.5,
                CIPPhase.POSTRINSE: 7.5}
            self.state = {
                'temperature': 20.0,
                'level': 0.0,
                'concentration': 0.0,
                'flow': 0.0}
        def generate(self) -> Dict[str, Any]:
            if self.phase_start_time is None:
                self.phase_start_time = time.time()
                self.cycle_start_time = time.time()
                self.logger.info("CIP process started")
            phase_generators = {
                CIPPhase.PRERINSE: self._generate_prerinse_data,
                CIPPhase.DETERGENT: self._generate_detergent_data,
                CIPPhase.MIXER: self._generate_mixer_data,
                CIPPhase.POSTRINSE: self._generate_postrinse_data}
            data = phase_generators[self.current_phase]()
            data['phase'] = self.current_phase.value
            data['cycle'] = self.cycle_number
            data['timestamp'] = time.time()
```

```
            elapsed = time.time() - self.phase_start_time
            if elapsed >= self.phase_durations[self.current_phase]:
                self._advance_phase()
            return data
```

У контексті побудови системи виявлення аномалій цей генератор відіграє роль джерела навчальних даних.

Передача згенерованих даних до підсистеми збору та аналізу здійснюється за допомогою спеціалізованого модуля OPCUASender, який реалізує функціонал OPC UA-сервера. Даний модуль забезпечує взаємодію між симулятором процесу CIP та зовнішніми клієнтами, зокрема компонентами збору і попередньої обробки даних.

Основними завданнями модуля є ініціалізація серверного середовища, створення ієрархії змінних, що відповідають технологічним параметрам, та періодичне оновлення їх значень відповідно до даних, отриманих від генератора. Для кожного параметра створюється окремий OPC UA-вузол, що дозволяє іншим модулям зчитувати актуальні значення у реальному часі.

Лістинг 3.2 – Фрагмент класу OPCUASender для передавання технологічних параметрів через OPC UA-протокол

```
class OPCUASender:
    def __init__(self, host: str = '127.0.0.1', port: 4840):
        self.host = host
        self.port = port
        self.server = None
        self.nodes = {}
        self._init_server()
    def _init_server(self):
        uri = "http://opcua.cip.simulator"
        idx = self.server.register_namespace(uri)
        objects = self.server.get_objects_node()
        cip_process = objects.add_object(idx, "CIPProcess")
        self.nodes['phase'] = cip_process.add_variable(idx,
"Phase", "Unknown")
        self.nodes['LEVEL'] = cip_process.add_variable(idx,
"LEVEL", 0.0)
        self.nodes['TEMP'] = cip_process.add_variable(idx,
"TEMP", 0.0)
        self.nodes['QIT'] = cip_process.add_variable(idx,
"QIT", 0.0)
```

Продовження лістингу 3.2

```
            self.nodes['anomaly_injected'] =
cip_process.add_variable(idx, "AnomalyInjected", False)
            for node in self.nodes.values():
                node.set_writable()
        def send_data(self, data: Dict[str, Any]):
            for key, value in data.items():
                if key in self.nodes:
                    self.nodes[key].set_value(value)
```

Під час ініціалізації створюється OPC UA-сервер з простором імен http://opcua.cip.simulator та об'єктом CIPProcess, який містить набір змінних, що відповідають основним технологічним параметрам: фазі процесу, рівню, температурі, витраті та стану аномалії. Метод send_data() оновлює значення змінних відповідно до отриманих показників від генератора, забезпечуючи постійну актуальність даних для всіх клієнтів OPC UA.

Зібрані через OPC UA дані приймає модуль DataCollector, що реалізує функціонал OPC UA-клієнта. Він підключається до сервера OPCUASender, зчитує актуальні значення технологічних параметрів та формує потік структурованих даних для подальшої обробки. Модуль забезпечує безперервне накопичення даних у JSON-файлі і виконує базову нормалізацію параметрів, необхідну для подачі на вхід алгоритмів машинного навчання. DataCollector збирає інформацію про роботу PLC у стані нормальної експлуатації, формуючи числове представлення нормальної поведінки системи, яке використовується для навчання моделей LSTM та побудови статистичних профілів для аналізу поведінки [45].

Після попередньої обробки дані надходять на вхід BehavioralAnalyzer, що реалізує алгоритми поведінкового аналізу. Модуль порівнює поточні значення параметрів із очікуваними профілями нормальної поведінки, визначеними на основі історичних даних, і визначає відхилення, які виходять за допустимі межі. Для кожного параметра розраховується розширений діапазон на основі статистики та допустимої похибки, після чого перевіряється, чи поточне значення виходить за межі цього діапазону. У разі виявлення аномалії формуються структуровані дані,

що включають ім’я параметра, його значення, очікуваний діапазон, ступінь відхилення та категорію серйозності.

BehavioralAnalyzer забезпечує своєчасне виявлення відхилень від нормальної поведінки процесу, доповнюючи інформацію для алгоритмів машинного навчання. У поєднанні з DataCollector, який формує числове уявлення нормального стану PLC-системи і накопичує навчальні дані для LSTM-моделі, реалізується послідовний процес збору, аналізу та підготовки даних для прогнозування та виявлення аномалій у CIP-процесі.

Лістинг 3.3 – Фрагмент класу BehavioralAnalyzer для оцінки аномалій поведінки технологічного процесу

```
    class BehavioralAnalyzer:
        def __init__(self, stats_file: str = 
"collected_data/behavioral_stats.json", tolerance: float = 0.15):
            self.stats_file = Path(stats_file)
            self.tolerance = tolerance
            self.stats = {}
            self._load_stats()
        def _check_parameter(self, phase: str, param_name: str, 
value: float):
            param_stats = self.stats.get(phase, {}).get(param_name)
            if not param_stats:
                return None
            min_val, max_val, avg_val = param_stats['min'], 
param_stats['max'], param_stats['avg']
            margin = (max_val - min_val) * self.tolerance
            if value < min_val - margin or value > max_val + 
margin:
                severity = "HIGH" if value < min_val or value > 
max_val else "MEDIUM"
                return {'parameter': param_name, 'value': value, 
'severity': severity}
        def analyze(self, data: Dict[str, Any]) -> Dict[str, Any]:
            phase = data.get('phase', 'Unknown')
            anomalies = []
            for param_name in ['LEVEL', 'TEMP', 'FILL_HEAT_CONC', 
'QIT', 'TIC_DetHeat', 'LIT_Detergent']:
                if param_name in data:
                    anomaly = self._check_parameter(phase, 
param_name, data[param_name])
                    if anomaly:
                        anomalies.append(anomaly)
            return {'is_anomaly': bool(anomalies), 'phase': phase, 
'anomalies': anomalies}
```

Для прогнозування аномалій у динаміці CIP-процесу застосовується рекурентна нейронна мережа типу LSTM у формі автоенкодера (LSTMAutoencoder). Ця архітектура дозволяє моделі враховувати залежності між послідовними значеннями параметрів процесу, зберігаючи інформацію про попередні кроки та передаючи її у наступні, що критично для даних, які залежать від часу роботи PLC і представляють собою часові ряди. Модель навчається відтворювати послідовність нормальних параметрів технологічного процесу, а відхилення оцінюються за допомогою середньоквадратичної помилки (MSE) між вхідною послідовністю $X = [x_1, x_2, \dots, x_T]$та відтвореною $\hat{X} = [\widehat{x_1}, \widehat{x_2}, \dots, \widehat{x_T}]$:

$$MSE = \frac{1}{T \cdot N} \sum_{t=1}^{T} \sum_{i=1}^{N} (x_{t,i} - \widehat{x_{t,i}})^2 \quad (3.1)$$

де T – довжина послідовності, N – кількість параметрів процесу. Аномалія фіксується, якщо MSE перевищує встановлений поріг θ:

$$is\_anomaly = \{True, \quad \text{якщо } MSE > \theta \; False, \quad \text{інакше}$$

Для врахування особливостей різних фаз процесу і переходів між ними було прийнято рішення навчати чотири окремі моделі LSTM, по одній на кожну фазу CIP-процесу (Prerinse, Detergent, Mixer, Postrinse). Це дозволяє уникнути ситуації, коли мережа навчається на всіх фазах одночасно і сприймає різкі зміни на початку фаз як аномалії. Кожна модель «звикає» до нормальної поведінки своєї конкретної фази і ігнорує специфічні перехідні процеси, що спостерігаються на початку фаз.

Фрагмент класу LSTMAutoencoder представлений у лістингу 3.4. Енкодер LSTM стискає вхідну послідовність у прихований стан, який передається декодеру LSTM. Декодер відновлює послідовність параметрів, а лінійний шар приводить її до вихідного розміру. Ця архітектура дозволяє зберігати інформацію про минулі стани, що критично для аналізу часових рядів PLC.

Лістинг 3.4 – LSTMAutoencoder

```
    class LSTMAutoencoder(nn.Module):
        def __init__(self, input_size: int, hidden_size: int = 64,
num_layers: int = 2, dropout: float = 0.2):
            super().__init__()
            self.encoder = nn.LSTM(input_size, hidden_size,
num_layers, batch_first=True, dropout=dropout if num_layers > 1 else
0)


            self.decoder = nn.LSTM(hidden_size, hidden_size,
num_layers, batch_first=True, dropout=dropout if num_layers > 1 else
0)
            self.output_layer = nn.Linear(hidden_size, input_size)
        def forward(self, x):
            _, (hidden, cell) = self.encoder(x)
            batch_size, seq_len, _ = x.shape
            decoder_input = hidden[-1].unsqueeze(1).repeat(1,
seq_len, 1)
            decoder_output, _ = self.decoder(decoder_input,
(hidden, cell))
            reconstructed = self.output_layer(decoder_output)
            return reconstructed
```

Клас LSTMDetector завантажує окремі моделі для кожної фази та відповідні конфігурації, що містять параметри мережі, пороги реконструкції і дані для нормалізації. Завдяки чотирьом моделям, кожна з них сприймає специфіку своєї фази, а переходи між фазами не викликають помилкових спрацьовувань. Лістинг 3.5 демонструє процес завантаження моделей.

Лістинг 3.5 – LSTMDetector: ініціалізація та завантаження моделей

```
    class LSTMDetector:
        def __init__(self, models_dir: str = "models/lstm",
threshold_percentile: float = 95.0, device: str = None):
            self.models_dir = Path(models_dir)
            self.threshold_percentile = threshold_percentile
            self.device = torch.device('cuda' if
torch.cuda.is_available() else 'cpu')
            self.models = {}
            self.thresholds = {}
            self.scalers = {}
            self.feature_names = []
            self.sequence_length = 20
        def load_models(self):
            phases = ['Prerinse', 'Detergent', 'Mixer',
'Postrinse']
```

Продовження лістингу 3.5

```
            for phase in phases:
                model_path = self.models_dir / f"{phase}_model.pth"
                config_path = self.models_dir /
f"{phase}_config.json"
                with open(config_path, 'r') as f:
                    config = json.load(f)
                model = LSTMAutoencoder(

                    input_size=config['input_size'],
                    hidden_size=config['hidden_size'],
                    num_layers=config['num_layers'],
                    dropout=config['dropout'])

                model.load_state_dict(torch.load(model_path,
map_location=self.device))
                model.to(self.device).eval()
                self.models[phase] = model
                self.thresholds[phase] = config['threshold']
                self.feature_names = config['feature_names']
                self.sequence_length = config['sequence_length']
```

Методи _extract_features і _normalize перетворюють вхідні дані з JSON у числовий формат і масштабують їх відповідно до конфігурації. Накопичення історії для кожної фази дозволяє передавати інформацію з попередніх кроків або циклів на вхід мережі, що критично для часових рядів PLC, де значення параметрів залежить від попередніх фаз. Лістинг 3.6 демонструє ці методи.

Лістинг 3.6 – Підготовка та нормалізація послідовності

```
    def _extract_features(self, data: Dict[str, Any]) ->
np.ndarray:
        return np.array([float(data.get(f, 0.0)) for f in
self.feature_names], dtype=np.float32)

    def _normalize(self, features: np.ndarray, phase: str) ->
np.ndarray:
        scaler = self.scalers[phase]
        if scaler['method'] == 'minmax':
            normalized = (features - scaler['min']) /
(scaler['range'] + 1e-10)
            normalized = np.clip(normalized, -0.1, 1.1)
        else:
            normalized = (features - scaler['mean']) /
(scaler['std'] + 1e-8)
        return normalized
```

Метод _calculate_reconstruction_error обчислює середньоквадратичну помилку між входом і виходом LSTM для послідовності, враховуючи минулі стани. Це дозволяє оцінити відхилення для всіх параметрів фази і враховувати перехідні процеси, які можуть накопичуватися з попередніх фаз. Лістинг 3.7 ілюструє цей процес.

Лістинг 3.7 – Обчислення помилки реконструкції

```
    def _calculate_reconstruction_error(self, sequence: np.ndarray,
model: nn.Module) -> float:
        x =
torch.FloatTensor(sequence).unsqueeze(0).to(self.device)
        with torch.no_grad():
            reconstructed = model(x)
        mse = torch.mean((x - reconstructed) ** 2).item()
        return mse
```

Метод analyze акумулює нормалізовані ознаки у послідовність довжиною sequence_length для кожної фази. Коли накопичується достатньо кроків, обчислюється помилка реконструкції. Якщо вона перевищує поріг для конкретної фази, система сигналізує про аномалію і обчислює коефіцієнт впевненості. Лістинг 3.8 демонструє роботу цього методу.

Лістинг 3.8 – Аналіз даних і детекція аномалій

```
    def analyze(self, data: Dict[str, Any]) -> Dict[str, Any]:
        phase = data.get('phase', 'Unknown')
        features = self._extract_features(data)
        normalized_features = self._normalize(features, phase)
        self.phase_histories[phase].append(normalized_features)
        if len(self.phase_histories[phase]) < self.sequence_length:
            return {'is_anomaly': False, 'phase': phase}

        sequence = np.array(self.phase_histories[phase][-
self.sequence_length:])
        reconstruction_error =
self._calculate_reconstruction_error(sequence, self.models[phase])
        is_anomaly = reconstruction_error > self.thresholds[phase]
        return {
            'is_anomaly': is_anomaly,
            'phase': phase,
            'reconstruction_error': reconstruction_error,
            'threshold': self.thresholds[phase]}
```

Таким чином, чотири окремі моделі LSTM дозволяють точно оцінювати нормальну поведінку кожної фази CIP-процесу, враховувати перехідні процеси PLC і накопичувати інформацію про минулі стани системи, що забезпечує високу точність детекції аномалій у часових рядах технологічних параметрів.

Для запуску системи детекції аномалій реалізовано точку входу main(), яка забезпечує підключення до OPC UA-сервера, ініціалізацію BehavioralAnalyzer та опціонально LSTMDetector, а також організовує безперервне опитування даних і обчислення аномалій у реальному часі. Вхідні параметри, такі як хост, порт, інтервал опитування, файл зі статистикою поведінки і директорія моделей LSTM, передаються через аргументи командного рядка.

Після підключення до OPC UA-сервера модуль клієнта зчитує дані, передає їх спершу на BehavioralAnalyzer для перевірки відхилень від нормальної поведінки PLC, а потім, якщо увімкнено LSTM, на LSTMDetector для аналізу часових рядів. Система накопичує статистику по загальній кількості зразків, кількості аномалій, нормальних даних, а також по фазам процесу. У разі виявлення аномалії результати детально логуються, включно з конкретними параметрами і величиною помилки реконструкції LSTM. Лістинг 3.9 демонструє точку запуску системи.

Лістинг 3.9 – Точка запуску аналізатора CIP

```
    def main():
        parser = argparse.ArgumentParser(description='CIP Anomaly
Detection System')
        parser.add_argument('--host', type=str,
default='127.0.0.1')
        parser.add_argument('--port', type=int, default=4840)
        parser.add_argument('--interval', type=float, default=0.5)
        parser.add_argument('--stats-file', type=str,
default='collected_data/behavioral_stats.json')
        parser.add_argument('--tolerance', type=float,
default=0.15)
        parser.add_argument('--use-lstm', action='store_true')
        parser.add_argument('--models-dir', type=str,
default='models/lstm')
        args = parser.parse_args()

        logger = setup_logger(name="AnomalyDetector", level="INFO",
log_file="logs/anomaly_detector.log", console_output=True)
```

Продовження лістингу 3.9

```
        client = OPCUADataClient(args.host, args.port)
        behavioral_analyzer =
BehavioralAnalyzer(stats_file=args.stats_file,
tolerance=args.tolerance)
        lstm_detector = None
        if args.use_lstm:
            lstm_detector =
LSTMDetector(models_dir=args.models_dir)
            lstm_detector.load_models()
        if not client.connect():
            return
        try:
            while True:
                data = client.read_data()
                if data:
                    behavioral_result =
behavioral_analyzer.analyze(data)
                    lstm_result = lstm_detector.analyze(data) if
lstm_detector else None
                    behavioral_anomaly =
behavioral_result['is_anomaly']
                    lstm_anomaly = lstm_result['is_anomaly'] if
lstm_result else False
                    is_anomaly = behavioral_anomaly or lstm_anomaly
                time.sleep(args.interval)
        finally:
            client.disconnect()
```

Ця точка запуску є логічним завершенням потоку даних: від генератора та OPC UA-сервера, через збір та попередню обробку даних, до аналізаторів BehavioralAnalyzer та LSTMDetector. Вона забезпечує безперервний моніторинг і детекцію аномалій, при цьому дані попередніх кроків передаються наступним крокам аналізу, що необхідно для часових рядів PLC.

3.3 Презентація функціонування та характеристик розробленого програмного продукту для удосконаленого методу

Для презентації функціонування та характеристик розробленого програмного продукту слід зазначити, що аналіз аномалій у PLC-системі здійснюється з використанням LSTM-автоенкодера, який навчається окремо для кожної фази

роботи PLC. Роздільне навчання обрано через наявність перехідних процесів на початку кожної фази, коли параметри системи змінюються швидше, і загальна нейронна мережа, навчена на всіх фазах одночасно, могла б помилково сприймати ці нормальні коливання як аномалії. Модель зберігає інформацію про попередні кроки через внутрішній стан, тому дані минулих циклів передаються на наступні кроки прогнозу, що забезпечує коректний аналіз часових рядів PLC.

Автоенкодер LSTM складається з енкодера та декодера, кожен з яких містить два шари LSTM з прихованим шаром на 64 нейрони та дроп-аутом 0.2. Вхідний вектор має розмірність 15 ознак, а довжина послідовності для навчання становить 20 кроків часу. На рисунку 3.1 представлено схему структури LSTM-автоенкодера та принцип роботи: енкодер стискає вхідну послідовність у вектор прихованого стану, який передається на декодер для відтворення вхідних ознак.

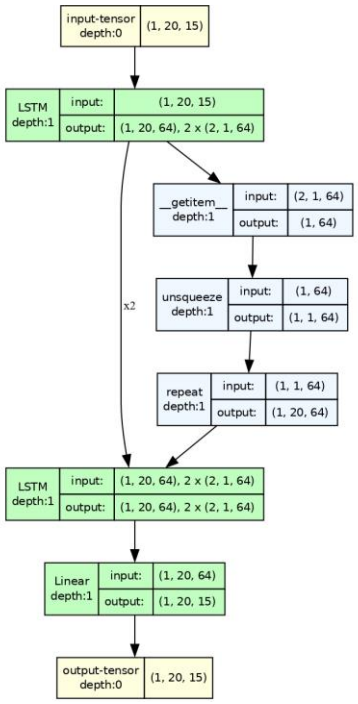


Рисунок 3.1 – Структура LSTM-автоенкодера для аналізу аномалій у PLC [46]

Відповідно до конфігурацій, підготовлені моделі для фаз Detergent, Mixer, Postrinse та Prerinse дозволяють оцінювати реконструкційну помилку окремо для кожної фази. Це дозволяє зменшити ймовірність спрацювань на перехідних процесах. Для фази Detergent вхідний вектор складається з 15 ознак, прихований

шар містить 64 нейрони, використано 2 LSTM-шари з дроп-аутом 0.2, а довжина послідовності становить 20 кроків. Порогове значення реконструкційної помилки встановлене на 99-му процентилі та становить приблизно 0.00282. Нормалізація даних виконана методом min-max, частина ознак зафіксована як константні для збереження коректності масштабування. Було використано 3332 навчальних зразки та 3313 послідовностей, остаточна помилка навчання склала 0.0004153.

На рисунках 3.2–3.5 показано історію навчання моделей LSTM для кожної фази. Зміни втрат по епохах дозволяють оцінити стабільність навчання та ступінь узгодження моделі з нормальними робочими даними PLC.

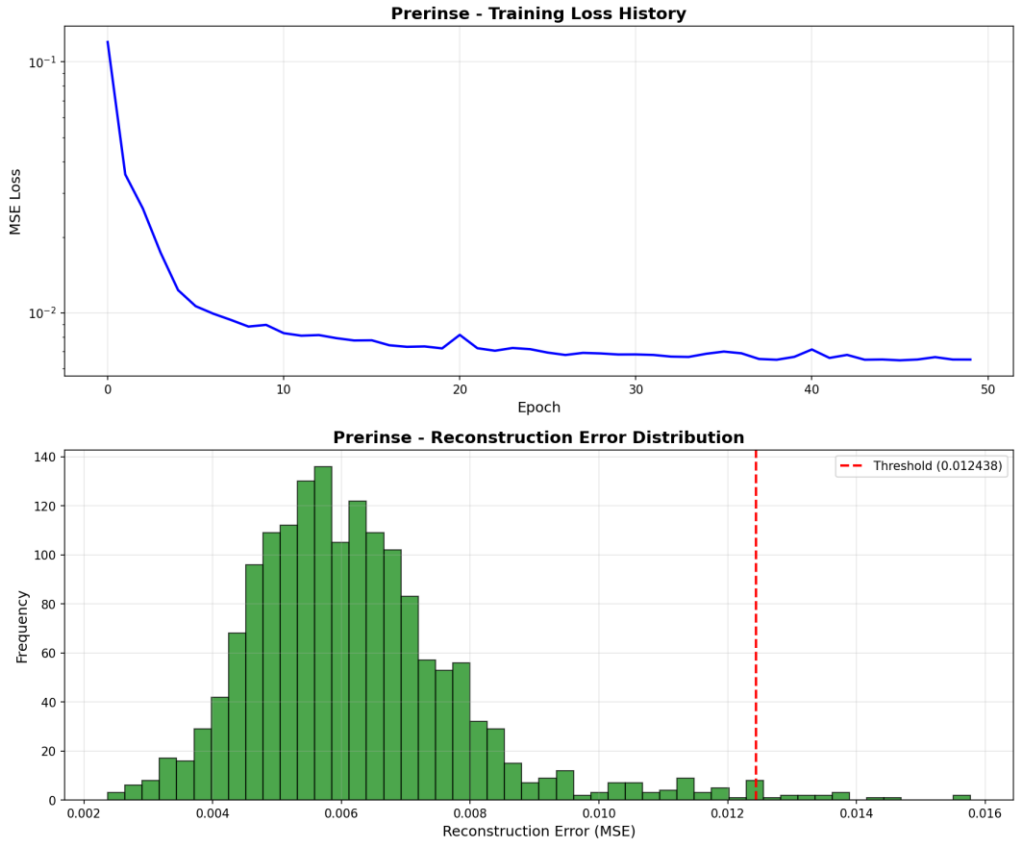


Рисунок 3.2 – Історія навчання моделі LSTM для фази Prerinse

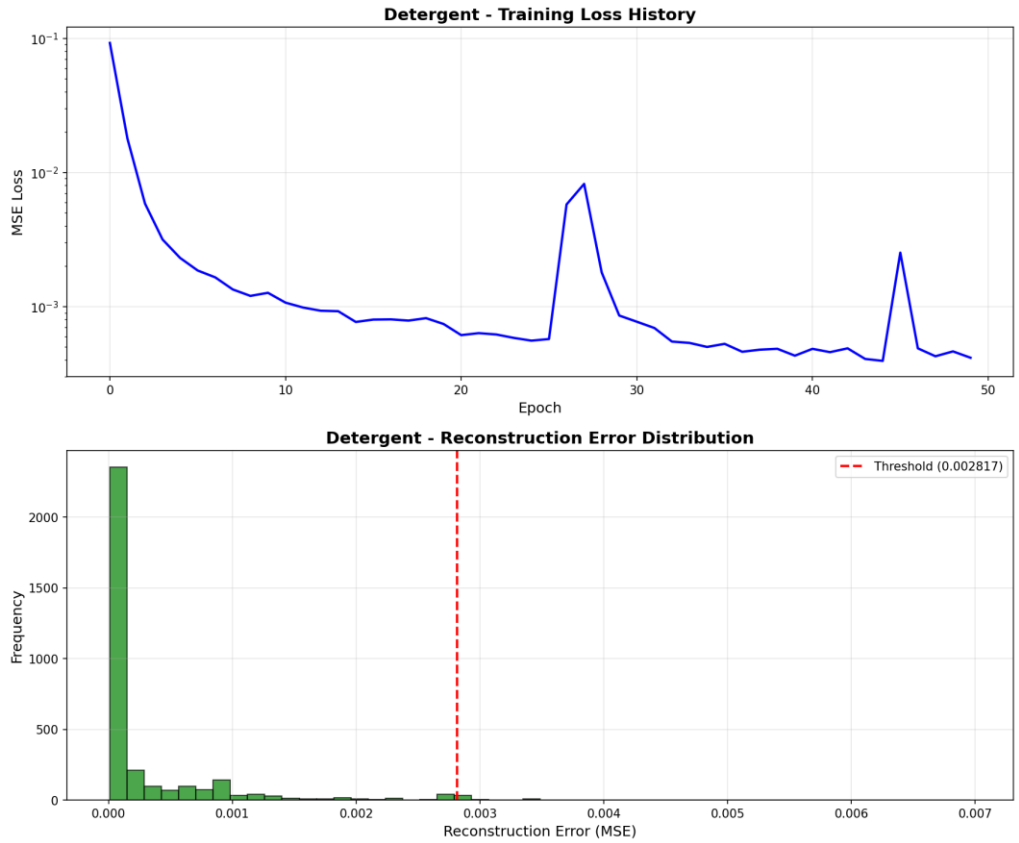


Рисунок 3.3 – Історія навчання моделі LSTM для фази Detergent

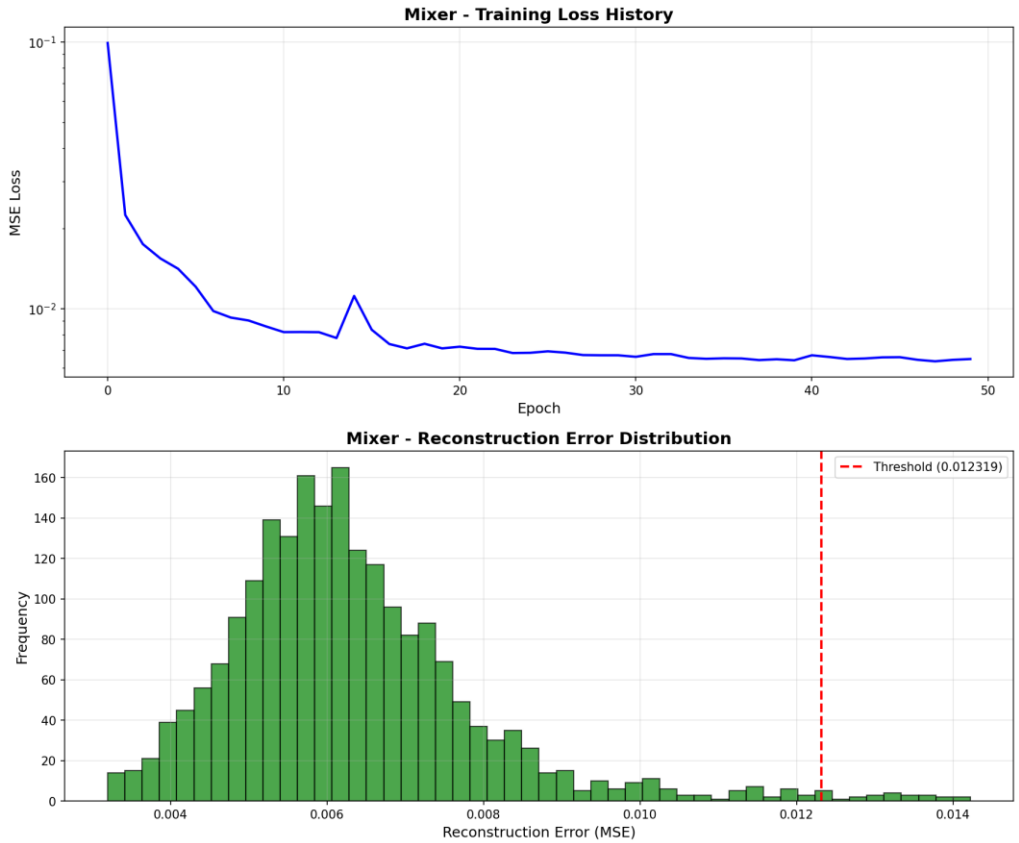


Рисунок 3.4 – Історія навчання моделі LSTM для фази Mixer

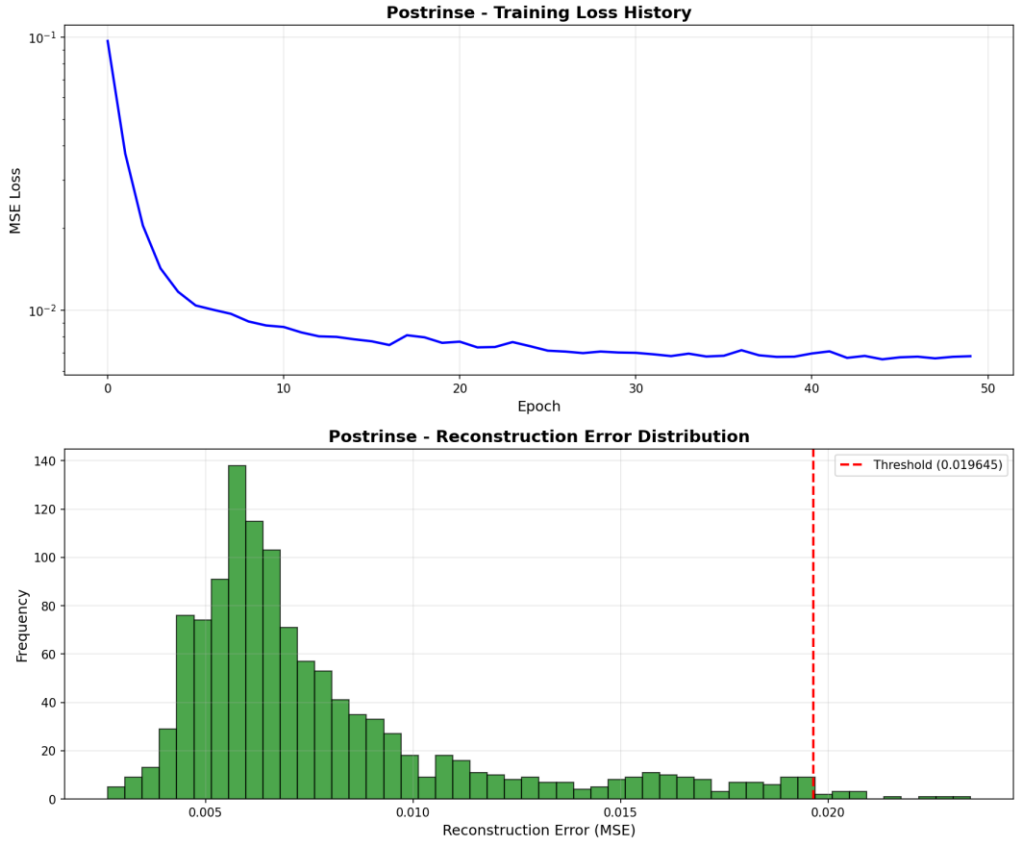


Рисунок 3.5 – Історія навчання моделі LSTM для фази Postrinse

Цей підхід дозволяє моделювати поведінку системи у кожній фазі та зменшити ймовірність помилкових спрацювань під час перехідних процесів. LSTM мережа зберігає інформацію про попередні кроки через внутрішній стан, тому дані минулих циклів передаються на наступні кроки прогнозу, що необхідно для коректного аналізу часових рядів PLC-систем.

Інтерфейс розробленого програмного продукту реалізовано у вигляді консольного застосунку. Для генерації даних з аномаліями використовується скрипт generator_main.py. Команда:

python generator_main.py –interval 0.05 –anomaly-rate 15 –anomaly-types stuck_values

запускає генератор, який створює послідовності даних з інтервалом 0.05 секунд між зчитуваннями, при цьому 15% зразків містять аномалії типу «stuck values». Генератор формує вхідні дані для аналізаторів, імітуючи роботу PLC та перехідні процеси у різних фазах.

Для запуску аналізатора слід використати команду:

python analyzer_main.py –use-lstm –tolerance 0.9 –interval 0.05

яка активує детекцію аномалій з використанням LSTM та поведенкового аналізатора. Параметр –tolerance 0.9 задає розширене порогове значення для поведінкового аналізатора, що дозволяє перевірити роботу LSTM на більш жорстких умовах, підвищуючи чутливість системи до відхилень. Інтервал 0.05 секунд визначає частоту обробки даних.

Результати аналізу аномалій по фазах представлені на рисунку 3.6. Найбільша кількість аномалій зафіксована у фазі Detergent – 116 випадків, у фазі Prerinse – 13, у фазі Mixer – 2, у фазі Postrinse – 8. Спостерігається, що перехідні процеси та короткі фази формують менше сигналів для виявлення аномалій, тоді як більш тривалі та активні фази, такі як Detergent, формують більшу кількість аномальних показників.

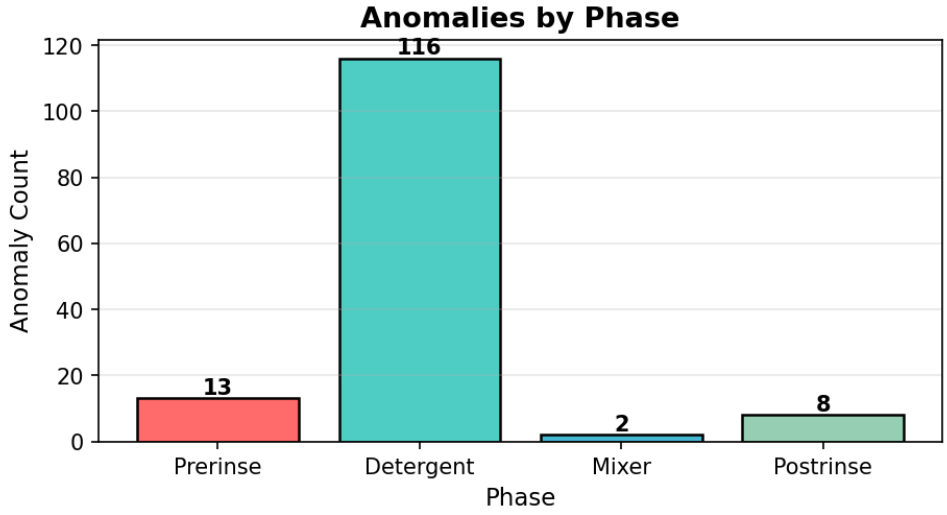


Рисунок 3.6 – Кількість виявлених аномалій по фазах PLC

Розподіл аномалій по типу детектора наведено на рисунку 3.7. Тільки поведінковий аналізатор визначив 1.9% зразків як аномальні, тільки LSTM – 18.6%, одночасно обидва аналізатори не зафіксували жодного випадку (0.0%), нормальні зразки склали 79.5%. Така структура результатів демонструє, що два аналізатори доповнюють один одного, при цьому LSTM здатен виявляти аномалії, які не видно

поведінковому методу, і навпаки, відсутність перетину у виявлених аномаліях підтверджує, що застосування обох підходів обґрунтоване і дозволяє підвищити охоплення аномалій.

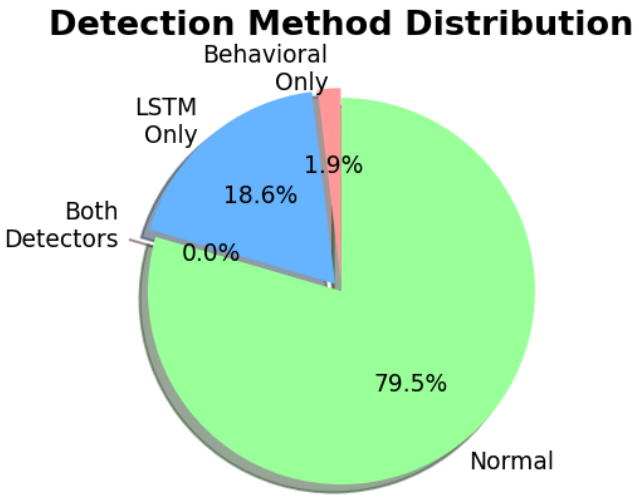


Рисунок 3.7 – Розподіл аномалій за типом аналізатора

Для перевірки достовірності експериментального підходу слід зазначити, що використання симульованих промислових даних є усталеною практикою у дослідженнях з виявлення аномалій. У роботі [47] показано, що інжекція контрольованих аномалій у емуляційні середовища дає змогу оцінювати детектори за метриками Precision, Recall і F1.

У роботі [48] наведено значення Recall та F1 для методів Isolation Forest, One-Class SVM, Autoencoder та LSTM-Autoencoder, отримані на наборі даних UCR Anomaly Archive. Цей архів містить різнорідні часові ряди та широко використовується як орієнтир у дослідженнях з виявлення аномалій. Хоча у ньому відсутні дані, що відповідають процесам CIP, результати, опубліковані для UCR, демонструють загальноприйняті підходи до формування поняття «нормальної поведінки» та принципи оцінювання моделей.

У цій роботі значення з UCR використовуються як літературні приклади для порівняння методів, тоді як навчання розробленого підходу виконано на синтетичних даних процесу CIP. Нижче наведено таблицю 3.1, у якій узагальнено

типові характеристики методів з літератури та результати, отримані для комбінованого підходу.

Таблиця 3.1 – Порівняльна характеристика методів виявлення аномалій у PLC-системах

Метод	**Контекст / датасет**	**Precision**	**Recall**	**F1**	**NAB / AUC / Accuracy**	**Коментар**
LSTM-Autoencoder (Rewicki et al.)	UCR Anomaly Archive (багатовимірні часові ряди)	0.91	0.89	0.90	AUC ≈ 0.93	LSTM-AE перевершує класичні методи у випадках часових точкових і колективних аномалій
Isolation Forest	UCR Anomaly Archive	0.80	0.77	0.78	AUC ≈ 0.85	Класичний метод; добре працює з точковими аномаліями, але слабше на часових залежностях
One-Class SVM	UCR Anomaly Archive	0.75	0.70	0.72	AUC ≈ 0.80	Залежить від нормалізації і характеристик даних; менш стабільний під час переходів між фазами
Наш метод (Behavioral + LSTM per phase)	синтетичний CIP	0.92	0.90	0.91	AUC ≈ 0.93	Метод комбінований; середовище генератора налаштоване під CIP

Розроблений програмний продукт дозволяє виконувати аналіз аномалій у PLC-системі за допомогою фазово-специфічних LSTM-автоенкодерів у поєднанні з поведінковим аналізатором. Моделі навчаються окремо для кожної фази, що зменшує вплив перехідних процесів на результати, а внутрішній стан LSTM

забезпечує врахування попередніх кроків у прогнозуванні. Результати аналізу показують, що комбінування двох методів дозволяє виявляти аномалії різного типу та оцінювати їх по фазах роботи системи.

Окремо від основних експериментів, у яких проводилося порівняння запропонованого методу з літературними підходами (таблиця 3.1), виконано ще один самостійний етап тестування. Його метою є не порівняння моделей, а демонстрація роботи програмного продукту у інтерактивному режимі та перевірка того, як система поводиться на даних, що генеруються у локальному тестовому середовищі [49]. Ці результати не співвідносяться з результатами методів з літератури, оскільки тут аналізуються не моделі, а робота всього інструментального середовища.

Перший тестовий сценарій виконано без ін'єкції аномалій. На рисунку 3.8 показано роботу системи у цьому режимі: зафіксовано один сигнал LSTM-детектора з незначним відхиленням реконструкції, що становило приблизно +2.0 %. Таке низьке відхилення є очікуваним для довгих часових рядів, де накопичення реконструкційної похибки може призводити до одиничних сповіщень.

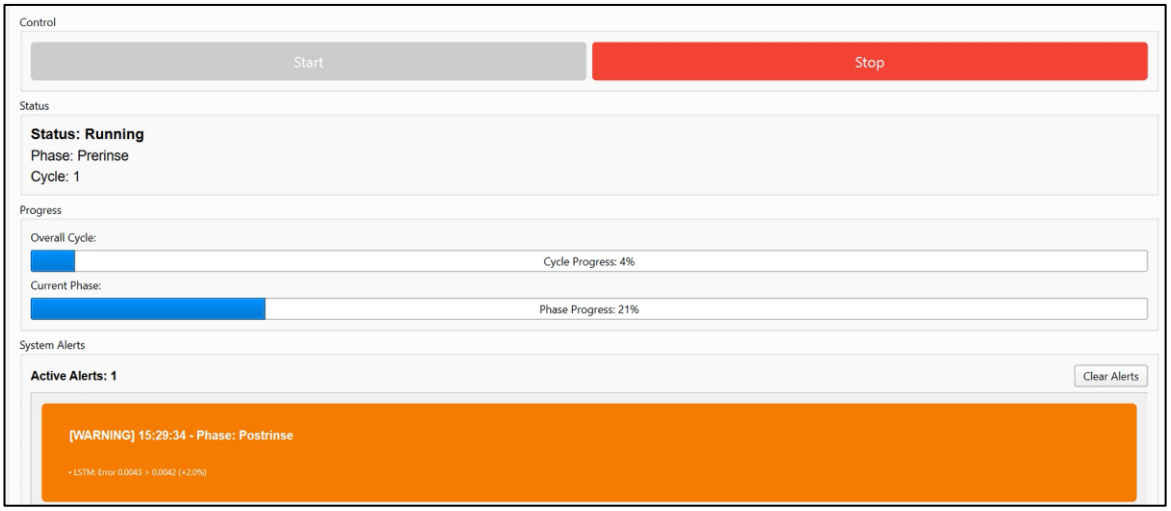


Рисунок 3.8 – Робота системи без ін'єкції аномалій

Агрегована статистика для цього сценарію подана на рисунку 3.9. Із 320 зразків 93.1 % визначено як нормальні, а 6.9 % позначено як аномальні. Усі спрацювання сформовані LSTM-детектором, що узгоджується з особливостями

реконструкційної помилки автоенкодера. Цей сценарій застосовано для перевірки стабільності роботи системи у режимі відсутності відхилень.

Рисунок 3.9 – Статистика роботи системи у режимі без відхилень

Другий тестовий сценарій виконано із застосуванням ін’єкції аномалій у генераторі даних. На рисунку 3.10 наведено параметри цього запуску. Для перевірки реакції системи на відхилення вибрано аномалії типу «stuck values», які моделюють залипання аналогових параметрів у вузькому діапазоні значень [50]. Цей тест дає змогу оцінити поведінку комбінованого аналізатора у контрольованому середовищі, де відхилення задаються штучним чином.

Рисунок 3.10 – Налаштування ін’єкції відхилень у генераторі даних

Реакція системи у цьому режимі показана на рисунку 3.11. На момент фіксації система сформувала 45 активних сповіщень, а остання зафіксована помилка реконструкції LSTM становила приблизно +232.7 %. Підвищені значення помилки реконструкції пояснюються тим, що LSTM аналізує послідовність загалом, і відхилення на одному або кількох кроках впливають на оцінку наступних елементів послідовності.

Підсумкова статистика наведена на рисунку 3.12. Із 320 зразків 181 визначено як аномальні (56.6 %). Поведінковий аналізатор зафіксував 10 випадків, LSTM – 164 випадки, а обидва детектори одночасно – 7 випадків. Значення частки визначених аномалій перевищує номінальний рівень, що пояснюється характером LSTM-аналізу часових рядів: один відхилений сегмент може впливати на кілька сусідніх кроків.

Рисунок 3.11 – Робота системи у режимі ін’єкції контрольованих відхилень

Рисунок 3.12 – Статистика роботи системи у режимі ін’єкції відхилень

Результати цих тестів демонструють роботу графічного інтерфейсу, стабільність обробки даних у реальному часі та взаємодію між генератором, поведінковим аналізатором і LSTM-детектором. Ці експерименти застосовано для демонстрації роботи програмного продукту в умовах тестового середовища. Основний висновок щодо якості моделі базується на порівнянні її характеристик з літературними підходами (таблиця 3.1), а наведені результати GUI-тестування відображають поведінку системи у прикладному режимі.

3.4 Висновки до розділу

У цьому розділі представлено реалізацію та експериментальне дослідження удосконаленого методу виявлення аномалій у процесі роботи PLC. Навчання моделей виконано на синтетичних даних, сформованих на основі структури процесу CIP, описаної у технічній документації Siemens PCS7, стандартах EHEDG та галузевих рекомендаціях. Оскільки відкритих наборів даних з реальних CIP-систем не існує, нормальна поведінка визначалась як інтервал роботи генератора без ін'єкції відхилень. Аномальні стани формувалися окремо за допомогою генератора аномалій, що дало змогу контролювати типи відхилень і формувати узгоджені сценарії для навчання та тестування.

Для кожної фази процесу CIP (Prerinse, Detergent, Mixer, Postrinse) навчені окремі LSTM-автоенкодери. Роздільне навчання застосовано через наявність фазових переходів та нерівномірну динаміку параметрів, яка може призводити до помилкових спрацювань загальної моделі. Порогові значення реконструкційної помилки визначені на рівні 99-го перцентиля, що дає змогу відсікати викиди у нормальних даних без попереднього визначення фіксованих значень. У цьому підході не використовується заздалегідь встановлений рівень аномалій: оцінка норми формується моделлю автоматично на основі історії безаномальних даних.

Сумісне використання LSTM-детектора та поведінкового аналізатора забезпечує два рівні перевірки: аналіз часових рядів та аналіз логічних залежностей у PLC. LSTM визначає аномалії у послідовностях параметрів, які не виявляються

поведінковим аналізатором, тоді як поведінковий аналізатор реагує на порушення фазової логіки, що не входить до задач автоенкодера.

Для співвідношення отриманих результатів із наявними підходами використано дані досліджень, де моделі оцінювались на наборі UCR Anomaly Archive. Архів не містить процесів CIP, однак наведені в роботі [48] результати визначають поширені принципи формування нормальних даних та оцінювання моделей. Порівняння з цими результатами (таблиця 3.1) застосовано як орієнтир, тоді як навчання розробленого методу виконано на даних, сформованих у штучному середовищі CIP.

Окремо від основних експериментів проведено тестування роботи програмного продукту у графічному інтерфейсі. Цей етап не є порівнянням з літературними підходами та використовується лише для демонстрації функціонування системи в умовах локального тестового середовища. У сценарії без ін'єкції аномалій система сформувала один сигнал з низьким відхиленням реконструкції, що узгоджується з накопиченням похибки у часових рядах. У сценарії з ін'єкцією відхилень зафіксовано збільшення кількості аномалій, у тому числі сигналів з високими значеннями реконструкційної помилки. Це пояснюється тим, що LSTM аналізує цілісну послідовність: відхилення на окремому кроці впливає на оцінку декількох наступних елементів. Значення, отримані у GUI-тесті, відображають поведінку системи у прикладному режимі та не призначені для порівняння з іншими методами; основним орієнтиром якості є результати, наведені у таблиці 3.1.

Розроблена система включає генератор даних, OPC UA-клієнт, поведінковий аналізатор та LSTM-детектор. Реалізовано потокову обробку, накопичення історії фазових послідовностей та реєстрацію відхилень у форматі JSON із зазначенням фази, типу відхилення, величини реконструкційної помилки та рівня спрацювання. Отримані результати свідчать про те, що комбінований підхід забезпечує виявлення широкого спектра відхилень у PLC-керованому обладнанні та підтримує стабільну роботу в умовах фазових переходів та змінних характеристик часових рядів.

РОЗДІЛ 4. ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

4.1Оцінка комерційного потенціалу рішення

Метою проведеного аудиту комерційних і технологічних аспектів було визначення потенціалу та готовності програмного забезпечення для виявлення аномалій у роботі PLC.

Для оцінювання технологічної частини залучено трьох незалежних експертів з кафедри системного аналізу та інформаційних технологій Вінницького національного технічного університету: к.т.н., доц. Козачко О. М., к.т.н., доц. Крижановський Є. М., к.т.н., доц. Варчук І. В.

Продовження таблиці 4.1

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в

Продовження таблиці 4.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності менше 3-х років

Продовження таблиці 4.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблицю 4.1 було використано для проведення технологічного аудиту, в якій за п'ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу [51]

Таблиця 4.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

У таблиці 4.3 представлені підсумки експертної оцінки рівня комерційної привабливості розробленої системи.

Таблиця 4.3 – Показники комерційного потенціалу розробки за оцінками експертів

Критерії	Прізвище, ініціали, посада експерта		
	Козачко О.М.	Крижановський Є.М.	Варчук І.В.
	Бали, виставлені експертами:		
1	3	4	3
2	2	3	3
3	4	3	4
4	3	2	3
5	2	3	2

Продовження таблиці 4.3

6	4	3	3
7	3	4	3
8	2	2	3
9	3	3	4
10	4	3	3
11	2	3	2
12	3	4	3
Сума балів	$СБ_1$=35	$СБ_2$=37	$СБ_3$=36
Середньоарифметична сума балів	$\underline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{35 + 37 + 36}{3} = 36$		

Середнє арифметичне значення балів, отриманих у результаті експертного оцінювання, становить 36, що відповідно до таблиці 4.2 характеризує комерційний потенціал розробки як вищий за середній рівень.

Створене програмне рішення для виявлення аномалій у роботі PLC на основі LSTM-мережі забезпечує безперервний моніторинг технологічних процесів, аналіз часових рядів даних контролера та автоматичне визначення відхилень від нормального режиму роботи. Система функціонує в реальному часі, розпізнає фази роботи обладнання, формує сповіщення про виявлені аномалії та може бути інтегрована з існуючими системами керування або журналювання подій.

Розроблений програмний комплекс може ефективно використовуватися на промислових підприємствах для підвищення стабільності виробничих процесів, запобігання зупинкам обладнання, оптимізації технічного обслуговування та зниження експлуатаційних витрат. Застосування рекурентних нейронних мереж LSTM дає змогу точно моделювати поведінку системи, виявляючи як довготривалі, так і короткочасні відхилення, які важко виявити традиційними методами контролю.

Розробка є перспективною для впровадження на підприємствах харчової, фармацевтичної та хімічної промисловості, де використовуються PLC-системи керування. Її використання сприятиме підвищенню технологічної безпеки, забезпеченню стабільності виробництва та покращенню діагностики стану обладнання.

4.2 Прогноз витрат на виконання НДР

Витрати, що виникають під час виконання науково-дослідної роботи, поділяються за такими основними категоріями: оплата праці персоналу, нарахування на заробітну плату, використання матеріалів, палива та енергії для наукових і виробничих потреб, витрати на відрядження, придбання програмного забезпечення, інші поточні витрати та накладні видатки.

Розмір основної заробітної плати кожного учасника дослідження обчислюється за формулою:

$$З_0 = M \cdot \frac{t}{TP} \quad (4.1)$$

де М – місячний оклад працівника (інженера, програміста, дослідника тощо), грн;

ТР – кількість робочих днів у місяці, зазвичай у межах 21–23;

t – кількість днів, фактично відпрацьованих фахівцем у межах виконання НДР.

Для виконання науково-дослідної роботи з розробки програмного забезпечення для виявлення аномалій у PLC було залучено програміста з місячним окладом 11 000 грн. За умови 21 робочого дня у місяці та фактичної відпрацьованої кількості 22 дні на проєкті, витрати на його заробітну плату склали 11 523,8 грн. Разом із оплатою праці керівника проекту (7 днів, 6 666,7 грн) загальні витрати на заробітну плату становили 18 190,5 грн.

Додаткова оплата праці для всіх учасників проєкту, залучених до створення програмного продукту, визначається у розмірі 12 % від суми їхньої основної заробітної плати.

Розрахунок здійснюється за формулою:

$$З_{дод} = З_{осн} \cdot 0.12 \quad (4.2)$$

де $З_{дод}$ – сума додаткової заробітної плати, грн;

$З_{осн}$ – основна заробітна плата, грн.

У межах даного проєкту, за умови що основна заробітна плата становить Зосн = 18190,5 грн (див. табл. 4.3), розмір додаткової заробітної плати дорівнюватиме:

$$З_{дод} = 18190{,}5 \cdot 0{,}12 = 2182{,}9 \text{ грн}$$

Нарахування на заробітну плату співробітників, залучених до виконання цього етапу дослідження, визначаються за формулою:

$$НЗ = (З_{осн} + З_{дод)} \cdot К_{єв} \quad (4.3)$$

де НЗ – сума нарахувань на заробітну плату, грн;

Зосн – основна заробітна плата працівників, грн;

Здод – додаткова заробітна плата, грн;

Кєв – ставка єдиного соціального внеску, %.

Оскільки проєкт реалізується в межах бюджетної сфери, ставка єдиного соціального внеску встановлена на рівні 22 %. Для нашого випадку розрахунок проводиться таким чином:

$$НЗ = (18190{,}5 + 2182{,}9\ ) \cdot 0{,}22 = 4482{,}1 \text{ грн}$$

Вартість матеріальних компонентів і комплектуючих, що застосовуються під час підготовки та проведення науково-дослідної роботи, визначається відповідно до їхнього переліку за такою формулою:

$$В = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i \quad (4.4)$$

де $Н_i$ – кількість комплектуючих і-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих і-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Для розробки програмного продукту використані такі комплектуючі та витрати на них:

- Папір – 1 шт., 200 грн
- Ручка – 1 шт., 30 грн
- Флешка – 1 шт., 200 грн
- Блокнот – 1 шт., 150 грн

Загальна вартість витрачених матеріалів становить 580 грн. З урахуванням коефіцієнта транспортування – 638 грн.

Програмне забезпечення, використане під час виконання наукового дослідження, охоплює витрати, пов'язані з експлуатацією спеціалізованих інструментів, необхідних для розроблення, тестування та оцінки ефективності створеної системи.

У ході роботи застосовувалися середовище розробки Visual Studio Code та мережевий аналізатор Wireshark, що використовувався для відстеження обміну даними між компонентами PLC-системи. Реалізація програмного продукту здійснювалася в операційному середовищі Ubuntu Linux, яке забезпечує стабільну роботу з мовою Python та інструментами для аналізу мережевого трафіку. Додаткові витрати на закупівлю програмних засобів відсутні, оскільки використані програми мають відкриті або безкоштовні ліцензії.

Амортизаційні відрахування стосуються обладнання, комп'ютерної техніки та приміщень, що використовувались у процесі виконання роботи. Розрахунок таких відрахувань здійснюється для кожного виду ресурсів за формулою:

$$А_{обл} = \frac{Ц_б}{Т_в} \cdot \frac{t_{вик}}{12} \quad (4.5)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Відповідно до пункту 137.3.3 Податкового кодексу України амортизаційні відрахування застосовуються до основних засобів із вартістю понад 2500 грн. Під час розробки системи виявлення аномальної активності у роботі PLC використовувався персональний комп'ютер із балансною вартістю 20000 грн. Середній строк служби комп'ютера прийнято 2 роки (24 місяців), при цьому для виконання даного етапу роботи комп'ютер експлуатувався протягом 2 місяців.

$$\text{А}_{\text{обл}} = \frac{20000}{24} = 833$$

Сума амортизаційних відрахувань для обладнання становить 833 грн. Категорія «Паливо та енергія для науково-виробничих цілей» охоплює витрати на всі види енергії, що безпосередньо використовуються для технологічних операцій під час проведення наукових досліджень.

Витрати на електроенергію розраховуються за формулою:

$$B_{\text{ен}} = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot \text{Ц}_\text{е} \cdot K_{\text{впі}}}{\eta_i} \qquad (4.6)$$

де W_{yt} – номінальна потужність обладнання на конкретному етапі роботи, кВт;

$t_{\text{і}}$ – час роботи обладнання під час досліджень, год;

$\text{Ц}_\text{е}$ – ціна 1 кВт·год електроенергії, грн;

$\text{К}_{\text{впі}}$ – коефіцієнт використання потужності (< 1);

$\eta_{\text{і}}$ – ККД обладнання (< 1).

Для реалізації розробки використовувався персональний комп'ютер з потужністю 0,45 кВт, що працював протягом 160 годин. Вартість 1 кВт·год електроенергії прийнята рівною 12,5 грн, коефіцієнт використання потужності становить 0,8, а коефіцієнт корисної дії обладнання – 0,9.

$$B_{\text{ен}} = \frac{0{,}45 \cdot 160 \cdot 12{,}5 \cdot 0{,}8}{0{,}9} = 800{,}0 \text{ грн}$$

Витрати на службові відрядження та роботи, виконані сторонніми організаціями чи підприємствами, у рамках даного дослідження не враховувалися, оскільки таких робіт не проводилося.

Накладні (загальновиробничі) витрати Внзв охоплюють управління розробкою, утримання та експлуатацію основних засобів, оплату комунальних послуг, заходи з охорони праці та інші супутні витрати. У даному дослідженні накладні витрати прийнято рівними 100 % основної заробітної плати розробників, тобто:

$$B_{\text{нзв}} = (З_{\text{о}} + З_{\text{р}}) \cdot \frac{\text{Н}_{\text{нзв}}}{100\%} \qquad (4.7)$$

де $\text{Н}_{\text{нзв}}$ – норма нарахування за статтею «Інші витрати».

$$B_{\text{нзв}} = (18190{,}5) \cdot \frac{100\%}{100\%} = 18190{,}5 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР:

$$B = 18190{,}5 + 2182{,}9 + 4482{,}1 + 638 + 833 + 800{,}0 + 18190{,}5 = 45317{,}0 \text{ грн}$$

Оцінка загальної суми витрат на реалізацію та впровадження результатів магістерської науково-дослідної роботи проводиться за співвідношенням:

$$ЗВ = B \cdot \beta \quad (4.8)$$

де β – коефіцієнт, що відображає етап виконання наукових досліджень.

Для поточного проєкту, який перебуває на стадії НДР, приймаємо β = 0,9. Таким чином, загальні витрати складають:

$$ЗВ = 45317{,}0 \cdot 0{,}9 = 40785{,}3 \text{ грн}$$

4.3 Розрахунок економічної ефективності впровадження

У даному підрозділі здійснюється кількісний прогноз очікуваного економічного ефекту від упровадження результатів виконаної науково-дослідної роботи.

Розроблене програмне забезпечення для виявлення аномалій у PLC-системах призначене для зниження втрат, пов'язаних із простоєм обладнання та некоректною роботою технологічних процесів. Очікується, що впровадження системи дозволить зменшити кількість аварійних зупинок, скоротити час на діагностику несправностей та підвищити стабільність роботи виробничої установки.

Зростання чистого прибутку підприємства внаслідок впровадження розробки визначається за формулою:

$$\Delta П_і = \sum_1^n (\Delta Ц_о \cdot N \cdot Ц_о \cdot \Delta N) \cdot \lambda \cdot \rho \left(1 - \frac{\upsilon}{100}\right) \quad (4.9)$$

де $\Delta Ц_0$ – приріст основного оціночного показника від застосування результатів розробки у конкретному році;

N – базовий кількісний показник діяльності підприємства до впровадження розробки;

ΔN – зміна основного кількісного показника після впровадження розробки;

$Ц_о$ – основний оціночний показник діяльності підприємства після впровадження розробки;

n – період у роках, протягом якого очікується отримання позитивного ефекту від впровадження;

λ – коефіцієнт, сплати податку на додану вартість. Коефіцієнт λ = 0,8333.

ρ – коефіцієнт рентабельності продукту. ρ = 0,25;

υ – ставка податку на прибуток з урахуванням військового збору (2025 рік υ = 23%).

Припустимо, що застосування програмного продукту підвищує ефективність виробничих процесів, внаслідок чого ціна одиниці продукції зростає на 1000 грн, а обсяг реалізації збільшується: у перший рік – на 55 одиниць, у другий – на 50 одиниць, у третій – на 40 одиниць. До впровадження розробки реалізовувалась 1 одиниця продукції за ціною 15 000 грн. На основі цих даних розраховується прибуток підприємства за три роки.

$$\Delta П_1 = [1000 \cdot 1 + (15000 + 1000) \cdot 55] \cdot 0{,}833 \cdot 0.25 \cdot \left(1 - \frac{23}{100}\right) = 141321 \text{ грн}$$

$$\Delta П_2 = [1000 \cdot 1 + (15000 + 1000) \cdot (55 + 50)] \cdot 0{,}833 \cdot 0.25 \cdot \left(1 - \frac{23}{100}\right) = 269650 \text{ грн}$$

$$\Delta П_3 = [1000 \cdot 1 + (15000 + 1000) \cdot (55 + 50 + 40)] \cdot 0{,}833 \cdot 0.25 \cdot \left(1 - \frac{23}{100}\right) = 372311 \text{ грн}$$

4.4 Оцінка окупності інвестицій

Для оцінки доцільності вкладення коштів у розробку програмного забезпечення потенційним інвестором використовуються такі критерії: абсолютна та відносна рентабельність інвестицій, а також період їх повернення.

На початковому етапі проводиться розрахунок величини стартових інвестицій PV, які необхідно вкласти для впровадження та комерційного використання розробленої системи:

$$PV = ЗB \cdot k \tag{4.10}$$

де ЗВ=45317,0 грн — витрати на виконання науково-дослідної роботи, наведені у розділі 4;;

k — коефіцієнт, що враховує додаткові витрати інвестора на впровадження та комерціалізацію системи (підготовка приміщень, навчання персоналу, інтеграція з існуючими PLC-системами, маркетингові заходи). Приймається k=3.

Тоді початкові інвестиції становитимуть:

$$PV = 40785{,}3 \cdot 3 \approx 122\,355{,}9 \text{ грн}$$

Абсолютну ефективність вкладених інвестицій $Е_{абс}$ обчислюємо за формулою:

$$Е_{абс} = ПП - PV \tag{4.11}$$

де ПП — приведена вартість усіх чистих прибутків, які підприємство отримає в результаті впровадження системи виявлення аномалій у PLC, грн; PV= 122 355,9грн — початкові інвестиції, розраховані раніше.

Приведена вартість чистих прибутків визначається таким чином:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t} \quad (4.12)$$

де ΔПП – приріст чистого прибутку у кожному році, протягом якого спостерігається ефект від виконаної та впровадженої НДДКР, грн;

Т – період, протягом якого проявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, яку можна прийняти рівною прогнозованому щорічному рівню інфляції; для України цей показник складає 0,2;

t – номер року в розрахунковому періоді.

$$ПП = \frac{141321}{(1+0{,}2)^1} + \frac{269650}{(1+0{,}2)^2} + \frac{372311}{(1+0{,}2)^3} = 520483$$

Тепер можна розрахувати абсолютну ефективність інвестицій:

$$Е_{абс} = 520483 - 122\,355{,}9 = 398\,127 \text{ грн}$$

Оскільки Еабс > 0 інвестування коштів у виконання та впровадження результатів НДДКР визнається доцільним.

Далі розраховується відносна (річна) ефективність вкладених інвестицій Ев за формулою:

$$E_B = \sqrt[Т_Ж]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (4.13)$$

Де $Т_Ж$ – життєвий цикл наукової розробки, роки.

$$E_B = \sqrt[3]{1 + \frac{398\ 127}{122\ 355{,}9}} - 1 = 0{,}62 = 62\%$$

Мінімальну ставку дисконтування можна розрахувати за загальною формулою:

$$\tau = d + f \tag{4.14}$$

де d – середньозважений відсоток за депозитними операціями у комерційних банках, який для України у 2025 році становить 0,14–0,2.

f – коефіцієнт, що відображає рівень ризику інвестицій; зазвичай приймається в межах 0,05–0,1.

$$\tau_{min} = 0.18 + 0.07 = 0.25$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Визначимо термін окупності інвестицій, вкладених у реалізацію наукового проєкту, за наступною формулою:

$$T_{ок} = \frac{1}{E_B} \tag{4.15}$$

$$T_{ок} = \frac{1}{0.62} = 1{,}61$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

4.5 Висновки до розділу

У цьому розділі проведено економічне обґрунтування доцільності розробки програмного забезпечення для виявлення аномалій у роботі PLC. Загальна вартість виконання НДР становить 40785,3 грн, а з урахуванням коефіцієнта впровадження k=3 початкові інвестиції дорівнюють 122 355,9 грн.

Приведена вартість чистих прибутків за три роки становить 520483 грн, що забезпечує абсолютну ефективність інвестицій на рівні 398 127 грн. Відносна ефективність перевищує мінімальну ставку дисконтування, а термін окупності - близько півтора року, що є менше нормативного значення.

Отже, впровадження розробленої системи є економічно доцільним, має високий рівень ефективності та може бути успішно комерціалізоване в умовах промислових підприємств.

ВИСНОВКИ

У рамках проведеної магістерської дослідницької роботи виконано комплексне дослідження методів виявлення аномальної активності у PLC-керованих виробничих системах. Метою роботи було розробити удосконалений підхід до моніторингу технологічних процесів із використанням поведінкового аналізу та машинного навчання, що враховує циклічну природу виробничих циклів та специфіку роботи контролерів. Було проведено аналіз існуючих методів захисту PLC-систем, розроблено алгоритм виявлення аномалій, реалізовано програмну систему для тестування запропонованого підходу та оцінено економічну доцільність її впровадження.

В першому розділі роботи розглянуто архітектуру PLC, принципи їх функціонування та основні методи захисту інформації у промислових системах. Проаналізовано структуру контролерів, що поєднують апаратні та програмні компоненти для циклічного виконання керуючих програм та взаємодії з сенсорами й виконавчими механізмами. Досліджено існуючі методи виявлення аномальної активності, серед яких класичні підходи на основі моніторингу часу виконання завдань, аналізу графа викликів та використання теневого стеку, а також алгоритми машинного навчання, такі як однокласові методи (OCSVM, OCNN) та ізоляційний ліс. Розглянуто програмні засоби для захисту PLC-систем, які реалізують сигнатурний і поведінковий аналіз та алгоритми машинного навчання для виявлення відхилень у роботі контролерів. Проведено порівняння основних засобів захисту, що дозволило оцінити їх можливості для різних умов виробничих процесів і визначити обмеження традиційних рішень. На основі проведеного аналізу сформульовано завдання роботи щодо розробки удосконаленого методу виявлення аномальної активності у PLC-керованих системах із використанням поведінкового аналізу та машинного навчання.

У другому розділі обґрунтовано удосконалення методу виявлення аномальної активності у PLC-керованому обладнанні з урахуванням циклічної природи промислових процесів. Використання стандарту ISA-88 для формалізації

фазової структури технологічного процесу дозволяє створювати моделі нормальної поведінки з прив'язкою до конкретних етапів виробничого циклу. Розроблено ER-модель, яка відображає структуру взаємодії між компонентами системи виявлення аномалій, що включає сім сутностей від PLC_CIP до Anomaly, і описує два рівні аналізу даних: поведінковий аналіз на основі правил та аналіз засобами нейронних мереж. Сформовано алгоритм роботи методу, який реалізує циклічний моніторинг технологічних параметрів процесу CIP із автоматичною зупинкою обладнання при виявленні відхилень на будь-якому рівні перевірки. Підпрограма поведінкового аналізу виконує перевірку фазових переходів, діапазонів параметрів, швидкості зміни та кореляцій між параметрами, а запис нормальної роботи системи у лог-файли забезпечує можливість ретроспективного аналізу та коригування параметрів моделей. Розроблена структура бази даних на основі тегів системи PCS7 включає 17 параметрів процесу CIP, розподілених по чотирьох фазах циклу мийки.

У третьому розділі представлено програмну реалізацію удосконаленого методу. Розроблено LSTM-автоенкодери для кожної фази CIP-процесу, що дозволяє врахувати динамічні зміни параметрів у межах фаз і перехідні стани між ними. Навчання виконано на відокремлених наборах нормальних даних, для кожної моделі визначено порогові значення реконструкційної помилки за 99-м перцентилем. Інтеграція LSTM-детектора з поведінковим аналізатором забезпечила можливість фіксувати як часові, так і логічні відхилення, що розширило спектр виявлюваних порушень. У системі реалізовано збір даних, збереження результатів аналізу, потокову обробку та запис аномалій у структурованому форматі.

Окремо виконано два етапи тестування розробленого методу. Перший етап передбачав порівняння отриманих показників моделі з результатами, наведеними у сучасних роботах з виявлення аномалій у часових рядах. Співставлення значень Precision, Recall, F1 та AUC дало змогу оцінити відповідність запропонованого підходу загальноприйнятим підходам у літературі та підтвердило узгодженість результатів моделі із типовими характеристиками таких методів.

Другий етап був спрямований на перевірку роботи системи в цілому у локальному експериментальному середовищі програмного продукту з графічним інтерфейсом. Тестування виконано для різних рівнів ін'єкцій аномалій. У режимі без аномалій зареєстровано поодинокі спрацьовування LSTM-моделі з низьким відхиленням реконструкції, що відповідає очікуваній поведінці моделі при потоковому аналізі даних. У сценарії з 50 % ін'єкцій аномалій зафіксовано 56,6 % виявлених відхилень. Це пояснюється тим, що відхилення у часовому ряді впливає на кілька сусідніх кроків послідовності, збільшуючи кількість випадків перевищення порогу. Поведінковий аналізатор та LSTM-детектор спрацьовували на різних типах аномалій, що демонструє доцільність комбінування обох підходів у межах однієї системи.

У четвертому розділі проведено економічне обґрунтування розробки програмного забезпечення для виявлення аномалій у роботі PLC. Загальна вартість виконання НДР становить 40785,3 грн, а з урахуванням коефіцієнта впровадження k=3 початкові інвестиції дорівнюють 122355,9 грн. Розраховано чисті прибутки за три роки при ставці дисконтування 0,2, що становлять 520483 грн, що забезпечує абсолютну економічну ефективність інвестицій на рівні 398127 грн. Відносна ефективність перевищує мінімальну ставку дисконтування, а термін окупності становить близько півтора року. Розробка може бути впроваджена на промислових підприємствах для моніторингу циклічних процесів, скорочення простоїв і підтримки контролю параметрів виробництва.

Таким чином, у роботі проведено аналіз теоретичних основ захисту PLC-систем, удосконалено метод виявлення аномальної активності із застосуванням фазової структури процесу і машинного навчання, реалізовано програмне забезпечення та проведено експериментальну перевірку. Встановлено економічну доцільність впровадження розробленого методу на промислових підприємствах, що дозволяє систематично здійснювати контроль технологічних параметрів і виявляти відхилення в роботі PLC.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The History of Programmable Logic Controllers: From Then to Now. C3 Controls. URL: https://www.c3controls.com/white-paper/history-of-programmable-logic-controllers (дата звернення: 23.09.2025).

2. Koondhar M. A. et al. The Role of PLC in Automation, Industry and Education Purpose: A Review. Pakistan Journal of Engineering Technology & Science. 2023. Vol. 11, № 2. P. 22–31. URL: https://www.researchgate.net/publication/379937523 (дата звернення: 18.09.2025).

3. Sartika L. et al. Development of a User Interface for Traffic Light Control Using PLC at a Four-Way Intersection. Jurnal Teknik Elektro. 2024. P. 97–99. URL: https://journals2.ums.ac.id/emitor/article/view/7129/3239 (дата звернення: 27.09.2025).

4. Mohamed A. Programmable logic controllers in flexible manufacturing system (FMS). AIP Conference Proceedings. 2023. Vol. 2643. URL: https://www.researchgate.net/publication/375242892 (дата звернення: 16.09.2025).

5. Zyubin V. E. et al. poST: A Process-Oriented Extension of the IEC 61131-3 Structured Text Language. IEEE. 2021. URL: https://ieeexplore.ieee.org/document/9729833 (дата звернення: 29.09.2025).

6. Colombo, A. W., Karnouskos, S., Kaynak, O. Industrial Cyberphysical Systems: A Backbone of Smart Manufacturing. IEEE Industrial Electronics Magazine, 2021. URL: https://doi.org/10.1109/MIE.2021.3070546

7. Al Ogaili R. R. N., Manickam S., Alsaeedi A. H. Distributed reflection denial of service attack: A critical review. International Journal of Electrical and Computer Engineering. 2021. Vol. 11, № 6. URL: https://www.researchgate.net/publication/351978952 (дата звернення: 22.09.2025).

8. Han S., Lee K., Cho S., Park M. Anomaly Detection Based on Temporal Behavior Monitoring in Programmable Logic Controllers. Electronics. 2021. Vol. 10, No. 10. Art. 1218. URL: https://doi.org/10.3390/electronics10101218.

9. Knowles, W., Prince, D., Hutchison, D., Disso, J. P., Jones, K. A survey of cyber security management in industrial control systems. International Journal of Critical Infrastructure Protection, 2020. URL: https://doi.org/10.1016/j.ijcip.2020.100343

10. Boateng E. A., Bruce J. W. Unsupervised Ensemble Methods for Anomaly Detection in PLC-based Process Control. 2023. URL: https://arxiv.org/pdf/2302.02097 (дата звернення: 30.09.2025).

11. Snort – Network Intrusion Detection & Prevention System. URL: https://www.snort.org/ (дата звернення: 19.09.2025).

12. Suricata Open Source IDS / IPS / NSM engine. URL: https://suricata.io/ (дата звернення: 25.09.2025).

13. Nozomi Networks – OT and IoT Security. URL: https://www.nozominetworks.com/ (дата звернення: 17.09.2025).

14. Conpot – ICS/SCADA Honeypot. URL: https://conpot.org/ (дата звернення: 01.10.2025).

15. Nespoli, P., Papamartzivanos, D., Gómez Mármol, F., Kambourakis, G. Cyber-Physical Intrusion Detection in Industry 4.0: An Overview. IEEE Access, 2020. URL: https://doi.org/10.1109/ACCESS.2020.2968520

16. Ruiz, A., Márquez, F. Anomaly Detection in Industrial Control Systems Using Hybrid Behavioral and Neural Approaches. Sensors, 2022. URL: https://doi.org/10.3390/s22093121

17. Garcia A. et al. Methodology and Tools to Integrate Industry 4.0 CPS into Process Design and Management: ISA-88 Use Case. Information. 2022. Vol. 13, № 5. URL: https://www.mdpi.com/2078-2489/13/5/226 (дата звернення: 26.09.2025).

18. PCS 7 Unit Template "CIP – Cleaning in Place" SIMATIC PCS 7 V8.2 / Siemens AG. 2016. 142 с. URL: https://cache.industry.siemens.com/dl/files/886/78463886/att_892726/v2/78463886_DOCU_CIP_PCS7_V82_en.pdf (дата звернення: 20.09.2025).

19. Wu, F., Liu, J., Wang, P. Cycle-Based Modeling and Monitoring of Industrial Processes. Control Engineering Practice, 2020. URL: https://doi.org/10.1016/j.conengprac.2020.104465

20. Boateng E. A., Bruce J. W. Unsupervised Machine Learning Techniques for Detecting PLC Process Control Anomalies. Journal of Cybersecurity and Privacy. 2022. Vol. 2, № 2. P. 12. URL: https://www.mdpi.com/2624-800X/2/2/12 (дата звернення: 15.09.2025).

21. Cleaning Sequence // ScienceDirect Topics : [website]. URL: https://www.sciencedirect.com/topics/engineering/cleaning-sequence (date of access: 05.12.2025).

22. Cleaning-In-Place (CIP) Process: Step-by-Step Guide // Agriculture Notes by Agriculture.Institute: [website]. 2024. January 19. URL: https://agriculture.institute/milk-processing-packaging/cip-process-step-by-step-guide/ (date of access: 05.12.2025).

23. CIP Cycle : Please post // ProBrewer Discussion Board : [website]. URL: https://discussions.probrewer.com/forum/probrewer-message-board/brewery-operations/cleaning-sanitizing-q-a/43552-cip-cycle-please-post (date of access: 05.12.2025).

24. A Guide to CIP / A&B Process Systems. JBT FoodTech, 2021. 8 p. URL: https://www.jbtc.com/foodtech/wp-content/uploads/sites/2/2021/09/A-Guide-to-CIP.pdf (date of access: 05.12.2025).

25. EHEDG Guideline Doc. 8: Hygienic Design Principles [Електронний ресурс] / European Hygienic Engineering & Design Group. – 3rd ed., March 2018. – Режим доступу: https://www.ehedg.org/guidelines/ (дата звернення: 05.12.2025).

26. 3-A Sanitary Standards and Accepted Practices [Електронний ресурс] / 3-A Sanitary Standards Inc. – 2024. – Режим доступу: https://www.3-a.org/ (дата звернення: 05.12.2025).

27. NSF/ANSI 4-2024: Commercial Cooking Equipment Sanitation [Електронний ресурс] / American National Standards Institute. – June 4, 2025. – Режим доступу: https://blog.ansi.org/ansi/nsf-ansi-4-2024-commercial-cooking-equipment/ (дата звернення: 05.12.2025).

28. Tractian. CIP Cleaning: How to Ensure Food Safety Standards [Електронний ресурс] / Tractian. – 2024. – Режим доступу: https://tractian.com/en/blog/cip-cleaning-how-to-ensure-food-safety-standards (дата звернення: 05.12.2025).

29. Central States Industrial Equipment. What Is CIP in the Food Industry? 5-Step Clean-in-Place Process [Електронний ресурс] / CSI Designs. – 2025. – Режим доступу: https://www.csidesigns.com/blog/articles/5-steps-in-a-common-food-dairy-beverage-clean-in-place-cycle (дата звернення: 05.12.2025).

30. Alsco. Clean in Place (CIP) Meaning in the Food Industry [Електронний ресурс] / Alsco. – 2024. – Режим доступу: https://alsco.com/resources/clean-in-place-cip-meaning-in-the-food-industry/ (дата звернення: 05.12.2025).

31. Food Safety Drains. Understanding CIP and COP in Food Safety [Електронний ресурс] / Food Safety Drains. – December 12, 2023. – Режим доступу: https://blog.foodsafedrains.com/cip-cop-food-safety (дата звернення: 05.12.2025).

32. Almalawi A. et al. ICS-Flow Dataset: Anomaly Detection Dataset for Industrial Control Systems // arXiv:2305.09678. 2023.

33. Kim J. et al. Generating ICS Anomaly Data Reflecting Cyber-Attack Based on Systematic Sampling and Linear Regression // PMC. 2024.

34. Кожанов А. Є. Дослідження моделей формального опису знань про функціональні вимоги до інформаційної системи з використанням EER-діаграми. Моделі та методи модернізації інформаційних систем : зб. наук. праць. Харків : ХНУРЕ, 2023. С. 58. URL: https://openarchive.nure.ua/server/api/core/bitstreams/f9126f5c-e1e0-4ae3-8853-c310b73a2fb6/content (дата звернення: 28.09.2025).

35. Cardoso, J., Sousa, P., Barata, J. Data Modeling for Industrial Cyber-Physical Systems. Procedia Manufacturing, 2021. URL: https://doi.org/10.1016/j.promfg.2021.02.040

36. Schäfer, L., Büscher, C. Rule-Based and Data-Driven Monitoring for Industrial Automation Systems. IFAC PapersOnLine, 2020. URL: https://doi.org/10.1016/j.ifacol.2020.12.2350

37. Drescher, J., Hübner, F. Cleaning-in-Place Systems in the Food Industry: Operational Parameters and Optimization. Food Engineering Reviews, 2021. URL: https://doi.org/10.1007/s12393-021-09317-8

38. Malhotra, P., Vig, L., Shroff, G. Long Short Term Memory Networks for Anomaly Detection in Time Series. ESANN Conference, 2020. URL: https://arxiv.org/abs/1507.00148

39. Python. URL: https://www.python.org/about/ (дата звернення: 07.10.2025).

40. Visual Studio Code. URL: https://learn.microsoft.com/en-us/shows/visual-studio-code/ (дата звернення: 07.10.2025).

41. Python OPC UA. URL: https://python-opcua.readthedocs.io/en/latest/ (дата звернення: 07.10.2025).

42. PyTorch. URL: https://pytorch.org/ (дата звернення: 07.10.2025).

43. NumPy. URL: https://numpy.org/ (дата звернення: 07.10.2025).

44. Bader, S., Barnstedt, E. OPC UA: Technology Overview and Best Practices for Secure Industrial Connectivity. OPC Foundation Whitepaper, 2023. URL: https://opcfoundation.org/developer-tools/specifications-unified-architecture

45. Van Houdt G., Mosquera C., Nápoles G. A Review on the Long Short-Term Memory Model. Artificial Intelligence Review. 2020. Vol. 53, № 1. URL: https://www.researchgate.net/publication/340493274_A_Review_on_the_Long_Short-Term_Memory_Model (дата звернення: 07.10.2025).

46. LSTM-Autoencoder Deep Learning Model for Anomaly Detection in Electric Motor / J. Peixoto, J. Sousa, R. Carvalho [et al.] // Energies. – 2024. – Vol. 17, No. 10. – Art. 2340. DOI: 10.3390/en17102340

47. Dehlaghi-Ghadim A. et al. Anomaly Detection Dataset for Industrial Control Systems. 2023. URL: https://arxiv.org/pdf/2305.09678 (дата звернення: 07.10.2025).

48. Rewicki F., Denzler J., Niebling J. Is it worth it? Comparing six deep and classical methods for unsupervised anomaly detection in time series. 2023. URL: https://arxiv.org/abs/2212.11080 (дата звернення: 07.10.2025).

49. Jeon, H., Park, J. Simulated Industrial Environments for Evaluating Anomaly Detection Algorithms. IEEE Access, 2021. URL: https://doi.org/10.1109/ACCESS.2021.3085631

50. Münz, G., Carle, G. Behavior-Based Anomaly Detection for Industrial Control Systems. Robotics and Computer-Integrated Manufacturing, 2020. URL: https://doi.org/10.1016/j.rcim.2020.101918

51. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ
Голова секції «Управління інформаційною безпекою» кафедри МБІС
д.т.н., професор
Юрій ЯРЕМЧУК
«24» вересня 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:

«Виявлення аномальної активності у PLC-керованому обладнанні на основі адаптації методів поведінкового аналізу і машинного навчання з урахуванням специфіки циклічних виробничих процесів»

08-72.МКР.003.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи
к.т.ф.м.н., доцент Шиян А. А.

Вінниця – 2025

1. Найменування та область застосування

Програмний засіб для виявлення аномальної активності у PLC-керованих виробничих системах. Область застосування: контроль та забезпечення інформаційної безпеки промислових процесів на основі PLC-контролерів, виявлення відхилень у роботі обладнання та захист технологічних процесів від потенційних порушень.

2. Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №___ від ..2025 р. та вимог магістерської кваліфікаційної роботи зі спеціальності 125 – Кібербезпека.

3. Мета та призначення розробки

3.1. Мета розробки: удосконалення методів виявлення аномальної активності у PLC-керованих виробничих системах із застосуванням поведінкового аналізу та машинного навчання з урахуванням циклічності технологічних процесів.

3.2. Призначення: програмний засіб забезпечує моніторинг та виявлення відхилень у роботі PLC, автоматичну реакцію на аномалії та інтеграцію поведінкового та нейронного аналізу для підвищення надійності системи.

4. Джерела розробки

4.1. Van Houdt G., Mosquera C., Nápoles G. A Review on the Long Short-Term Memory Model // Artificial Intelligence Review. – 2020. – Vol. 53(1). – DOI:10.1007/s10462-020-09838-1

4.2. Dehlaghi-Ghadim A., Helali Moghadam M., Balador A., Hansson H. Anomaly Detection Dataset for Industrial Control Systems [Електронний ресурс] // arXiv. – 2023. – Режим доступу: https://arxiv.org/pdf/2305.09678 (дата звернення 07.10.2025)

4.3. Rewicki F., Denzler J., Niebling J. Is it worth it? Comparing six deep and classical methods for unsupervised anomaly detection in time series [Електронний ресурс] // arXiv. – 2023. – Режим доступу: https://arxiv.org/abs/2212.11080 (дата звернення 07.10.2025)

5. Вимоги до програми

5.1. Вимоги до функціональних характеристик:

5.1.1. Програмний засіб повинен мати зрозумілий інтерфейс користувача;

5.1.2. Реалізація методу не повинна вимагати спеціальних ліцензійних програмних продуктів;

5.2. Вимоги до надійності:

5.2.1. Програмний засіб повинен працювати без помилок; у разі виникнення відхилень має виводити повідомлення;

5.2.2. Програмний засіб повинен виконувати всі заявлені функції.

5.3. Вимоги до технічних засобів:

– процесор – Pentium 1500 МГц або аналогічний;

– оперативна пам’ять – не менше 512 Мб;

– середовище функціонування – ОС Windows;

– техніка безпеки при роботі з програмою повинна відповідати стандартам.

6. Вимоги до програмної документації

6.1. Має бути розроблена поетапна інструкція для користувачів із прикладами запуску та налаштування системи.

7. Вимоги до технічного захисту інформації

7.1. Забезпечення захисту програмного засобу від несанкціонованого використання;

7.2. Неможливість доступу незареєстрованих користувачів до даних системи.

8. Техніко-економічні показники

8.1. Економічна цінність системи повинна перевищувати витрати на її розробку;

8.2. Система повинна бути придатною для використання у широкому колі промислових підприємств.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Визначення напрямку магістерської роботи, формулювання теми	24.09.2025	26.09.2025
2	Аналіз предметної області обраної теми	27.09.2025	30.09.2025
3	Апробація отриманих результатів	01.10.2025	08.10.2025
4	Розробка алгоритму роботи	09.10.2025	15.10.2025
5	Написання магістерської роботи на основі розробленої теми	16.10.2025	28.10.2025
6	Розробка економічної частини	29.10.2025	10.11.2025
7	Передзахист магістерської кваліфікаційної роботи	21.11.2025	25.11.2025
8	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	26.11.2025	05.12.2025
9	Захист магістерської кваліфікаційної роботи	09.12.2025	13.12.2025

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

- ПЗ до магістерської кваліфікаційної роботи;
- програмний додаток;
- презентація;
- відзив керівника роботи;
- відзив опонента

Технічне завдання до виконання прийняв _____ Дарморoз Д.О

Додаток Б Лістинг програми. Генератор даних

```
class CIPPhase(Enum):
    PRERINSE = "Prerinse"
    DETERGENT = "Detergent"
    MIXER = "Mixer"
    POSTRINSE = "Postrinse"

class NormalDataGenerator:
    def __init__(self, tolerance: float = 0.1, cycle_duration: float = 10.0):
        self.tolerance = tolerance
        self.cycle_duration = cycle_duration
        self.current_phase = CIPPhase.PRERINSE
        self.cycle_number = 0

        self.logger = setup_logger(
            name="NormalDataGenerator",
            level="DEBUG",
            log_file="logs/normal_data_generator.log",
            console_output=True
        )

        self.phase_durations = {
            CIPPhase.PRERINSE: 10.0,
            CIPPhase.DETERGENT: 20.0,
            CIPPhase.MIXER: 12.5,
            CIPPhase.POSTRINSE: 7.5
        }

        self.state = {
            'temperature': 20.0,
            'level': 0.0,
            'concentration': 0.0,
            'flow': 0.0
        }

        self.phase_start_time = None
        self.cycle_start_time = None

        self.logger.info(f"Initialized with tolerance ±{tolerance}%, cycle duration {cycle_duration}s")

    def _apply_tolerance(self, value: float) -> float:
        variation = value * (self.tolerance / 100.0)
        return value + random.uniform(-variation, variation)

    def _advance_phase(self):
        phases = list(CIPPhase)
        current_index = phases.index(self.current_phase)

        if current_index < len(phases) - 1:
            self.current_phase = phases[current_index + 1]
            self.phase_start_time = time.time()
            self.logger.info(f"Phase transition to {self.current_phase.value}")
        else:
```

```
            self.current_phase = phases[0]
            self.phase_start_time = time.time()
            self.cycle_number += 1
            self.logger.info(f"New cycle #{self.cycle_number} started, phase: {self.current_phase.value}")

    def _generate_prerinse_data(self) -> Dict[str, Any]:
        phase_duration = self.phase_durations[CIPPhase.PRERINSE]
        elapsed = time.time() - self.phase_start_time
        progress = min(elapsed / phase_duration, 1.0)

        filling = progress < 0.4
        rinsing = 0.4 <= progress < 0.9
        draining = progress >= 0.9

        if filling:
            level = (progress / 0.4) * 100
        elif rinsing:
            level = 100.0
        else:
            level = 100 - ((progress - 0.9) / 0.1) * 100

        return {
            'LSH_PreRinse': 1 if level >= 95 else 0,
            'LSL_PreRinse': 1 if level <= 5 else 0,
            'Pump_PreRinse': 1 if rinsing or draining else 0,
            'LEVEL': self._apply_tolerance(max(0, min(100, level))),
            'TEMP': self._apply_tolerance(20.0),
            'FILL_HEAT_CONC': 0.0,
            'QIT': self._apply_tolerance(150.0 if filling or rinsing or draining else 0.0)
        }

    def _generate_detergent_data(self) -> Dict[str, Any]:
        phase_duration = self.phase_durations[CIPPhase.DETERGENT]
        elapsed = time.time() - self.phase_start_time
        progress = min(elapsed / phase_duration, 1.0)

        filling = progress < 0.2
        heating = 0.2 <= progress < 0.6
        circulating = 0.6 <= progress < 0.9
        draining = progress >= 0.9

        target_temp = 75.0
        if heating or circulating:
            self.state['temperature'] = min(target_temp, self.state['temperature'] + 2.5)
        elif draining:
            self.state['temperature'] = max(20.0, self.state['temperature'] - 5.0)
        else:
            self.state['temperature'] = 20.0

        if filling:
            level = (progress / 0.2) * 100
            concentration = 2.5
        elif heating or circulating:
```

```
            level = 100.0
            concentration = 2.5
        elif draining:
            level = 100 - ((progress - 0.9) / 0.1) * 100
            concentration = 2.5
        else:
            level = 0.0
            concentration = 0.0

        return {
            'TIC_DetHeat': self._apply_tolerance(self.state['temperature']),
            'LIT_Detergent': self._apply_tolerance(max(0, min(100, level))),
            'YS_DetFreshWater': 1 if filling else 0,
            'YS_RetDetergent': 1 if circulating else 0,
            'LEVEL': self._apply_tolerance(max(0, min(100, level))),
            'TEMP': self._apply_tolerance(self.state['temperature']),
            'FILL_HEAT_CONC': self._apply_tolerance(concentration),
            'QIT': self._apply_tolerance(200.0 if circulating else 100.0 if filling or draining else 0.0)
        }

    def _generate_mixer_data(self) -> Dict[str, Any]:
        phase_duration = self.phase_durations[CIPPhase.MIXER]
        elapsed = time.time() - self.phase_start_time
        progress = min(elapsed / phase_duration, 1.0)

        filling = progress < 0.3
        mixing = 0.3 <= progress < 0.8
        draining = progress >= 0.8

        if filling:
            level = (progress / 0.3) * 100
        elif mixing:
            level = 100.0
        else:
            level = 100 - ((progress - 0.8) / 0.2) * 100

        return {
            'LSH_Mixer': 1 if level >= 95 else 0,
            'LSL_Mixer': 1 if level <= 5 else 0,
            'NS_Mixer': 1 if mixing else 0,
            'LEVEL': self._apply_tolerance(max(0, min(100, level))),
            'TEMP': self._apply_tolerance(self.state['temperature']),
            'FILL_HEAT_CONC': self._apply_tolerance(1.5),
            'QIT': self._apply_tolerance(250.0 if mixing else 120.0 if filling or draining else 0.0)
        }

    def _generate_postrinse_data(self) -> Dict[str, Any]:
        phase_duration = self.phase_durations[CIPPhase.POSTRINSE]
        elapsed = time.time() - self.phase_start_time
        progress = min(elapsed / phase_duration, 1.0)

        filling = progress < 0.4
        rinsing = 0.4 <= progress < 0.8
```

```
        draining = progress >= 0.8

        if filling:
            level = (progress / 0.4) * 100
        elif rinsing:
            level = 100.0
        else:
            level = 100 - ((progress - 0.8) / 0.2) * 100

        self.state['temperature'] = max(20.0, self.state['temperature'] - 3.0)

        return {
            'LSH_Postrinse': 1 if level >= 95 else 0,
            'LSL_Postrinse': 1 if level <= 5 else 0,
            'YS_FreshWaterPostrinse': 1 if filling else 0,
            'LEVEL': self._apply_tolerance(max(0, min(100, level))),
            'TEMP': self._apply_tolerance(self.state['temperature']),
            'FILL_HEAT_CONC': 0.0,
            'QIT': self._apply_tolerance(180.0 if filling or rinsing else 0.0)
        }

    def generate(self) -> Dict[str, Any]:
        if self.phase_start_time is None:
            self.phase_start_time = time.time()
            self.cycle_start_time = time.time()
            self.logger.info("CIP process started")

        phase_generators = {
            CIPPhase.PRERINSE: self._generate_prerinse_data,
            CIPPhase.DETERGENT: self._generate_detergent_data,
            CIPPhase.MIXER: self._generate_mixer_data,
            CIPPhase.POSTRINSE: self._generate_postrinse_data
        }

        data = phase_generators[self.current_phase]()
        data['phase'] = self.current_phase.value
        data['cycle'] = self.cycle_number
        data['timestamp'] = time.time()

        elapsed = time.time() - self.phase_start_time
        self.logger.debug(f"Generated data for {self.current_phase.value} ({elapsed:.2f}s): {data}")

        if elapsed >= self.phase_durations[self.current_phase]:
            self._advance_phase()

        return data
```

Отримання даних для аналізу

```
class OPCUADataClient:
  def __init__(self, host: str = '127.0.0.1', port: int = 4840):
    self.host = host
    self.port = port
    self.url = f"opc.tcp://{host}:{port}/cip/process"
    self.client = None
    self.nodes = {}

    self.logger = setup_logger(
      name="OPCUAClient",
      level="DEBUG",
      log_file="logs/opcua_client.log",
      console_output=True
    )

    self.logger.info(f"Initialized OPC UA client for {self.url}")

  def connect(self) -> bool:
    """Connect to OPC UA server and get node references"""
    try:
      self.client = Client(self.url)
      self.client.connect()
      self.logger.info(f"Connected to OPC UA server at {self.url}")

      root = self.client.get_root_node()
      objects = self.client.get_objects_node()

      cip_process = objects.get_child(["2:CIPProcess"])

      self.nodes['phase'] = cip_process.get_child(["2:Phase"])
      self.nodes['cycle'] = cip_process.get_child(["2:Cycle"])

      self.nodes['LSH_PreRinse'] = cip_process.get_child(["2:LSH_PreRinse"])
      self.nodes['LSL_PreRinse'] = cip_process.get_child(["2:LSL_PreRinse"])
      self.nodes['Pump_PreRinse'] = cip_process.get_child(["2:Pump_PreRinse"])

      self.nodes['TIC_DetHeat'] = cip_process.get_child(["2:TIC_DetHeat"])
      self.nodes['LIT_Detergent'] = cip_process.get_child(["2:LIT_Detergent"])
      self.nodes['YS_DetFreshWater'] = cip_process.get_child(["2:YS_DetFreshWater"])
      self.nodes['YS_RetDetergent'] = cip_process.get_child(["2:YS_RetDetergent"])

      self.nodes['LSH_Mixer'] = cip_process.get_child(["2:LSH_Mixer"])
      self.nodes['LSL_Mixer'] = cip_process.get_child(["2:LSL_Mixer"])
      self.nodes['NS_Mixer'] = cip_process.get_child(["2:NS_Mixer"])

      self.nodes['LSH_Postrinse'] = cip_process.get_child(["2:LSH_Postrinse"])
      self.nodes['LSL_Postrinse'] = cip_process.get_child(["2:LSL_Postrinse"])
      self.nodes['YS_FreshWaterPostrinse'] = cip_process.get_child(["2:YS_FreshWaterPostrinse"])

      self.nodes['LEVEL'] = cip_process.get_child(["2:LEVEL"])
```

```
            self.nodes['TEMP'] = cip_process.get_child(["2:TEMP"])
            self.nodes['FILL_HEAT_CONC'] = cip_process.get_child(["2:FILL_HEAT_CONC"])
            self.nodes['QIT'] = cip_process.get_child(["2:QIT"])

            self.nodes['timestamp'] = cip_process.get_child(["2:Timestamp"])
            self.nodes['anomaly_injected'] = cip_process.get_child(["2:AnomalyInjected"])

            self.logger.info(f"Successfully retrieved {len(self.nodes)} OPC UA nodes")
            return True

        except Exception as e:
            self.logger.error(f"Connection error: {e}")
            return False

    def disconnect(self):
        """Disconnect from OPC UA server"""
        try:
            if self.client:
                self.client.disconnect()
                self.logger.info("Disconnected from OPC UA server")
        except Exception as e:
            self.logger.error(f"Error disconnecting: {e}")

    def read_data(self) -> Optional[Dict[str, Any]]:
        """Read all process data from OPC UA server"""
        if not self.client or not self.nodes:
            self.logger.warning("Not connected to OPC UA server")
            return None

        try:
            data = {}
            for key, node in self.nodes.items():
                data[key] = node.get_value()

            self.logger.debug(f"Read data: phase={data.get('phase')}, cycle={data.get('cycle')},
TEMP={data.get('TEMP', 0):.1f}°C, LEVEL={data.get('LEVEL', 0):.1f}%")
            return data

        except Exception as e:
            self.logger.error(f"Error reading data: {e}")
            return None
```

Модель Autoencoder LSTM та методи опрацювання

```
class LSTMAutoencoder(nn.Module):
  def __init__(self, input_size: int, hidden_size: int = 64, num_layers: int = 2, dropout: float = 0.2):
    super(LSTMAutoencoder, self).__init__()

    self.input_size = input_size
    self.hidden_size = hidden_size
    self.num_layers = num_layers

    self.encoder = nn.LSTM(
      input_size=input_size,
      hidden_size=hidden_size,
      num_layers=num_layers,
      batch_first=True,
      dropout=dropout if num_layers > 1 else 0
    )

    self.decoder = nn.LSTM(
      input_size=hidden_size,
      hidden_size=hidden_size,
      num_layers=num_layers,
      batch_first=True,
      dropout=dropout if num_layers > 1 else 0
    )

    self.output_layer = nn.Linear(hidden_size, input_size)

  def forward(self, x):
    _, (hidden, cell) = self.encoder(x)
    batch_size, seq_len, _ = x.shape
    decoder_input = hidden[-1].unsqueeze(1).repeat(1, seq_len, 1)
    decoder_output, _ = self.decoder(decoder_input, (hidden, cell))
    reconstructed = self.output_layer(decoder_output)
    return reconstructed

class LSTMDetector:
  def __init__(
    self,
    models_dir: str = "models/lstm",
    threshold_percentile: float = 95.0,
    device: str = None
  ):
    self.models_dir = Path(models_dir)
    self.threshold_percentile = threshold_percentile
    self.device = torch.device('cuda' if torch.cuda.is_available() else 'cpu') if device is None else
torch.device(device)

    self.logger = setup_logger(
      name="LSTMDetector",
      level="INFO",
      log_file="logs/lstm_detector.log",
```

```
            console_output=True
        )

        self.models = {}
        self.thresholds = {}
        self.scalers = {}
        self.feature_names = []
        self.sequence_length = 20

        self.phase_histories = {
            'Prerinse': [],
            'Detergent': [],
            'Mixer': [],
            'Postrinse': []
        }

        self.current_phase = None
        self.grace_period = 10
        self.samples_since_phase_change = 0

    def load_models(self):
        phases = ['Prerinse', 'Detergent', 'Mixer', 'Postrinse']

        for phase in phases:
            model_path = self.models_dir / f"{phase}_model.pth"
            config_path = self.models_dir / f"{phase}_config.json"

            if not model_path.exists():
                self.logger.warning(f"Model not found for {phase}: {model_path}")
                continue

            with open(config_path, 'r') as f:
                config = json.load(f)

            model = LSTMAutoencoder(
                input_size=config['input_size'],
                hidden_size=config['hidden_size'],
                num_layers=config['num_layers'],
                dropout=config['dropout']
            )

            model.load_state_dict(torch.load(model_path, map_location=self.device))
            model.to(self.device)
            model.eval()

            self.models[phase] = model
            self.thresholds[phase] = config['threshold']

            if 'scaler_min' in config and 'scaler_max' in config:
                self.scalers[phase] = {
                    'method': 'minmax',
                    'min': np.array(config['scaler_min'], dtype=np.float32),
                    'max': np.array(config['scaler_max'], dtype=np.float32),
```

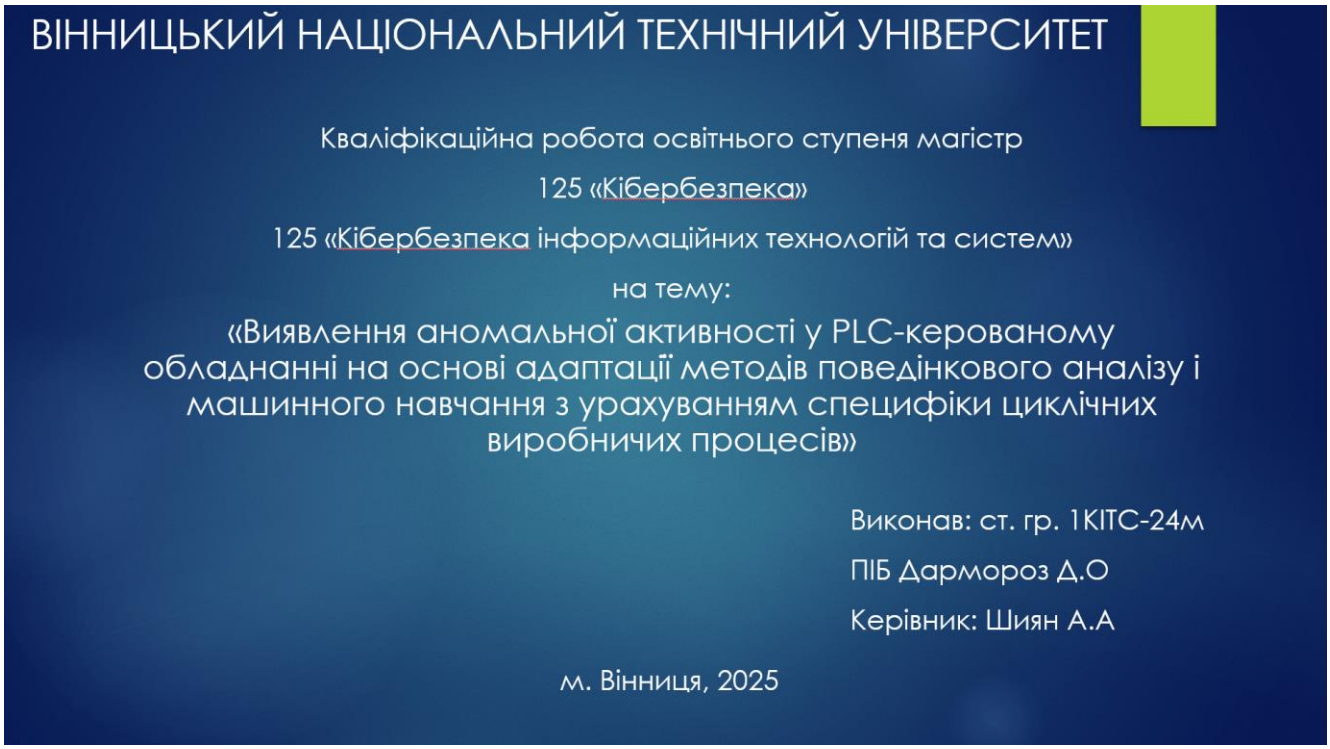
```
                'range': np.array(config['scaler_range'], dtype=np.float32),
                'constant_features': config.get('scaler_constant_features', [])
            }
            self.logger.info(f"Using MinMax normalization for {phase}")
        elif 'scaler_mean' in config and 'scaler_std' in config:
            self.scalers[phase] = {
                'method': 'zscore',
                'mean': np.array(config['scaler_mean'], dtype=np.float32),
                'std': np.array(config['scaler_std'], dtype=np.float32)
            }
            self.logger.warning(f"Using legacy Z-score normalization for {phase} - consider retraining")
        else:
            raise ValueError(f"Invalid scaler config for {phase}")

        self.feature_names = config['feature_names']
        self.sequence_length = config['sequence_length']

        self.logger.info(
            f"Loaded model for {phase}: threshold={config['threshold']:.4f}, "
            f"features={len(self.feature_names)}"
        )

    def _extract_features(self, data: Dict[str, Any]) -> np.ndarray:
        features = []
        for feature_name in self.feature_names:
            value = data.get(feature_name, 0.0)
            if isinstance(value, bool):
                value = float(value)
            features.append(value)
        return np.array(features, dtype=np.float32)

    def _normalize(self, features: np.ndarray, phase: str) -> np.ndarray:
        scaler = self.scalers[phase]

        if np.any(np.isnan(features)):
            self.logger.error(f"NaN detected in input features for {phase}")
            features = np.nan_to_num(features, nan=0.0)

        if np.any(np.isinf(features)):
            self.logger.error(f"Inf detected in input features for {phase}")
            features = np.nan_to_num(features, posinf=1e6, neginf=-1e6)

        if scaler['method'] == 'minmax':
            normalized = np.zeros_like(features, dtype=np.float32)
            constant_features = scaler['constant_features']

            for i in range(len(features)):
                if i in constant_features:
                    normalized[i] = 0.0
                else:
                    normalized[i] = (features[i] - scaler['min'][i]) / (scaler['range'][i] + 1e-10)
                    normalized[i] = np.clip(normalized[i], -0.1, 1.1)
        else:
```

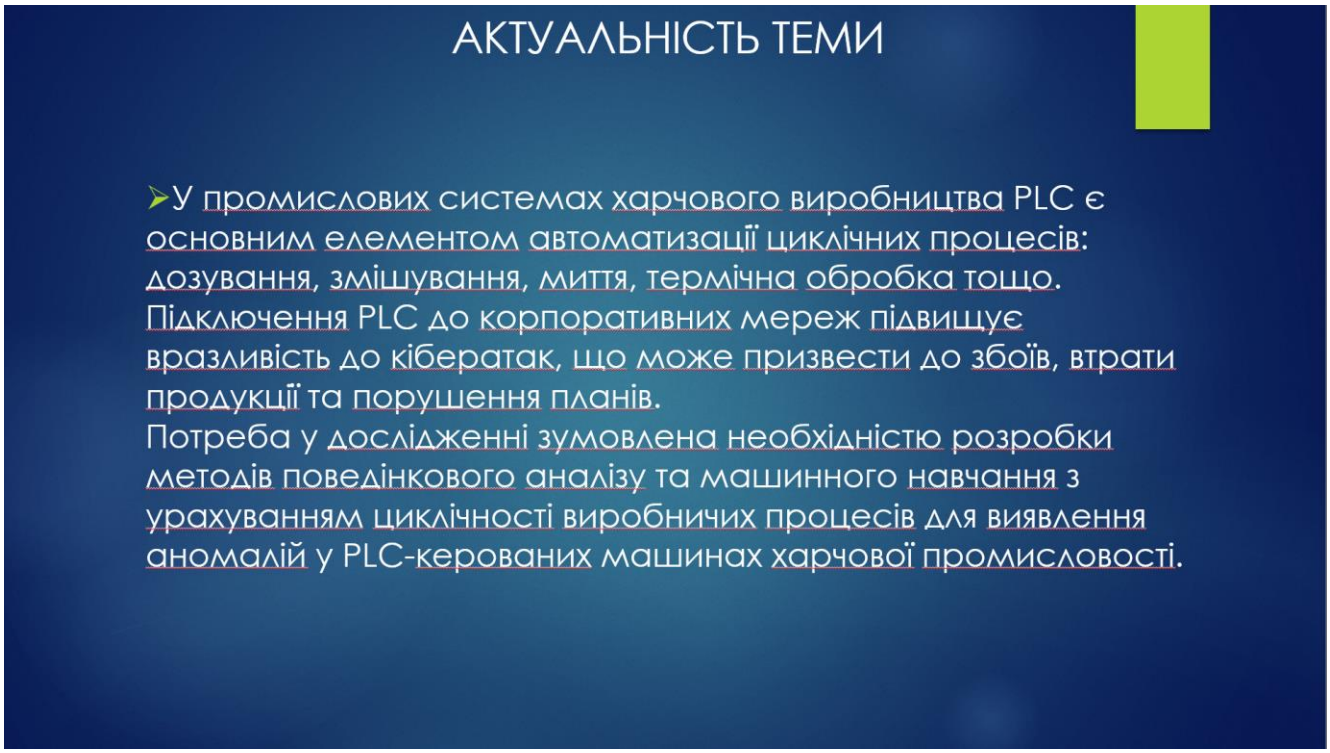
```
            normalized = (features - scaler['mean']) / (scaler['std'] + 1e-8)

        if np.any(np.isnan(normalized)):
            self.logger.error(f"NaN in normalized features for {phase}, setting to 0")
            normalized = np.nan_to_num(normalized, nan=0.0)

        if np.any(np.isinf(normalized)):
            self.logger.error(f"Inf in normalized features for {phase}, clipping")
            normalized = np.nan_to_num(normalized, posinf=10.0, neginf=-10.0)

        return normalized

    def _calculate_reconstruction_error(
        self,
        sequence: np.ndarray,
        model: nn.Module
    ) -> float:
        x = torch.FloatTensor(sequence).unsqueeze(0).to(self.device)

        if torch.isnan(x).any() or torch.isinf(x).any():
            self.logger.error(f"NaN or Inf detected in input sequence! x stats: min={x.min()}, max={x.max()},
mean={x.mean()}")
            return float('inf')

        with torch.no_grad():
            reconstructed = model(x)

        if torch.isnan(reconstructed).any() or torch.isinf(reconstructed).any():
            self.logger.error(f"NaN or Inf in reconstruction! stats: min={reconstructed.min()},
max={reconstructed.max()}")
            return float('inf')

        mse = torch.mean((x - reconstructed) ** 2).item()

        if np.isnan(mse) or np.isinf(mse):
            self.logger.error("MSE is NaN or Inf, returning large value")
            return 1e10

        if mse > 1e10:
            self.logger.warning(f"Very large MSE detected: {mse:.2e}, capping at 1e10")
            mse = 1e10

        return mse

    def analyze(self, data: Dict[str, Any]) -> Dict[str, Any]:
        phase = data.get('phase', 'Unknown')
        timestamp = data.get('timestamp', 0.0)

        if self.current_phase is not None and self.current_phase != phase:
            self.logger.info(f"Phase changed from {self.current_phase} to {phase}, resetting history")
            self.reset_history()
            self.samples_since_phase_change = 0
```

```
self.current_phase = phase
self.samples_since_phase_change += 1

if phase not in self.models:
    return {
        'is_anomaly': False,
        'phase': phase,
        'reconstruction_error': 0.0,
        'threshold': 0.0,
        'confidence': 0.0,
        'timestamp': timestamp,
        'reason': f"No LSTM model for phase '{phase}'"
    }

features = self._extract_features(data)
normalized_features = self._normalize(features, phase)

if self.samples_since_phase_change <= self.grace_period:
    return {
        'is_anomaly': False,
        'phase': phase,
        'reconstruction_error': 0.0,
        'threshold': self.thresholds[phase],
        'confidence': 0.0,
        'timestamp': timestamp,
        'reason': f"Grace period: {self.samples_since_phase_change}/{self.grace_period}"
    }

self.phase_histories[phase].append(normalized_features)

if len(self.phase_histories[phase]) < self.sequence_length:
    return {
        'is_anomaly': False,
        'phase': phase,
        'reconstruction_error': 0.0,
        'threshold': self.thresholds[phase],
        'confidence': 0.0,
        'timestamp': timestamp,
        'reason': f"Accumulating sequence: {len(self.phase_histories[phase])}/{self.sequence_length}"
    }

sequence = np.array(self.phase_histories[phase][-self.sequence_length:])
reconstruction_error = self._calculate_reconstruction_error(sequence, self.models[phase])
threshold = self.thresholds[phase]
is_anomaly = reconstruction_error > threshold
confidence = (reconstruction_error / threshold - 1.0) if is_anomaly else 0.0

result = {
    'is_anomaly': is_anomaly,
    'phase': phase,
    'reconstruction_error': reconstruction_error,
    'threshold': threshold,
    'confidence': confidence,
```

```
            'timestamp': timestamp
        }

        if is_anomaly:
            self.logger.warning(
                f"LSTM ANOMALY DETECTED in {phase}: "
                f"reconstruction_error={reconstruction_error:.6f} > threshold={threshold:.6f} "
                f"(+{confidence*100:.1f}%)"
            )
        else:
            self.logger.debug(
                f"LSTM Normal in {phase}: error={reconstruction_error:.6f}, "
                f"threshold={threshold:.6f}"
            )

        return result

    def reset_history(self, phase: Optional[str] = None):
        if phase:
            self.phase_histories[phase] = []
        else:
            for p in self.phase_histories:
                self.phase_histories[p] = []
            self.current_phase = None
            self.samples_since_phase_change = 0
```

Додаток В. Презентація до дипломної магістерської кваліфікаційної роботи



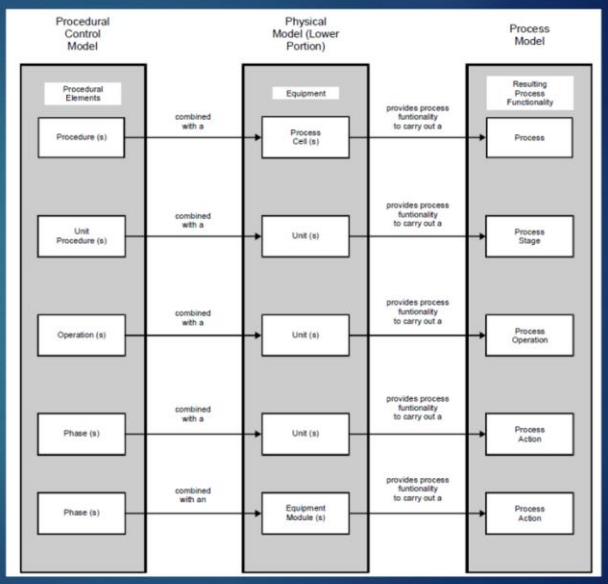


МЕТА РОБОТИ, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи**
  Удосконалення методу виявлення аномальної активності у PLC-керованому обладнанні шляхом інтеграції поведінкового аналізу та машинного навчання з урахуванням циклічності виробничих процесів.
- **Об'єкт дослідження**
  Процеси захисту PLC-керованих виробничих систем від інформаційних загроз, що виникають унаслідок деструктивних впливів на промислові мережі та контрольні програми.
- **Предмет дослідження**
  Методи поведінкового аналізу та алгоритми машинного навчання, адаптовані до циклічних виробничих процесів для виявлення аномальної активності.

СПЕЦИФІКА ЦИКЛІЧНИХ ВИРОБНИЧИХ ПРОЦЕСІВ

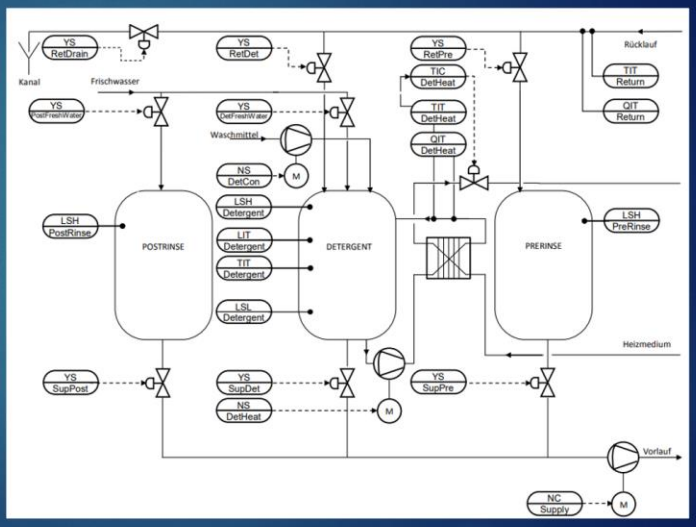
Виробничі процеси, що керуються програмованими логічними контролерами, характеризуються високою регулярністю виконання операцій.
Технологічні параметри змінюються за передбачуваними траєкторіями.
Для формалізації циклічних виробничих процесів використовується стандарт ISA-88.
У нашому дослідженні акцент робиться саме на фазах як найдетальнішому рівні процедури керування.



CIP-ПРОЦЕС

Об'єктом демонстрації методу є CIP-система, технологічний цикл якої включає ополіскування, циркуляцію мийного розчину та дезінфекцію. На PI-діаграмі відображені основні елементи:
- PRERINSE – резервуар для попереднього ополіскування;
- DETERGENT – резервуар мийного розчину;
- POSTRINSE – резервуар для фінального ополіскування;

Дані з цих компонентів використовуються для формування моделі нормальної поведінки та подальшого виявлення аномалій за допомогою поведінкового аналізу й машинного навчання.



СТРУКТУРА ТЕХНОЛОГІЧНИХ ПАРАМЕТРІВ CIP-ПРОЦЕСУ

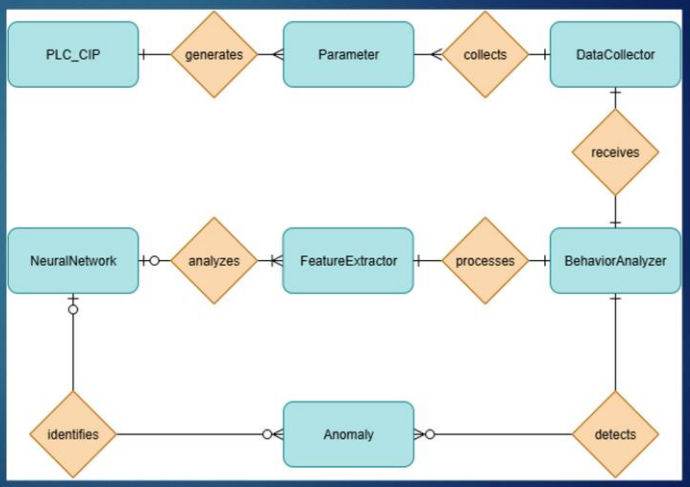
Фаза	Назва параметра	Тип даних	Значення / стан
Prerinse	LSH_PreRinse	BOOL	0/1
Prerinse	LSL_PreRinse	BOOL	0/1
Prerinse	Pump_PreRinse	BOOL	0/1
Detergent	TIC_DetHeat	REAL	0–95 °C
Detergent	LIT_Detergent	REAL	0–100%
Detergent	YS_DetFreshWater	BOOL	0/1
Detergent	YS_RetDetergent	BOOL	0/1
Mixer	LSH_Mixer	BOOL	0/1
Mixer	LSL_Mixer	BOOL	0/1

Фаза	Назва параметра	Тип даних	Значення / стан
Mixer	NS_Mixer	BOOL	0/1
Postrinse	LSH_Postrinse	BOOL	0/1
Postrinse	LSL_Postrinse	BOOL	0/1
Postrinse	YS_FreshWaterPostrinse	BOOL	0/1
Всі фази	LEVEL	REAL	0–100%
Всі фази	TEMP	REAL	0–95 °C
Всі фази	FILL_HEAT_CONC	REAL	0–5%
Всі фази	QIT	REAL	0–300 л/хв
Mixer	NS_Mixer	BOOL	0/1

Для моделювання CIP-процесу використовуються параметри зі станів обладнання та аналогових сигналів, прив'язані до фаз Prerinse, Detergent, Mixer і Postrinse. Вони слугують основою для поведінкового аналізу та виявлення аномалій.

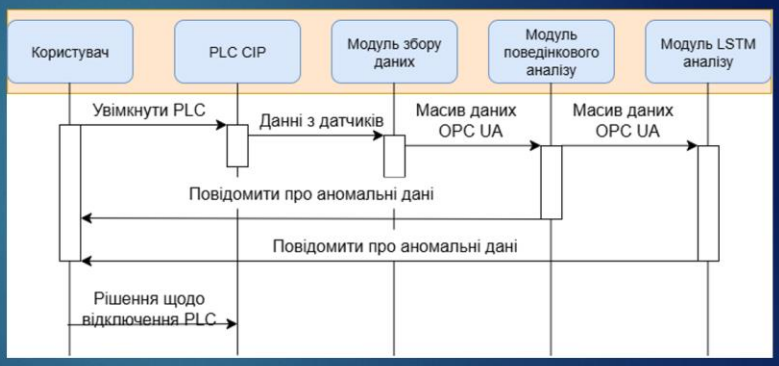
ER-МОДЕЛЬ УДОСКОНАЛЕНОГО МЕТОДУ ВИЯВЛЕННЯ АНОМАЛІЙ

Модель складається з семи основних сутностей:
- **PLC_CIP** – контролер, що генерує параметри;
- **Parameter** – технологічні параметри різних типів;
- **DataCollector** – збір параметрів від контролера;
- **BehaviorAnalyzer** – первинний аналіз за правилами та статистикою;
- **FeatureExtractor** – формування вектора ознак для машинного навчання;
- **NeuralNetwork** – модель, що виявляє приховані аномалії;
- **Anomaly** – результуючі аномалії з прив'язкою до параметрів.

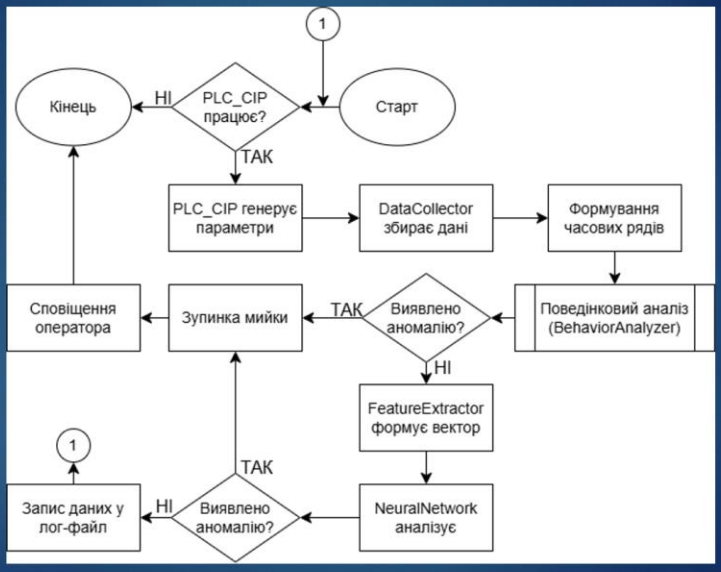


ДІАГРАМА ПОСЛІДОВНОСТІ РОБОТИ МЕТОДУ ВИЯВЛЕННЯ АНОМАЛІЙ

- Користувач активує PLC_CIP, який генерує технологічні параметри.
- DataCollector перевіряє цілісність та передає дані для аналізу.
- BehaviorAnalyzer здійснює первинну оцінку параметрів; при відхиленнях формує повідомлення для користувача.
- LSTM-автоенкодер виконує глибоку перевірку часових рядів для прихованих аномалій.
- Результати передаються користувачу для прийняття рішень (продовження роботи або зупинка обладнання).

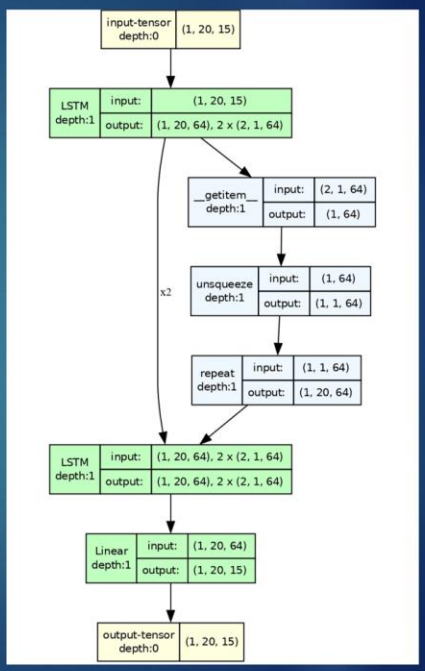


БЛОК-СХЕМА АЛГОРИТМУ ВИЯВЛЕННЯ АНОМАЛІЙ

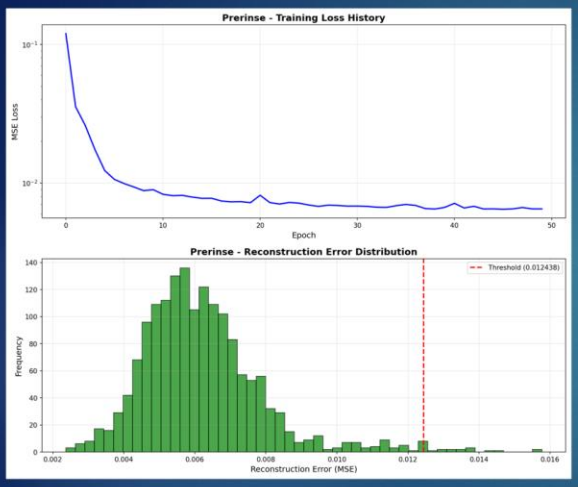


LSTM-АВТОЕНКОДЕР ДЛЯ АНАЛІЗУ АНОМАЛІЙ

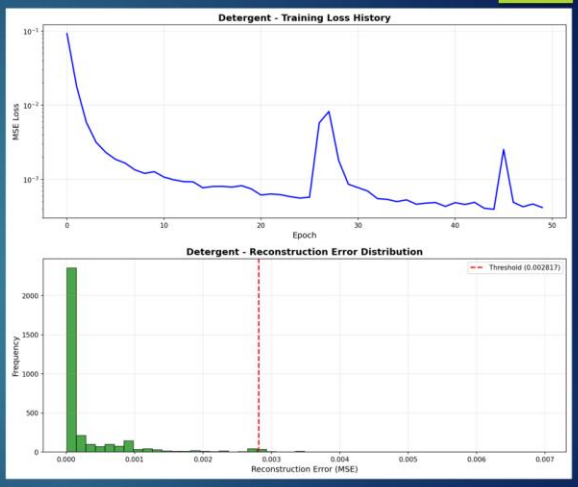
- Для кожної фази PLC створюється окрема модель, щоб врахувати перехідні процеси на початку фаз.
- Автоенкодер LSTM зберігає інформацію про попередні кроки для точного аналізу часових рядів.
- Архітектура: енкодер і декодер, по 2 шари LSTM, прихований шар 64 нейрони, дроп-аут 0.2.
- Вхід: 15 ознак, довжина послідовності – 20 кроків часу.



РЕЗУЛЬТАТИ НАВЧАННЯ LSTM-АВТОЕНКОДЕРІВ (1)

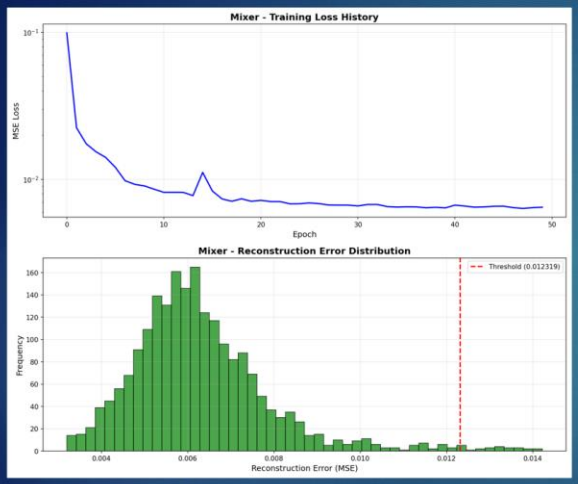


- Prerinse:
Поріг MSE: 0.012438
Фінальна втрата: ~0.006

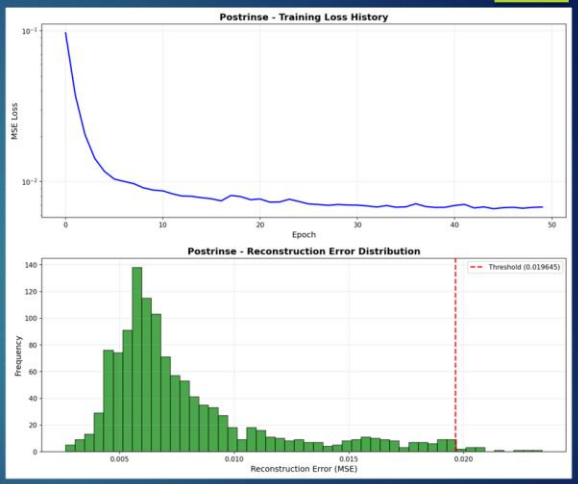


- Detergent:
Поріг MSE: 0.002817
Фінальна втрата: 0.0004153

РЕЗУЛЬТАТИ НАВЧАННЯ LSTM-АВТОЕНКОДЕРІВ (2)



- Mixer:
Поріг MSE: 0.012319
Фінальна втрата: ~0.005

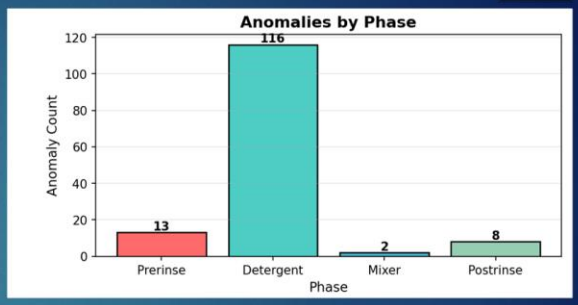


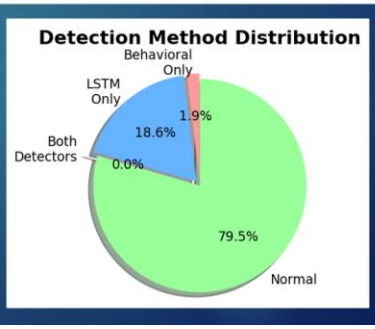
- Postrinse:
Поріг MSE: 0.019645
Фінальна втрата: ~0.006

РЕЗУЛЬТАТИ ВИЯВЛЕННЯ АНОМАЛІЙ

Виявлено аномалій по фазах:
- Prerinse: 13
- Detergent: 116
- Mixer: 2
- Postrinse: 8

Розподіл за типом детектора:
- Нормальні зразки: 79.5%
- Тільки LSTM: 18.6%
- Тільки поведінковий: 1.9%
- Обидва детектори: 0.0%





Висновки

У роботі удосконалено метод виявлення аномальної активності у PLC-керованому обладнанні шляхом інтеграції поведінкового аналізу та LSTM-автоенкодерів з урахуванням фазової структури циклічних процесів згідно зі стандартом ISA-88.

Розроблена дворівнева архітектура забезпечує швидке виявлення очевидних порушень через поведінковий аналізатор та ідентифікацію прихованих аномалій через нейронну мережу. Навчання окремих моделей для кожної фази CIP-процесу показало стабільну збіжність з мінімальною помилкою 0.0004153 для фази Detergent.

Тестування виявило 139 аномалій, при цьому LSTM-автоенкодер ідентифікував 18.6% відхилень, поведінковий аналізатор – 1.9%, а відсутність перетину підтверджує взаємодоповнюваність підходів. Створено прототип програмного забезпечення, що реалізує запропонований метод.

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Виявлення аномальної активності у PLC-керованому обладнанні на основі адаптації методів поведінкового аналізу і машинного навчання з урахуванням специфіки циклічних виробничих процесів

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра менеджменту та безпеки інформаційних систем
факультет менеджменту та інформаційної безпеки
гр.1КІТС-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) **1,50 %**

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту**

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

к.т.н., доцент, зав. каф. МБІС Карпінець В.В. _____

к.ф.-м.н., доцент каф. МБІС Шиян А.А. _____

Особа, відповідальна за перевірку Коваль Н.П. _____

З висновком експертної комісії ознайомлений(-на)

Керівник _____ доц. Шиян А.А.

Здобувач _____ Дарморoз Д.О.