Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Кіберфізична система моніторингу оптичних мереж.

КвРКІ 210483.21.04.33 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент IV курсу, група КІ2-21-4 _____ Максим ГУМЕНЮК
Підпис Ініціали, прізвище

Керівник _____ Олексій ІВАНОВ
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис Ініціали, прізвище

«16» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень Бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Максиму ГУМЕНЮКУ
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система моніторингу оптичних мереж.

Керівник проекту (роботи) Олексій ІВАНОВ, к.т.н., доцент.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 24.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Теоретичні основи досліджуваної проблеми

Проектування та розгортання серверу

Реалізація та тестування серверу та його функціоналу

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Блок-схеми алгоритмів роботи програмного забезпечення

Структура кіберфізичної системи моніторингу оптичних мереж

Результати роботи програмного забезпечення

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «_10_» 01 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – проектування програмно-технічного засобу	01.04.2025	виконано
5	Робота над розділом 3 – програмна реалізація та тестування програмно-технічного засобу	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Максим ГУМЕНЮК
Підпис Ініціали, прізвище

Керівник роботи _____ Олексій ІВАНОВ
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ.210483.21.04.33 ПЗ	Пояснювальна записка	57		
			Графічні матеріали			
2		КвРКІ.210483.21.04.33 Е8	Архітектура ПЗ проєкту	1		
3		КвРКІ.210483.21.04.33 Е8	Архітектура ПЗ для кіберфізичної системи	1		
4		КвРКІ.210483.21.04.33 Е8	Апаратне забезпечення проєкту	1		

КвРКІ.210483.21.04.33 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Гуменюк М		16.06.25
Перевір.		Іванов О		16.06.25
Н. контр.		Кисіль М		16.06.25
Затв.		Павлова О		16.06.25

Відомість проекту

Літера	Аркуш	Аркушів
У	1	64

ХНУ, КІ2-21-4

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Кіберфізична система моніторингу оптичних мереж.».

Автор роботи: Гуменюк Максим Андрійович.

Керівник роботи: Іванов Олексій Валентинович.

Пояснювальна записка:

Графічна частина:

МОНІТОРИНГ, БАЗА ДАНИХ, СЕРВЕР, ВЕБІНТЕРФЕЙС

Метою дипломної роботи є розробка програмного забезпечення для кіберфізичної системи моніторингу оптичної мережі, моделювання її функціонування у віртуальному середовищі, реалізація автоматизованої візуалізації параметрів стану мережевих вузлів та впровадження механізмів нотифікації.

Об'єктом дослідження оптична телекомунікаційна мережа, є яка потребує безперервного контролю працездатності та стабільності функціонування її елементів.

Предметом дослідження є програмна архітектура кіберфізичної системи моніторингу оптичної мережі, включно з її компонентами збору даних, аналізу подій, візуального представлення інформації та засобами оперативного сповіщення.

Підпис студента

30.05.2025

Дата

ЗМІСТ

ВСТУП 3

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ 4

1.1 Аналіз предметної області і виявлення наявних проблем і завдань .. 4

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень 7

1.3 Методологічні підходи до вирішення задачі за темою дослідження 12

1.4 Постановка задачі 13

1.5 Висновки до розділу 14

2 ПРОЕКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ 17

2.1 Обґрунтування вибору мов програмування та програмного забезпечення 17

2.2 Функційні вимоги програмного забезпечення 22

2.3 Проектування роботи та методологічний підхід 26

2.4 Проектування майбутньої системи 28

2.5 Висновки до розділу 36

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ 39

3.1 Принцип роботи програмного забезпечення модуля зчитування компонентів навколишнього середовища 39

3.2 Налаштування та запуск серверної частини моніторингу 43

3.3 Результати роботи програмного забезпечення кіберфізичної системи моніторингу 49

3.4 Інтеграція системи оповіщення через Telegram 53

3.5 Результати візуалізації та роботи веб-інтерфейсу Zabbix 54

ВИСНОВОК 57

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 58

ДОДАТОК А 62

ДОДАТОК Б 63

ДОДАТОК В 64

ВСТУП

У сучасному світі стрімкий розвиток інформаційно-комунікаційних технологій призводить до все ширшого впровадження автоматизованих рішень у різних галузях людської діяльності. Однією з таких перспективних технологій є кіберфізичні системи, які забезпечують тісну інтеграцію між фізичними процесами та обчислювальними ресурсами. Ці системи дозволяють здійснювати моніторинг, управління та аналіз стану різноманітних об'єктів у реальномучасі [3].

Зокрема, в контексті телекомунікацій, актуальним є застосування кіберфізичних систем для моніторингу оптичних мереж, які становлять основу сучасної інфраструктури передачі даних. Збої або пошкодження в таких мережах можуть призвести до значних втрат, як фінансових, так і інформаційних. Тому створення ефективної системи моніторингу є важливим завданням для забезпечення безперервної, надійної та безпечної роботи мереж [2].

Метою даної роботи є дослідження та розробка підходу до побудови інтерфейсу користувача для кіберфізичної системи моніторингу оптичних мереж, що дозволить своєчасно виявляти відхилення у роботі мережевої інфраструктури та оперативно реагувати на надзвичайні ситуації.сучасному світі.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Сучасний стан інформаційних технологій і стрімкий розвиток цифрової інфраструктури зумовили широке впровадження автоматизованих систем моніторингу в різноманітних сферах - від промисловості до побуту [1]. Особливої уваги вимагають телекомунікаційні мережі, зокрема оптичні, які стали основою для передачі даних у багатьох критично важливих системах. Моніторинг таких мереж є ключовим елементом у забезпеченні стабільної та безпечної роботи інформаційної інфраструктури.

Історично, ідея централізованого моніторингу комп'ютерних систем бере свій початок із 1980 - 1990-х років, коли з'явилися перші спроби створення інструментів для адміністраторів мереж. Одним із перших прикладів була система Nagios , яка дозволяла відстежувати доступність хостів та сервісів. З розвитком телекомунікацій з'явилися більш комплексні рішення, як-от Zabbix, Cacti, PRTG, Prometheus, які забезпечують гнучкий збір та візуалізацію даних, налаштування сповіщень і автоматизоване реагування на інциденти [4].

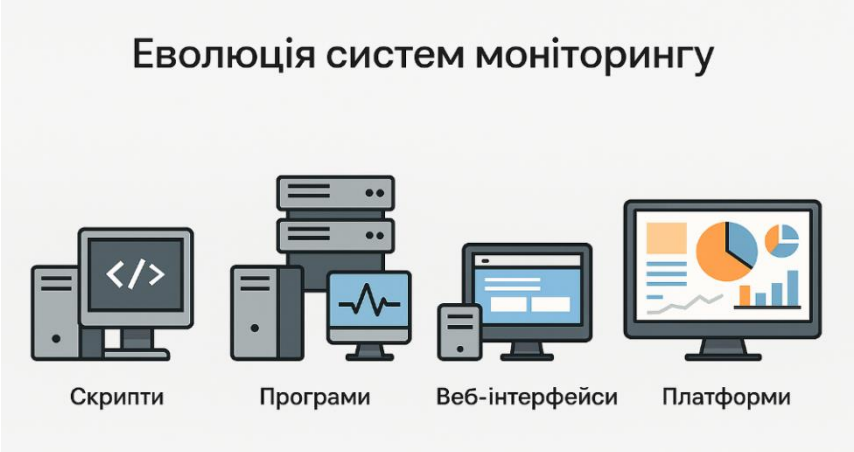


Рисунок 1.1 – Еволюція систем моніторингу

У контексті розвитку індустрії дедалі більшого поширення набуває концепція кіберфізичних систем, які поєднують фізичні об'єкти з віртуальними компонентами через комп’ютерні мережі. У випадку оптичних мереж це дозволяє інтегрувати моніторинг фізичного стану кабелів, активного обладнання, параметрів навантаження та інших метрик у єдину систему, здатну в режимі реального часу відслідковувати, аналізувати та реагувати на зміни [2].

Проблеми, що виникають у сфері моніторингу оптичних мереж:

У процесі аналізу вже існуючих рішень було виявлено низку типових проблем, які суттєво обмежують ефективність класичних систем моніторингу. Однією з ключових проблем є відсутність уніфікованих рішень, адже багато платформ розроблені з фокусом на конкретний тип збору даних - наприклад, лише SNMP або лише агентське збирання. Це створює труднощі при інтеграції з різнорідним обладнанням, що часто зустрічається в умовах реальної інфраструктури, особливо в телекомунікаційній галузі [12].

Іншою проблемою є низький рівень автоматизації реакції на події. Багато рішень обмежуються фіксацією проблем або інцидентів, без подальших дій - навіть елементарного сповіщення адміністратора. У результаті адміністратор може пропустити критичну подію, що безпосередньо впливає на стабільність та безперервність мережевих сервісів [4, 9].

Також суттєвою перешкодою для впровадження на великих об’єктах є складність масштабування. Коли йдеться про десятки чи сотні хостів у різних населених пунктах, системи можуть втрачати ефективність через перевантаження сервера моніторингу, затримки в обробці метрик або надмірну кількість одночасних подій.

Проблеми безпеки також залишаються актуальними. Багато рішень не реалізовують шифрування трафіку між агентом і сервером, що створює потенційні вектори атаки [3]. До того ж, некоректно налаштовані права доступу до веб-інтерфейсу можуть відкрити шлях для несанкціонованого втручання в систему.

Не слід ігнорувати людський фактор - навіть у сучасних системах

адміністратор не завжди має можливість постійно відстежувати події вручну. Без автоматичних тригерів, які формують події і запускають оповіщення, зростає ризик несвоєчасної реакції на збій.

У відповідь на ці проблеми система моніторингу, що розробляється, повинна відповідати певному набору ключових функцій і технічних вимог. Насамперед, важливо забезпечити збір метрик у реальному часі. Серед основних параметрів, які повинні контролюватися - завантаження центрального процесора, використання оперативної пам'яті, простір на диску, статуси мережевих інтерфейсів, а також стан SNMP-сумісного обладнання[6]. Крім того, система повинна передбачати централізоване керування - через єдиний інтерфейс адміністратор має бачити всі вузли, шаблони, події, сповіщення, що спрощує аналіз ситуації та прийняття рішень. Без цього неможливо побудувати масштабовану систему контролю за великою кількістю об'єктів.

Ще однією важливою складовою є гнучка система тригерів і сповіщень. Реакція на перевищення порогових значень має бути не лише зафіксована, а й супроводжуватися повідомленням відповідальній особі - через Telegram, email, або навіть SMS. Це дозволяє миттєво реагувати на зміни у стані системи.

Гнучкість у налаштуваннях передбачає можливість інтеграції сторонніх скриптів, виклику вебхуків. Це відкриває простір для адаптації системи під конкретні потреби замовника або умов експлуатації [5].

Зважаючи на активний розвиток інфраструктури 5G, та хмарних сервісів, роль систем моніторингу лише зростатиме. У майбутньому очікується перехід до використання predictive analytics (прогнозної аналітики), яка дозволить не лише реагувати на проблеми, а й передбачати їх появу. Також важливим напрямом стане підвищення кібербезпеки систем моніторингу через впровадження сучасних протоколів шифрування та контроль доступу на основі ролей.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

Сучасні інформаційні та телекомунікаційні інфраструктури вимагають безперервного спостереження, оцінки стану та своєчасного реагування на аномалії. У відповідь на ці виклики сформувався цілий клас технологій - системи моніторингу, що можуть функціонувати як окремі програмні продукти або як складові елементи кіберфізичних систем. У даному розділі розглянемо порівняння провідних рішень у сфері моніторингу мережевої інфраструктури, серверів і комп'ютерних систем, а також проаналізуємо їхні переваги й недоліки з погляду придатності до інтеграції в кіберфізичних систем.

Рішення з відкритим кодом Zabbix - це система моніторингу корпоративного рівня з відкритим кодом, яка дозволяє збирати метрики з серверів, мережевого обладнання, додатків та віртуалізованих середовищ. Її основними перевагами є [7, 8]: Zabbix є надзвичайно гнучкою та масштабованою платформою, що дозволяє створювати власні шаблони, тригери, елементи даних та адаптувати систему до конкретних потреб. Завдяки підтримці як активного, так і пасивного моніторингу, Zabbix ефективно працює як у невеликих мережах, так і в розподілених інфраструктурах із сотнями хостів. Платформа також підтримує інтеграцію з популярними сервісами сповіщення, такими як Telegram, Slack, Email, що забезпечує оперативне інформування про критичні події. Вбудовані засоби візуалізації - графіки, інформаційні панелі, карти - роблять інтерфейс зручним для аналізу й керування.

Попри свої переваги, Zabbix має низку недоліків. Зокрема, складність первинного налаштування може бути проблемою для недосвідчених користувачів, особливо в частині конфігурації бази даних, агентів і шаблонів. Також система потребує виділеного серверного середовища або розгортання у віртуальній машині, що ускладнює розгортання в умовах обмежених ресурсів. При великій кількості хостів Zabbix може створювати значне навантаження на апаратну частину, особливо на базу даних, що вимагає додаткової оптимізації.

Zabbix добре вписується в архітектуру кіберфізичних систем завдяки можливості працювати з агентами та SNMP, надаючи актуальні дані про стан фізичних або віртуальних компонентів системи.

Також є Prometheus у поєднанні з Grafana є поширеним вибором для побудови систем моніторингу, особливо у хмарних середовищах або при роботі з контейнеризованими сервісами. Prometheus зберігає дані у форматі time-series, що дозволяє ефективно працювати з історичними метриками та масштабувати рішення без значного навантаження [10]. Його природна інтеграція з такими технологіями, як Kubernetes, Docker та Linux-служби, робить його зручним для DevOps-команд і сучасних інфраструктур. Використовуючи Grafana, можна будувати гнучкі панелі візуалізації з великою кількістю налаштувань і оновленням даних у реальному часі.

Водночас, у Prometheus відсутня вбудована система оповіщень - для цього потрібен окремий компонент Alertmanager, що додає складності конфігурації. Слабка підтримка SNMP-протоколу, а також обмежені можливості роботи з Windows-системами ускладнюють використання у змішаних середовищах. Крім того, Prometheus не має централізованої авторизації та детального керування доступами, що може бути критично для великих команд або компаній із жорсткими вимогами до безпеки.

Prometheus більше підходить для хмарних і DevOps-середовищ, але може використовуватись і в рамках кіберфізичних систем, де важлива висока частота збору метрик. Також існують комерційні рішення для велеких компаній. Ось пару прикладів:

Наступним рішенням є PRTG Network Monitor – це потужне комерційне рішення для моніторингу мережевої інфраструктури, серверів, IoT-пристроїв та навіть хмарних сервісів. Його ключовою перевагою є простий та інтуїтивно зрозумілий інтерфейс, що робить його зручним навіть для користувачів без глибоких технічних знань [18]. PRTG автоматично виявляє пристрої у мережі, створює відповідні сенсори, а також пропонує готові шаблони для різних типів обладнання. Крім цього, система має гнучкий механізм нотифікацій, звітів та

можливість візуалізації даних за допомогою інтерактивних дашбордів.

Втім, як і будь-яке рішення корпоративного рівня, PRTG має свої обмеження. По-перше, безкоштовна версія дозволяє створити лише до 100 сенсорів, що може бути замало для складних інфраструктур. По-друге, це закрите програмне забезпечення, що зменшує можливості для кастомізації і не дозволяє модифікувати логіку моніторингу під специфічні задачі. І нарешті, система працює виключно в середовищі Windows, що ускладнює її інтеграцію в гібридні або Linux-орієнтовані мережі [11].

PRTG можна адаптувати під кіберфізичні системи, особливо коли мова йде про гібридні чи промислові системи моніторингу, але закритість платформи ускладнює масштабування. Гарним рішенням для великих підприємств також буде SolarWinds з акцентом на мережевий моніторинг.

Серед переваг PRTG Network Monitor варто відзначити його широку підтримку протоколів збору даних, включаючи SNMP, NetFlow, WMI та власні API-інтерфейси. Це дозволяє гнучко під'єднувати практично будь-яке обладнання незалежно від виробника. Однією з особливостей системи є здатність до глибокого аналізу продуктивності мережі, що включає в себе деталізовану статистику трафіку, навантаження на інтерфейси, час відгуку сервісів тощо [18]. Також варто згадати можливість автоматичної генерації карт залежностей між вузлами, що значно спрощує розуміння топології мережі та локалізацію збоїв.

Проте, не зважаючи на свою потужність, PRTG має ряд недоліків. Найбільшим з них є висока вартість повнофункціональної ліцензії, яка робить систему недосяжною для невеликих організацій або освітніх проєктів. Також впровадження вимагає ретельного планування та певного досвіду, що може створювати труднощі для IT-команд без відповідної підготовки. Крім того, через орієнтацію на великий бізнес, у системи спостерігаються обмеження щодо підтримки локалізованих рішень або специфічних потреб малих підприємств.

Таблиця 1.1 – Порівняння різних систем

Система	Тип	Платформа	Переваги	Недоліки
Zabbix	OpenSource	Linux/ Windows	Потужний функціонал, SNMP, нотифікації	Складне налаштування, важкий UI
Prometheus + Grafana	OpenSource	Linux	Гнучкість, масштабованість	Відсутність нативного SNMP
PRTG	Комерційне	Windows	Зручність, автовиєвлення	Обмеження в безкоштовній версії
SolarWinds	Комерційне	Windows	Потужні можливості, інтеграції	Дуже дорога ліцензія

У контексті кіберфізичних систем, особливо тих, що мають справу з телекомунікаційною інфраструктурою або критичними IT-мережами, важливо враховувати не лише технічні можливості моніторингового рішення, а й його здатність до адаптації, масштабування, зручність підтримки та гнучкість розгортання у різноманітних середовищах [12]. Традиційно більшість моніторингових систем розроблялись із прицілом на центри обробки даних або серверні кімнати, однак сучасні тренди, включно з розширенням IoT та Edge-технологій, вимагають принципово нових підходів до побудови таких систем.

Zabbix, попри свою відносну складність для початківців, виділяється як одна з найбільш гнучких платформ, придатних до побудови розподілених та ієрархічних систем моніторингу. Завдяки підтримці проксі-серверів, ця система дозволяє організовувати багаторівневий моніторинг, що особливо корисно для кіберфізичних систем, які охоплюють декілька майданчиків або географічно розподілені об'єкти [13]. Важливою перевагою Zabbix є його вбудовані механізми тригерів і логіки реагування, що дозволяє автоматизувати процеси виявлення інцидентів і навіть запускати сценарії реагування без участі оператора.

Prometheus, хоч і спочатку був орієнтований на хмарні DevOps-середовища, знайшов своє застосування і в контексті кіберфізичних систем завдяки можливості

роботи з time-series даними. У поєднанні з Grafana ця система дозволяє будувати високоінформативні та кастомізовані панелі, що ідеально підходить для візуалізації технічного стану об'єктів в реальному часі. Проте в разі використання Prometheus для моніторингу гетерогенних інфраструктур, особливо тих, що містять обладнання з SNMP або нестандартними інтерфейсами, можуть виникати певні труднощі з сумісністю та об'єднанням даних із різних джерел.

Варто також звернути увагу на те, що більшість open source-рішень вимагають глибокого технічного розуміння для налаштування та підтримки. Це створює бар'єр для підприємств, де не передбачено виділених IT-фахівців [15,17]. Тут на сцену виходять комерційні рішення, зокрема PRTG Network Monitor. Він орієнтований на простоту та зручність, має зручний GUI, підтримує автообнаруження пристроїв, а також дозволяє швидко налаштувати моніторинг з мінімальними зусиллями. У рамках невеликої кіберфізичних систем, де важливо швидко розгорнути базовий функціонал і отримати візуальні інструменти спостереження, PRTG виглядає дуже привабливо. Проте обмеження безкоштовної версії та закритий код зменшують гнучкість і унеможливлюють повну кастомізації.

Зі зростанням вимог до адаптивності та інтелектуальності кіберфізичних систем виникає потреба у рішеннях, які не просто збирають метрики, а й дозволяють здійснювати аналітику, прогнозування та самовідновлення систем. У цьому контексті найбільш перспективними виглядають рішення, що підтримують машинне навчання, адаптивні тригери, кореляцію подій - хоча наразі вони здебільшого зустрічаються у платних рішеннях або як експериментальні надбудови над існуючими open source системами [14].

Іще один важливий аспект – інтеграція з месенджерами та засобами комунікації. У сучасних кіберфізичних систем важливо не лише вчасно виявити проблему, а й ефективно повідомити про неї відповідальних осіб. Інтеграція з Telegram, Discord, Slack або Microsoft Teams стає стандартом де-факто. У цьому плані Zabbix та Grafana (через Alertmanager) забезпечують широкі можливості налаштування сповіщень, у тому числі за сценаріями: для кого, коли і як надсилати

інформацію. Це не лише полегшує реагування, а й дає змогу будувати ефективну комунікацію всередині кіберфізичних систем. Таким чином, підсумовуючи, можна виокремити загальні тенденції в розвитку систем моніторингу для кіберфізичних систем:

У сучасних системах моніторингу спостерігається перехід від централізованих рішень до розподілених архітектур із використанням агентів та проксі-серверів. Це забезпечує кращу масштабованість, зниження навантаження на центральний сервер і підвищену надійність у великих мережах [19], [20].

Зростає увага до якості інтерфейсів – сучасні системи мають зручні дашборди, гнучку візуалізацію та адаптивний дизайн. Також поширюється інтеграція з месенджерами (Telegram, Slack), що дозволяє отримувати сповіщення в реальному часі.

Нові рішення підтримують прогнозну аналітику та автоматичне виконання сценаріїв реагування. Окрім IT-інфраструктури, дедалі частіше системи охоплюють OT-сегмент (сенсори, контролери), забезпечуючи єдиний моніторинг на всіх рівнях.

Для побудови кіберфізичних систем нового покоління важливо обирати рішення, які не лише відповідають поточним вимогам, а й мають потенціал до адаптації, розширення та підтримки нових протоколів і стандартів. Це особливо актуально в умовах швидкої еволюції інфраструктур, де моніторинг стає не лише інструментом контролю, а основою для прийняття рішень і підвищення надійності всієї системи.

1.3 Методологічні підходи до вирішення задачі за темою дослідження

У вирішенні задачі моніторингу стану оптичних мереж критичним є вибір методологічної бази, що дозволяє ефективно зібрати, обробити та інтерпретувати інформацію про стан інфраструктури. Залежно від вимог до точності, швидкодії, масштабованості та зручності керування, розглядаються кілька підходів до

побудови таких систем. Один з найпоширеніших підходів передбачає використання спеціалізованих платформ для моніторингу, таких як Zabbix, Nagios, Prometheus або PRTG. Кожна з них має свої особливості. Наприклад, Nagios є легким у розгортанні та чудово підходить для базового моніторингу, але вимагає багато ручної конфігурації. Prometheus, у свою чергу, краще працює зі збором метрик у хмарних інфраструктурах і має тісну інтеграцію з Grafana для візуалізації. Zabbix – це комплексне рішення з багатими можливостями, включно з централізованим керуванням, сповіщеннями та детальним журналюванням подій.

Для оповіщення про критичні події все частіше використовуються канали зв'язку в режимі реального часу, зокрема месенджери. Telegram, Slack або Microsoft Teams часто інтегруються в системи моніторингу за допомогою ботів. Такий підхід дозволяє оперативно інформувати відповідальних осіб про несправності або зміни в роботі мережі.

З метою візуалізації інформації про стан системи можуть бути задіяні додаткові платформи, як-от Grafana або Kibana [25]. Вони дають змогу створювати інформативні дашборди, будувати графіки змін параметрів у часі та відстежувати історичні тенденції. Наприклад, інтеграція Zabbix з Grafana дозволяє створювати більш гнучкі панелі керування зі зручним фільтруванням і агрегацією даних [4, 25]

На рівні архітектури системи можлива реалізація як централізованого, так і розподіленого підходу. Централізований варіант передбачає збирання всіх даних на одному сервері, тоді як розподілена модель дозволяє делегувати окремі функції на кілька вузлів, що покращує надійність та масштабованість [27].

1.4 Постановка задачі

Задачею даного дослідження є створення програмного рішення для моніторингу параметрів оптичної мережі з використанням засобів автоматичного збору, обробки, візуалізації та сповіщення про критичні події.

Першим етапом є формування об'єкта дослідження та уточнення мети розробки, що передбачає теоретичне вивчення сучасного стану

телекомунікаційних мереж, зокрема волоконно-оптичних технологій, та принципів їх моніторингу.

Другим етапом є дослідження функціональних можливостей існуючих систем моніторингу інформаційної інфраструктури, зокрема таких рішень як Zabbix, Nagios, Prometheus, а також аналіз способів інтеграції сповіщень через Telegram-ботів та інших каналів [30].

Третім етапом є проведення огляду програмних і апаратних рішень, що можуть бути використані для побудови стенду моніторингу оптичної мережі. Це включає вибір відповідних мов програмування, середовищ розробки, інструментів візуалізації (наприклад, Grafana), та технологій віртуалізації для реалізації проєкту на тестовому середовищі.

Четвертим етапом є уточнення предметної області - топології мережі, що моделюється, параметрів для збору (таких як стан вузлів, пропускна здатність, затримки, навантаження), та визначення логічної структури системи моніторингу.

П’ятим етапом є формування функціональних вимог до системи моніторингу. Серед ключових вимог: здатність до масштабування, підтримка агентського та безагентського моніторингу, конфігурація сповіщень, інтерфейс для візуалізації та можливість роботи у тестовому середовищі без реальної мережі.

Завершальним етапом є розробка, налаштування та тестування стенду програмного забезпечення, яке дозволяє реалізувати повноцінну систему моніторингу параметрів мережі, а також демонстрація її працездатності в умовах симуляції. результати аналізу результати аналізу [6, 18].

1.5 Висновки до розділу

Беручи до уваги результати аналізу предметної області, існуючі рішення у сфері моніторингу телекомунікаційної інфраструктури, а також виявлені потреби щодо контролю та візуалізації параметрів оптичної мережі, сформульовано технічне завдання на розробку системи моніторингу, орієнтованої на виявлення

аномалій, навантаження та несправностей у вузлах мережі.

Запропоноване рішення передбачає побудову програмного комплексу, що забезпечуватиме збирання, обробку, зберігання та візуалізацію інформації про стан компонентів мережі. Воно повинне відповідати визначеним функціональним вимогам, бути масштабованим, стабільним та інформативним для оператора.

Програмна реалізація розроблюваної системи повинна бути структурована за чіткими етапами, з алгоритмічним описом кожної фази, з урахуванням особливостей вибраного середовища розробки, інструментів моніторингу та платформи симуляції. У якості програмних рішень можуть використовуватися платформи з відкритим кодом, зокрема Zabbix, що надає гнучкі можливості з точки зору інтеграції, автоматизації сповіщень та візуального контролю [6, 9, 14].

Такий підхід дозволяє не лише підтримувати високу структурованість розробки, але й гарантує гнучкість, масштабованість та можливість подальшої інтеграції нових компонентів. В основі архітектури розроблюваного рішення лежить поєднання серверної інфраструктури на базі Ubuntu Server із системою моніторингу Zabbix, що є одним із найпотужніших інструментів відкритого програмного забезпечення у сфері IT-спостереження.

Процес розробки системи передбачає послідовне виконання чотирьох ключових етапів, які забезпечують якість і функціональність кінцевого продукту. Першим етапом є програмна реалізація, що базується на обраній архітектурній моделі та технологічних рішеннях. На цьому етапі створюються основні компоненти системи, налаштовуються взаємодії між ними, а також забезпечується інтеграція необхідних модулів і сервісів.

Другий етап включає повний цикл тестування, під час якого система перевіряється на коректність роботи в різних умовах та у відповідь на типові події. Тестування охоплює функціональні можливості, стабільність роботи, реакцію на помилки та ефективність оповіщень. Результати тестування є основою для подальшого вдосконалення системи [3].

Третій етап спрямований на оптимізацію та доопрацювання реалізації з

урахуванням отриманих даних тестування. Вносяться зміни, що покращують продуктивність, підвищують зручність адміністрування і забезпечують більш надійну роботу системи в реальних умовах експлуатації. Одночасно розробляється логічна структура і докладний опис алгоритмів взаємодії компонентів, що дозволяє впорядкувати архітектуру та спрощує подальшу підтримку проекту.

У результаті розробки, система моніторингу має забезпечувати повноцінне охоплення всієї заданої топології оптичної мережі, гарантуючи безперервне спостереження за її ключовими параметрами. Вона повинна відповідати актуальним вимогам до автоматизованого управління інформаційними інфраструктурами, включаючи масштабованість, надійність, адаптивність до змін у мережевій структурі, а також зручність у користуванні для операторів та адміністраторів [18]. Крім цього, важливою є можливість інтеграції з іншими сервісами або модулями, що дозволяє розширювати функціональність системи у майбутньому без потреби в повній реконструкції вже існуючого середовища.

Розробка програмного забезпечення для кіберфізичної системи моніторингу вимагає цілісного підходу, що передбачає кілька ключових етапів. Перш за все, необхідно створити ефективне середовище для розгортання системи, враховуючи віртуалізацію, ізоляцію сервісів і відповідність обчислювальним ресурсам. Далі реалізуються функції збору, обробки й візуалізації телеметричних даних, що дозволяє оперативно виявляти аномалії чи збої. Особливу увагу слід приділити механізмам автоматичного оповіщення про інциденти, які повинні бути гнучкими, надійними та орієнтованими на швидке реагування. Завершальним етапом є тестування та оцінка працездатності системи в умовах, максимально наближених до реальних, що гарантує її відповідність сучасним стандартам у сфері телекомунікаційного моніторингу [2, 5, 8].

2 ПРОЕКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

2.1 Обґрунтування вибору мов програмування та програмного забезпечення

У сучасних телекомунікаційних інфраструктурах, що характеризуються значною складністю і високими вимогами до якості обслуговування (QoS), моніторинг мереж є невід'ємною частиною забезпечення стабільної роботи. Особливо це стосується оптичних мереж, які, з одного боку, демонструють високу пропускну здатність та мінімальні затримки, але з іншого - є вразливими до пошкоджень та деградації сигналу. Зниження рівня оптичного сигналу, збої в роботі активного обладнання або перевантаження певних ділянок мережі можуть призвести до серйозних наслідків: втрати зв'язку, перебоїв у наданні послуг, зменшення пропускної здатності [21].

Щоб запобігти таким інцидентам, оператори зв'язку та інженери використовують системи моніторингу, які дозволяють виявляти аномалії у роботі інфраструктури, швидко реагувати на інциденти та здійснювати превентивне обслуговування. В умовах дипломного проєкту розглядається приклад розгортання кіберфізичної системи моніторингу оптичної мережі, побудованої на основі платформи Zabbix – одного з найпопулярніших рішень з відкритим кодом у цій галузі.

Zabbix – це комплексна система для агентського та безагентського моніторингу, яка дозволяє отримувати дані з обладнання різних виробників, використовуючи стандартизовані протоколи а також власні агенти Zabbix Agent. Серед ключових можливостей Zabbix можна виділити[25, 26].:

Серед основних можливостей Zabbix варто виділити гнучкий механізм збору даних. Система підтримує стандартні протоколи, а також використовує власні агенти для отримання розширеної інформації з хостів. Це дозволяє контролювати такі параметри, як завантаження процесора, використання оперативної пам'яті, доступність дискового простору та інші системні метрики.

Окремою перевагою є використання шаблонів, що значно спрощує конфігурацію. Шаблони включають типові метрики, тригери та графіки, і можуть бути застосовані до кількох хостів одночасно. Це дозволяє швидко масштабувати моніторинг при розширенні мережі.

Zabbix також має потужну систему оповіщення: за допомогою тригерів система може виявляти відхилення від нормальних параметрів і надсилати повідомлення адміністраторам через Telegram, електронну пошту або інші інтегровані сервіси [27].

Через веб-інтерфейс адміністратор має змогу керувати всіма аспектами системи: переглядати метрики, створювати графіки, редагувати конфігурації та створювати індивідуальні дашборди. Інтерфейс працює через Apache або Nginx і доступний за локальною IP-адресою.

Таким чином, Zabbix поєднує широкі технічні можливості з високою гнучкістю, що робить його оптимальним рішенням для побудови системи моніторингу як у навчальному середовищі, так і в реальних інфраструктурних проектах [30, 31].

Однією з ключових переваг Zabbix є масштабованість. Система здатна обробляти десятки тисяч хостів з мільйонами параметрів, що дозволяє використовувати її як у невеликих лабораторіях, так і на рівні національних операторів зв’язку [32].

Рисунок 2.1 – Графічний інтерфейс середовища Zabbix

Розробка сучасних систем моніторингу, зокрема в контексті телекомунікаційних середовищ, потребує комплексного підходу, який об'єднує програмні та інфраструктурні компоненти. В умовах реалізації симуляційної кіберфізичної системи моніторингу оптичної мережі особливу увагу було приділено підбору оптимальних інструментів, які забезпечують масштабованість, надійність, зручність у налаштуванні, а також можливість моделювання віртуального середовища без потреби у фізичному обладнанні [1, 12]. Одним із головних компонентів системи виступає платформа

Оскільки йдеться про симуляцію, критично важливою є можливість налаштувати систему сповіщень про події, які виникають у процесі моніторингу. Для цього було обрано Telegram-бота, як засіб доставки повідомлень [34]. Telegram має низку переваг:

Однією з ключових переваг Telegram є його офіційне API, яке підтримує як просту передачу текстових повідомлень, так і надсилання структурованих повідомлень із використанням розмітки Markdown або HTML. Це дозволяє форматувати сповіщення, наприклад, виділяти типи подій, пріоритети або параметри метрик, що порушили норму.

Telegram працює на широкому спектрі платформ – мобільних (Android, iOS), десктопних (Windows, Linux, macOS) та у веб-версії. Це забезпечує зручність доступу для адміністратора системи незалежно від того, де він перебуває.

Додатковою перевагою є можливість створювати приватних ботів із автентифікацією через токен, який генерується за допомогою сервісу @BotFather. Такий бот може бути прив’язаний до одного або кількох користувачів, а також доданий до групи або каналу для розсилки сповіщень кільком адміністраторам одночасно. Telegram також підтримує двофакторну автентифікацію, шифрування повідомлень та обмеження доступу до чатів, що підвищує безпеку системи

Щоб налаштувати Telegram-бота, необхідно створити його через офіційного бота Telegram під назвою BotFather. Після реєстрації бота користувач отримує унікальний токен, за допомогою якого система отримує доступ до API. Надсилання

повідомлень реалізується за допомогою HTTP POST-запитів на адресу https://api.telegram.org/bot<TOKEN>/sendMessage, з відповідними параметрами чату та тексту повідомлення. У Zabbix ця інтеграція оформлюється через Media Type типу Script, що запускає зовнішній shell або Python-скрипт при виникненні події. Це забезпечує миттєве інформування адміністратора без використання додаткових сервісів поштового або SMS-зв'язку [34].

У сучасній інженерній практиці все частіше застосовується віртуалізація як ефективний засіб тестування, моделювання та розгортання інформаційних систем. Це особливо актуально для студентських досліджень, навчальних лабораторій та проектів з обмеженим бюджетом, де відсутній доступ до промислового обладнання або реальної інфраструктури. У рамках даної дипломної роботи було вирішено відмовитися від фізичних серверів і використовувати віртуальне середовище, створене за допомогою Oracle VirtualBox [36].

Oracle VirtualBox – це кросплатформенний інструмент віртуалізації, який дозволяє запускати кілька операційних систем одночасно на одному фізичному комп'ютері. У рамках реалізації кіберфізичної системи моніторингу VirtualBox надає можливість моделювати реальну мережу без потреби в додатковому обладнанні, що особливо актуально для навчальних або тестових середовищ.

Серед ключових технічних переваг – підтримка основних типів мережевих адаптерів, включно з NAT та Bridge, що дозволяє налаштувати взаємодію віртуальних машин між собою та з зовнішньою мережею. Також VirtualBox дозволяє виділяти необхідну кількість ресурсів для кожної віртуальної машини (CPU, RAM, HDD), що робить симуляцію більш наближеною до реальних умов. Додатково, функціональність знімків (snapshots) дозволяє зберігати поточний стан системи і повертатися до нього у разі збоїв чи помилок у конфігурації [37].

У нашому випадку, використання VirtualBox значно спростило процес тестування та відлагодження кіберфізичної системи моніторингу - від розгортання Zabbix Server до конфігурації агентів, моделювання хостів і подій. Це дозволило уникнути витрат на фізичну інфраструктуру та забезпечити повну

контрольованість середовища. Вибір цього програмного забезпечення забезпечив повну гнучкість у створенні симульованого середовища, яке максимально наближене до реальної структури кіберфізичної системи моніторингу оптичної мережі.

У якості серверної операційної системи для побудови кіберфізичної системи моніторингу було обрано Ubuntu Server 24.02 LTS**.** Цей дистрибутив є одним з найпопулярніших у сфері інфраструктурних рішень завдяки стабільності, активному розвитку та офіційній підтримці спільноти [38]. Версія LTS (Long Term Support) гарантує оновлення безпеки та підтримку протягом кількох років, що дозволяє не турбуватись про регулярні оновлення та несумісність компонентів.

Ubuntu Server відзначається широкою підтримкою відкритого ПЗ, включно з інструментами, які критично важливі для розгортання системи моніторингу - такими як Zabbix, Apache2, MariaDB, PHP, Net-tools та інші. Усі ці пакети доступні безпосередньо з офіційних репозиторіїв, що спрощує автоматизоване встановлення та забезпечує максимальну сумісність між компонентами [39].

Ще однією суттєвою перевагою Ubuntu Server є його легкість і мінімалізм - базове середовище без графічного інтерфейсу споживає дуже мало ресурсів, що особливо важливо при роботі у віртуальному середовищі. Це дозволяє запускати повноцінний стек моніторингу навіть на обмежених апаратних ресурсах, наприклад 2 ядра CPU, 2 GB RAM і 20 GB HDD, як було реалізовано в рамках даного проєкту.

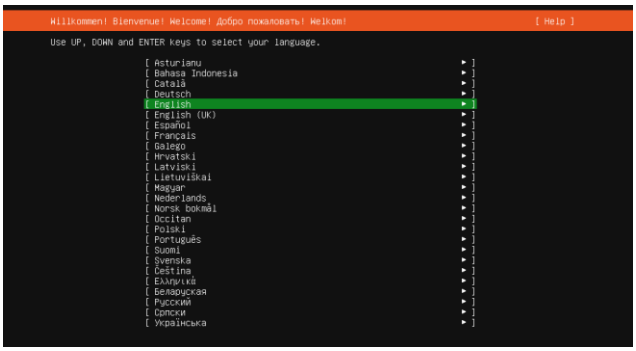


Рисунок 2.2 – Графічний інтерфейс та зображення встановлення Ubuntu Server 24.04

2.2 Функційні вимоги програмного забезпечення

У процесі розробки кіберфізичної системи моніторингу оптичної мережі одним із ключових етапів є формування функційних вимог до її програмного забезпечення. Ці вимоги визначають, якими саме характеристиками повинна володіти система, щоб забезпечити її відповідність потребам моніторингу, своєчасного реагування на інциденти, збереження історичних даних, а також надання користувачу зручного інтерфейсу для аналізу та керування.

Функційні вимоги ґрунтуються на концепції безперервного контролю за станом телекомунікаційної інфраструктури. Система повинна мати змогу здійснювати цілодобовий збір телеметричної інформації з мережевого обладнання, зокрема з вузлів оптичної мережі. Оскільки оптичні канали є критично важливими для високошвидкісної передачі даних, будь-яка втрата зв'язку, зниження якості сигналу чи фізичне пошкодження лінії повинні бути зафіксовані і оброблені негайно [22, 23].

Програмне забезпечення має забезпечувати автоматичний моніторинг основних показників – таких як затримка, доступність вузлів, навантаження на центральний процесор та пам’ять пристроїв, температура, стан каналів передачі, тощо. Важливо, щоб система підтримувала як агентський (через Zabbix Agent), так і безагентський підхід до збору інформації. Це дозволить інтегрувати різноманітні типи обладнання, включно з промисловими комутаторами, оптичними трансиверами, або навіть віртуальними пристроями [24, 25].

Однією з ключових вимог є реалізація механізму сповіщення про інциденти. У випадку відхилення показників від заданих меж, система повинна формувати тригер, що активує подію. Ця подія має бути передана адміністратору через інтегрований канал - у нашому випадку, через Telegram-бота. Повідомлення повинні бути короткими, інформативними, містити точний час події, ідентифікатор вузла та опис проблеми. Надзвичайно важливо, щоб система реагувала в режимі реального часу або з мінімальною затримкою. Програмне забезпечення повинно

забезпечувати можливість цілодобового збору даних про стан мережевої інфраструктури. Це охоплює не лише базові параметри, як-от доступність пристроїв або навантаження на системні ресурси, а й більш глибоку інформацію про якість роботи каналів зв'язку, стабільність інтерфейсів, а також виявлення аномалій у роботі мережевого обладнання. Важливим є не лише спостереження за вже наявними параметрами, а й можливість розширення системи новими типами даних у разі зміни або ускладнення інфраструктури. Така гнучкість досягається за рахунок модульної архітектури системи, підтримки шаблонів моніторингу, можливості інтеграції зі сторонніми інструментами, такими як SNMP або IPMI.

Ключовим напрямом, який програмне забезпечення повинне реалізовувати, є виявлення порушень у роботі мережі та своєчасне реагування на них. Для цього необхідно передбачити механізм побудови тригерів, що визначають допустимі межі значень тих чи інших параметрів [26]. Перевищення цих меж або втрата зв'язку з обладнанням має автоматично породжувати подію, яка далі передається в систему нотифікацій. Оповіщення повинні бути оперативними, інформативними та доступними для адміністратора з будь-якого пристрою, тому особливо важливою є інтеграція з месенджерами, зокрема Telegram, що дозволяє забезпечити мобільність управління.

Програмне забезпечення повинне підтримувати централізоване зберігання історичних даних. Це забезпечується через використання повноцінної бази даних, яка зберігає кожен показник у хронологічному порядку. Такий підхід дозволяє не лише відстежувати поточний стан мережі, а й проводити ретроспективний аналіз, виявляти закономірності, планувати розширення інфраструктури або модернізацію обладнання. Крім того, на основі накопиченої статистики можна формувати дашборди та графіки, що є надзвичайно корисним для візуалізації навантажень та оцінки ефективності управління мережею.

Візуалізація, як функційна складова системи, повинна забезпечувати гнучке представлення даних. Користувач має мати змогу обирати, які параметри виводити, за який період, у якому форматі. Наприклад, для адміністраторів важливо бачити

реальний стан хостів у вигляді кольорових індикаторів, тоді як для аналітиків може бути актуальним перегляд тенденцій навантаження у вигляді діаграм. Особливо важливо, щоб система дозволяла створювати персоналізовані дашборди з потрібним набором графіків, діаграм і таблиць.

Ще однією вимогою до системи є її безпека. Всі дані, що передаються між агентами та сервером, повинні бути захищені. Це досягається шляхом використання протоколів шифрування, авторизації користувачів, обмеження прав доступу та ведення журналу дій. Адміністративний інтерфейс повинен бути захищеним за допомогою HTTPS-з'єднання, а облікові записи повинні мати чітке розмежування за ролями та дозволами. З міркувань безпеки також має бути можливість контролювати спроби входу до системи, перегляд змін конфігурації, відновлення даних із резервної копії.

Особливу увагу варто приділити масштабованості програмного забезпечення. Навіть якщо на першому етапі планується моніторинг одного або кількох хостів, архітектура системи має передбачати можливість масштабування до десятків або сотень пристроїв без необхідності кардинальних змін. Це означає, що система повинна мати добре організовану ієрархію шаблонів, груп користувачів, логіку застосування тригерів до нових хостів, а також оптимізовану обробку великої кількості даних [27].

Zabbix Frontend виконує функцію основного інструмента взаємодії користувача із системою моніторингу. Інтерфейс реалізовано у вигляді веб-додатку, що працює через браузер і забезпечує зручний доступ до всіх функцій керування мережею. Це рішення не потребує встановлення додаткового програмного забезпечення на клієнтський пристрій - адміністратор може керувати системою з будь-якої точки, маючи лише доступ до мережі. Веб-інтерфейс працює через порт 80 (HTTP) або 443 (HTTPS), залежно від конфігурації веб-сервера Apache2.

Функціонально інтерфейс дозволяє виконувати широкий спектр задач: перегляд і створення хостів, конфігурація шаблонів моніторингу, налаштування

тригерів, перегляд графіків, подій та логів. Розділ Configuration – Hosts надає можливість додавати нові вузли моніторингу, вказувати їхні IP-адреси, групи, інтерфейси зв'язку та прив'язувати шаблони. Розділ Monitoring – Latest data дозволяє у режимі реального часу переглядати метрики з хостів. Завдяки структурованому меню та логічній навігації, інтерфейс є інтуїтивно зрозумілим і легко адаптується навіть для користувачів без великого досвіду в системному адмініструванні [27].

Крім цього, функційною вимогою є підтримка масштабованості. Система повинна дозволяти додавання нових вузлів без необхідності перебудови всієї архітектури. Це реалізується через централізовану модель, де всі нові агенти реєструються на сервері за допомогою ключа або автоматичного виявлення через активний агент.

Особливу увагу необхідно звернути на відмовостійкість. У випадку недоступності агента або збоїв у передачі даних, система повинна або повторити запит через заданий інтервал, або записати відповідний статус у лог. Усі події мають бути належно логовані і доступні для перегляду.

Нарешті, система повинна підтримувати експортування даних та звітів у форматах CSV, JSON або PDF. Це дає змогу проводити аналіз поза межами самої платформи, інтегрувати моніторингові дані з іншими інформаційними системами або використовувати їх для звітності [30, 33].

Таким чином, програмне забезпечення для кіберфізичної системи моніторингу оптичної мережі повинне поєднувати в собі низку важливих характеристик: надійність, стабільність, безперервність моніторингу, гнучкість конфігурації, візуальну наочність, високий рівень безпеки та масштабованість. Всі ці властивості є ключовими для створення інструменту, який буде ефективно використовуватись у реальному телекомунікаційному середовищі для забезпечення якісного контролю за станом оптичної інфраструктури. Така система повинна не просто фіксувати події, а допомагати передбачати відмови, аналізувати тенденції та реагувати на загрози, що виникають [14]. У сучасних умовах зростання

обсягів даних і підвищених вимог до стабільності каналів зв'язку, саме така програмна основа є необхідною для впровадження на підприємствах, що працюють із критично важливими оптичними мережами.

2.3 Проектування роботи та методологічний підхід

У процесі розробки кіберфізичної системи моніторингу оптичних мереж важливо обрати методологічний підхід, що поєднує гнучкість, масштабованість та адаптивність. Такий підхід дозволяє забезпечити своєчасне виявлення проблем, автоматичне інформування відповідального персоналу, а також створення зручного інтерфейсу для моніторингу та аналізу стану мережі. Кіберфізична система складається з кількох тісно пов'язаних компонентів, які спільно реалізують задачі збору, аналізу, збереження і передачі інформації. Серцем рішення є система моніторингу Zabbix, яка реалізує збір телеметрії з хостів за допомогою агента. Zabbix Server виконує обробку даних, керує тригерами та сповіщеннями, а також забезпечує доступ до історичних метрик через веб-інтерфейс. Окремим блоком є Telegram-бот, що забезпечує пряме сповіщення адміністратора про події в мережі у зручному месенджері. На нижньому рівні рішення охоплює агентів, встановлених на хости, які симулюють або зчитують дані з оптичного обладнання [31].

Застосування Zabbix обумовлене його стабільністю, широким набором інтеграцій, гнучкістю у створенні шаблонів моніторингу та підтримкою повідомлень через API. Додатковою перевагою є зручна реалізація на базі популярних серверних дистрибутивів Linux, включно з Ubuntu Server. Веб-інтерфейс Zabbix дозволяє створювати адаптивні дашборди, відслідковувати тригери у реальному часі та керувати хостами[32].

Telegram було обрано як канал оповіщення через його швидкість, надійність, простоту налаштування та високу популярність серед користувачів. Інтеграція здійснюється через офіційне API Telegram, що дозволяє реалізувати бот, який приймає сповіщення із Zabbix.

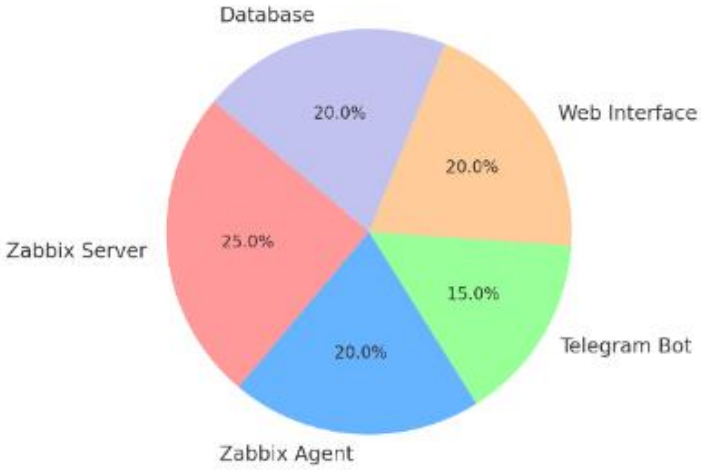


Рисунок 2.3 – Компоненти кіберфізичної системи моніторингу.

Уся інформація, що циркулює в системі моніторингу, проходить послідовний шлях через кілька логічно відокремлених етапів - від моменту формування телеметричних даних на рівні обладнання до їх остаточного виведення у вигляді візуалізації або сповіщення через зовнішні сервіси, зокрема Telegram. Цей процес має бути повністю безперервним, з мінімальними затримками, щоб забезпечити актуальність і достовірність усіх подій. Ключовим завданням при проєктуванні є забезпечення надійності на кожному з етапів обробки даних - включаючи агента, що виконує первинний збір параметрів, серверну частину Zabbix, яка здійснює агрегацію та аналіз, базу даних (MariaDB), де відбувається збереження інформації, а також логування, яке фіксує усі транзакції та події системи [31, 32]. Така модель дозволяє не тільки ідентифікувати джерело можливих проблем, але й реалізувати резервування - наприклад, через регулярне створення копій бази даних або дублювання критичних служб.

З архітектурної точки зору система логічно поділяється на три умовні шари: інфраструктурний, логічний та інтерфейсний. Інфраструктурний рівень включає всі технічні компоненти, необхідні для базового функціонування - це агенти, встановлені на вузлах моніторингу, сервер Zabbix, база даних, веб-сервер Apache, а також усі пов’язані з ними конфігурації та налаштування. На логічному рівні

виконується основна обробка отриманих даних: система формує тригери, які порівнюють параметри з допустимими межами, генерує події та ініціює відповідні реакції згідно із заданими сценаріями. Нарешті, інтерфейсний шар відповідає за представлення інформації користувачеві – через веб-інтерфейс Zabbix Frontend із можливістю побудови дашбордів, перегляду журналів подій та керування об'єктами, а також через Telegram-бота, який реалізує функцію оперативного інформування. Такий багаторівневий поділ дозволяє краще структурувати систему як з точки зору логіки її роботи, так і з точки зору розгортання, масштабування, налагодження та технічної підтримки в процесі експлуатації.

2.4 Проектування майбутньої системи

Проєктування кіберфізичної системи моніторингу оптичної мережі включає комплексний підхід до формування логічної структури, що забезпечує ефективну взаємодію всіх компонентів системи [35]. Визначення чітких зв’язків між елементами – від датчиків і агентів збору даних до серверної інфраструктури та інтерфейсів користувача – дозволяє створити надійну архітектуру, здатну адаптуватись до змін у мережевому середовищі.

Особливу увагу приділено забезпеченню масштабованості системи, що дозволяє інтегрувати нові вузли та розширювати глибину моніторингу без суттєвих втрат у продуктивності або складності адміністрування. Крім того, проєкт передбачає реалізацію безперервного збору телеметричних даних у реальному часі, що є критично важливим для своєчасного виявлення відхилень і підтримки високого рівня доступності мережі.

Для досягнення поставлених цілей у системі використовуються сучасні технології з відкритим вихідним кодом, які мають широкі можливості кастомізації та інтеграції через API. Це забезпечує не лише прозорість роботи системи, але й дозволяє легко впроваджувати додаткові функції або інтегрувати зовнішні сервіси для оповіщення, аналізу чи візуалізації даних. Використання таких інструментів дозволяє знизити вартість розробки та обслуговування, одночасно підтримуючи

високу надійність та гнучкість у роботі системи. Завдяки цьому архітектурне рішення відповідає сучасним вимогам кіберфізичних систем та забезпечує основу для подальшого розвитку та масштабування.

Загальна структура майбутньої системи поділяється на кілька логічних рівнів: Інфраструктурний рівень майбутньої системи забезпечує фізичну та віртуальну основу для функціонування всіх компонентів. До його складу входять сервери (реальні або віртуальні машини), які розгортаються у VirtualBox та пов’язані в єдину внутрішню мережу [36]. Основним вузлом є сервер з операційною системою Ubuntu Server, на якому встановлюється Zabbix Server – центральний компонент системи моніторингу. Поряд із ним інсталюється MariaDB, що виконує роль СКБД для зберігання всієї інформації про вузли моніторингу, події, тригери, графіки та історію сповіщень. Для реалізації веб-доступу до інтерфейсу Zabbix використовується веб-сервер Apache, налаштований з урахуванням базових принципів безпеки (закриті зайві порти, використання HTTPS при потребі тощо). На кожному клієнтському вузлі або віртуальній машині, які мають бути об’єктами моніторингу, встановлюється Zabbix Agent, що передає інформацію про стан системи до центрального сервера. Інфраструктурний рівень також враховує параметри системного моніторингу, такі як доступність CPU, RAM, файлової системи, мережевих інтерфейсів, що дозволяє створити повноцінну картину функціонування об'єктів мережі.

Логічний рівень відповідає за інтелектуальну обробку зібраних даних і реалізацію алгоритмів реагування. У межах цього рівня проєктуються шаблони моніторингу, які включають параметри перевірки, періодичність збору даних та набір ключів (items) для різних типів обладнання або сервісів [37]. На основі цих шаблонів створюються тригери - логічні умови, які визначають, коли стан об'єкта відхиляється від норми (наприклад, перевищення навантаження на процесор або втрата доступності вузла). Тригери можуть мати різні рівні пріоритету (від інформаційного до критичного), що дозволяє ранжувати сповіщення та фокусувати увагу адміністратора на критичних подіях. Також розробляються сценарії

автоматизованого реагування, наприклад, надсилання повідомлень, виконання скриптів чи повторна перевірка проблемного вузла через певний інтервал часу. Таким чином, логічний рівень виконує роль ядра інтелектуальної обробки даних у системі.

Інтерфейсний рівень забезпечує взаємодію користувача з системою моніторингу через два основні канали - графічний веб-інтерфейс Zabbix та систему оперативного інформування, реалізовану за допомогою Telegram-бота. Веб-інтерфейс дозволяє адміністратору переглядати графіки, карти мережі, журнали подій, налаштовувати нові вузли моніторингу, шаблони та правила тригерів. Інтерфейс підтримує розширені фільтри та сортування, що робить аналіз подій зручним навіть у великих мережах. Telegram-бот, у свою чергу, реалізує мобільний канал сповіщень, надсилаючи повідомлення про спрацювання тригерів безпосередньо в чат. Бот також підтримує двосторонню взаємодію – користувач може надіслати команду для отримання поточного статусу системи або підтвердження реагування на подію. Така побудова інтерфейсного рівня підвищує швидкість реакції на інциденти, дозволяє оперативно втручатися в роботу системи та спрощує моніторинг у реальному часі навіть за відсутності постійного доступу до веб-інтерфейсу [38].

Таке розділення дозволяє досягти високої гнучкості в адмініструванні, спростити налаштування та забезпечити розширення системи в майбутньому, наприклад, через додавання нових типів обладнання, нових вузлів чи розширених сценаріїв сповіщення.

Проектування системи слід починати з формування архітектури рішення, яка включає кілька взаємопов’язаних рівнів:

1. Рівень збору даних. На цьому рівні планується використання програмного агента, встановленого на вузлових пристроях (реальних чи віртуальних), який виконує збір телеметричної інформації – навантаження на процесор, стан каналів зв’язку, лог-файли, затримки та втрати пакетів. У разі емульованого середовища, ці пристрої можуть симулювати роботу оптичного обладнання.

Для цього передбачається використання Zabbix Agent, як одного з найбільш стабільних, налаштовуваних та підтримуваних агентів для збору інформації в середовищах Linux/Windows [39].

2. Рівень моніторингу та обробки. Zabbix Server виступає ключовим елементом усієї архітектури кіберфізичної системи моніторингу, реалізуючи функції центрального аналізатора і диспетчера даних. Його основне завдання – забезпечити безперервний збір, обробку та збереження телеметрії з усіх підключених вузлів мережі. Дані надходять як у пасивному режимі (агенти відповідають на запити сервера), так і в активному (агенти самі ініціюють передачу даних). Після надходження інформації сервер аналізує її згідно з попередньо налаштованими тригерами - логічними умовами, які визначають, чи перевищено порогові значення, що сигналізують про інцидент.

Zabbix Server також зберігає всі історичні значення у базі даних MariaDB Це дозволяє не лише переглядати поточні стани об'єктів, а й формувати графіки з динамікою змін параметрів у часі, виявляти тренди, а також здійснювати глибокий аналіз на основі накопиченої статистики. Завдяки наявності повного історичного архіву, система здатна підтримувати середньо - та довгостроковий аналіз продуктивності окремих вузлів або сервісів, що є особливо важливим у середовищах з високими вимогами до надійності та стабільності інфраструктури. Запити до бази даних оптимізовано для ефективного зберігання та швидкого доступу до великих обсягів даних, що забезпечує мінімальні затримки при відображенні графіків та побудові звітів.

Однією з найпотужніших можливостей Zabbix є використання шаблонів, які суттєво спрощують процес масштабування системи моніторингу. Шаблон у Zabbix - це набір наперед визначених елементів даних, тригерів, графіків, сценаріїв та дій, які можуть бути прив'язані до будь-якої кількості хостів. Завдяки цьому адміністратор може за кілька кліків застосувати однакові правила моніторингу до десятків або навіть сотень вузлів, що унеможливлює помилки, притаманні ручному налаштуванню, і значно пришвидшує процес розгортання нових об'єктів у системі.

Крім цього, використання шаблонів забезпечує централізоване оновлення логіки моніторингу: зміни, внесені в шаблон, автоматично застосовуються до всіх пов'язаних хостів, що підвищує гнучкість та керованість системою.

Автоматизація реагування є ще одним важливим компонентом, що реалізується на рівні тригерів та дій. При фіксації відхилення від норми система не лише реєструє подію, а й може негайно ініціювати відповідні заходи. До таких заходів належать надсилання сповіщень через Telegram-бота, створення записів у логах, а також запуск зовнішніх скриптів або команд, які виконують, наприклад, перезапуск служби, блокування доступу або створення резервної копії. Завдяки такій інтеграції моніторингова система перетворюється з пасивного спостерігача на активний елемент інфраструктури, здатний миттєво реагувати на події, що загрожують стабільній роботі мережі або серверів. У результаті досягається не лише підвищена швидкість реагування на інциденти, а й зменшення часу простою критичних сервісів [29, 31].

3. Рівень візуалізації системі моніторингу відіграє ключову роль у забезпеченні зручної та оперативної взаємодії користувача з поточним станом мережевої інфраструктури. Для цього використовується веб-інтерфейс Zabbix Frontend, який надає широкі можливості для побудови інформаційних панелей, аналітичних графіків, перегляду журналів подій, сповіщень, а також централізованого керування усіма елементами системи - від шаблонів до окремих хостів. Веб-інтерфейс працює у реальному часі, оновлюючи дані без необхідності ручного перезавантаження сторінок, що дає змогу адміністраторам миттєво відстежувати зміни в стані мережі та приймати обґрунтовані рішення. Система підтримує створення дашбордів з можливістю додавання віджетів для відображення ключових показників, таких як навантаження на процесор, обсяг спожитої пам’яті, мережеві затримки, активність інтерфейсів, статус доступності хостів, а також кількість та рівень активних тригерів. Крім того, передбачено реалізацію спеціалізованої інформаційної панелі, яка буде використовуватись як центральне візуальне табло. На цьому табло виводитимуться агреговані дані про

стан мережі, включно з критичними показниками, такими як поточне навантаження CPU на ключових серверах, наявність затримок у передачі даних, стабільність і якість каналів зв'язку. Таке візуальне подання забезпечує швидкий аналіз ситуації без необхідності глибокого занурення в технічні деталі, дозволяє оперативно локалізувати потенційні проблеми та формує основу для ефективного управління інфраструктурою в режимі реального часу.

4. Рівень оповіщення. Особливістю проєкту стане реалізація оповіщення через Telegram, що дозволяє миттєво сповіщати адміністратора у разі настання критичних подій (відмова пристрою, перевантаження, втрата зв'язку тощо).

У межах проєкту передбачається інтеграція Telegram як основного каналу сповіщення. Це дозволяє миттєво доставляти повідомлення про інциденти (наприклад, перевантаження процесора, відмова агента, недоступність вузла) без затримок та прив'язки до електронної пошти чи SMS [40].

Для реалізації цього підходу буде створено Telegram-бота через сервіс BotFather, після чого розроблено скрипт на Bash або Python для надсилання повідомлень через Telegram API. Далі цей скрипт інтегрується в систему Zabbix як новий тип сповіщення (Media Type - Script), після чого прив'язується до облікового запису користувача. Завдяки налаштуванню дій (Actions), бот автоматично надсилатиме повідомлення при спрацюванні відповідних тригерів. Такий підхід забезпечує гнучкість, мобільність і зручність в управлінні інцидентами.

5. Мережева структура проєктованої системи ґрунтується на створенні ізольованого локального середовища, яке не має прямого доступу до зовнішніх мереж, зокрема інтернету. Такий підхід забезпечує високий рівень безпеки, що критично важливо при тестуванні й експлуатації систем моніторингу, особливо у випадках, коли йдеться про обробку чутливих даних або конфігурацій корпоративної інфраструктури. Ізоляція також дозволяє точно контролювати мережеву активність, уникати стороннього втручання та створювати повністю контрольоване середовище для аналізу реакції системи на змодельовані події.

Для побудови обчислювальної інфраструктури було використано VirtualBox

- гнучке і безкоштовне середовище віртуалізації, яке дозволяє створювати віртуальні машини з різними конфігураціями апаратного забезпечення. У межах цього середовища було розгорнуто декілька взаємопов'язаних віртуальних машин: одна з них виконує роль сервера моніторингу на базі Ubuntu Server, інші діють як клієнтські вузли, на яких встановлено Zabbix Agent для передачі телеметричних даних [34, 35]. Усі машини об'єднані в одну внутрішню мережу VirtualBox (Internal Network), що забезпечує обмін даними між вузлами без участі зовнішніх маршрутизаторів або шлюзів.

Таке середовище дозволяє не лише моделювати роботу системи у максимально наближених до реальних умовах, а й здійснювати гнучке конфігурування мережевих сценаріїв. Наприклад, можна легко змінювати кількість клієнтів, задавати специфічні параметри мережевого трафіку, тестувати поведінку системи при навантаженні чи втраті зв'язку з окремими вузлами. Крім того, внутрішня мережа VirtualBox дозволяє створити симульовану схему передачі даних, де імітуються типові події для моніторингу: перевищення навантаження, збій сервісів, або затримка відповіді. Це дає змогу повноцінно перевірити реакції системи, логіку сповіщень і достовірність виведених метрик у безпечному ізольованому середовищі [32].

6. Проєктована система моніторингу закладається як масштабована за своєю архітектурою, що дозволяє без суттєвих змін у структурі розширювати її в обох напрямках – як у площині кількості вузлів (агентів), так і за рівнем глибини та деталізації моніторингу.

Zabbix, як основа системи, спроектований для роботи в середовищах будь-якої складності – від малих локальних мереж до багаторівневих корпоративних інфраструктур. У разі збільшення кількості хостів достатньо лише реєстрації нового агента, призначення його до відповідної хост-групи та прив'язки до необхідного шаблону, після чого система автоматично починає збір і обробку даних. Жодних змін до основного ядра або конфігурації сервера при цьому не потрібно, що забезпечує зручність у підтримці та розвиткові. Такий підхід дозволяє

не лише ефективно керувати поточним навантаженням, а й бути готовим до масштабування відповідно до зростання інфраструктури або інтеграції нових елементів[39].

Гнучкість реалізації досягається завдяки широким можливостям у налаштуванні шаблонів, тригерів, груп хостів та умов сповіщень. Наприклад, шаблони можуть містити як базові елементи (перевірка доступності, навантаження, використання ресурсів), так і вузькоспеціалізовані - для моніторингу веб-сервісів, серверів баз даних, мережевих пристроїв. При цьому всі елементи шаблонів легко змінюються, клонуються або комбінуються без порушення загальної логіки системи. Для ефективного управління великою кількістю об’єктів застосовуються розподілені хост-групи – вони дозволяють об'єднувати вузли за функціональними, фізичними або логічними ознаками (наприклад, "офісні ПК", "мережеве обладнання", "виробничі сервери"). Це спрощує навігацію в системі, полегшує масове оновлення конфігурацій та створення окремих звітів і дашбордів для кожної категорії.

Централізовані дашборди доповнюють можливості масштабування і гнучкого керування, дозволяючи швидко адаптувати візуальні інструменти до нових вимог. Зміни в структурі моніторингу, додавання нових об'єктів або параметрів миттєво відображаються на панелях – без потреби створювати інтерфейс заново. Оскільки дашборди підтримують шаблонізацію, можна створити стандартні макети для різних відділів, типів пристроїв або сервісів. Усі ці компоненти разом дозволяють системі залишатися максимально адаптивною, легко оновлюваною та масштабованою без втрати стабільності або продуктивності. Такий підхід забезпечує надійне функціонування системи в умовах змін, зростання навантажень або розширення функціональності без потреби в повній реконфігурації інфраструктури [22, 23].

7. Безпека і надійність У проектуванні враховується необхідність. Реалізованої системи автентифікації що передбачає перевірку облікових даних користувачів при вході до інтерфейсу керування та моніторингу. Для кожного

користувача визначаються чіткі рівні доступу, що дозволяє розмежувати адміністративні функції, функції перегляду та аналітики, а також обмежити дії звичайних операторів. Такий підхід мінімізує ризики несанкціонованих змін конфігурації системи чи доступу до конфіденційної інформації. У разі багатокористувацького доступу застосовуються ролі (рольова модель доступу), що забезпечує масштабованість та централізоване управління правами.

Окремим елементом системи є Telegram-бот, через якого здійснюється отримання сповіщень та взаємодія з користувачами. Щоб запобігти несанкціонованому доступу до бота, реалізовано контроль авторизованих ID користувачів: лише попередньо зареєстровані облікові записи мають право отримувати дані та надсилати команди [25]. Крім того, для забезпечення надійності функціонування системи реалізовано регулярне резервне копіювання бази даних. Копії зберігаються у захищеному середовищі з можливістю відновлення у разі збоїв, втрати даних або пошкодження основного сховища. Це дозволяє гарантувати збереження інформації про стан мережі та історію моніторингу навіть у випадку критичних збоїв системи [27, 28].

2.5 Висновки до розділу

У другому розділі було виконано глибокий аналіз предметної області, що охоплює розвиток систем моніторингу в контексті кіберфізичних технологій, зокрема в телекомунікаційних структурах, які використовують оптичні канали передачі даних. Було досліджено як історичний аспект становлення засобів автоматизованого контролю мереж, так і сучасні інструменти, що забезпечують функціонування інтелектуальних систем моніторингу.

Проведено порівняльний аналіз наявних рішень, що вже впроваджені в IT-інфраструктурах різного масштабу. Встановлено, що найперспективнішими для впровадження у симуляційній кіберфізичній системі є відкриті системи моніторингу, зокрема Zabbix, завдяки широкому функціоналу, активному розвитку, наявності великої спільноти, підтримці стандартних протоколів збору

телеметрії та можливості інтеграції з зовнішніми сервісами [28, 29].

Під час дослідження було встановлено, що сучасна інфраструктура потребує не лише збору інформації про стан мережевого обладнання, але й адаптивного реагування на позаштатні ситуації. У зв'язку з цим особливу увагу приділено механізмам оповіщення – зокрема, через Telegram API, що дозволяє забезпечити миттєву доставку повідомлень адміністратору, незалежно від географічного положення чи типу пристрою.

Окремо було визначено функціональні та нефункціональні вимоги до системи, з урахуванням специфіки моніторингу оптичного середовища. Основними критеріями стали: гнучкість масштабування, підтримка віртуального середовища, сумісність з Linux-системами, автономність роботи, простота візуалізації даних та адаптивність до реального середовища.

Методологічні підходи до реалізації системи ґрунтуються на модульному та поетапному принципі. Це дозволяє реалізовувати окремі компоненти незалежно один від одного, забезпечуючи при цьому загальну узгодженість у роботі системи. Розглянуто різні варіанти реалізації: як апаратно-орієнтовані (на базі мікроконтролерів та одноплатних комп’ютерів), так і програмні (з використанням віртуального середовища Ubuntu Server, платформи Zabbix, Telegram API) [31].

Поставлена задача сформульована як комплексна – від аналізу технологічних підходів та існуючих рішень до розробки архітектури майбутньої симуляційної системи та її теоретичного обґрунтування. Передбачено створення повноцінної системи моніторингу, здатної працювати в умовах віртуалізованого середовища, з імітацією роботи реального телекомунікаційного обладнання, з можливістю виявлення аномалій, відстеження навантаження та реагування на інциденти.

У результаті проведеного аналізу та моделювання в межах другого розділу було сформовано повноцінне теоретичне підґрунтя для подальшої практичної реалізації системи моніторингу. Було детально розглянуто ключові аспекти архітектури, включаючи поділ на інфраструктурний, логічний та інтерфейсний рівні, що дозволяє досягти високої структурованості, модульності та гнучкості в

майбутньому розгортанні та масштабуванні. Значну увагу приділено забезпеченню безперервності потоку даних – від телеметричних джерел до рівня візуалізації та сповіщення. Це охоплює не лише засоби збору та зберігання інформації, але й питання резервування, логування, безпеки доступу та централізованого адміністрування [15, 17]. Усі розглянуті елементи були поєднані в уніфіковану модель системи, яка відповідає сучасним вимогам до кіберфізичних систем спостереження та контролю.

Також здійснений аналіз підтвердив доцільність вибору конкретних технологій та інструментів, таких як Zabbix, MariaDB, Telegram-бот та використання шаблонізованих механізмів моніторингу. Обґрунтовано вибір інструментів віртуалізації, що забезпечують ізоляцію середовища і повторюваність розгортання системи [32, 33]. Ретельне планування структури компонентів, логіки обробки подій та механізмів візуалізації створює основу для ефективного управління складною телекомунікаційною інфраструктурою. Таким чином, у цьому розділі було закладено не лише концептуальні, а й архітектурно-функціональні рамки, що слугуватимуть базисом для практичної реалізації проєкту, яка буде детально описана у наступному розділі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

3.1 Принцип роботи програмного забезпечення модуля зчитування компонентів навколишнього середовища

У даному проєкті Zabbix виступає як основне ядро системи: він відповідає за збір, зберігання, обробку та візуалізацію даних. У парі з ним функціонує система сповіщень, яка реалізована через Telegram-бота. Оскільки проєкт орієнтовано на роботу у віртуальному середовищі, особливу увагу було приділено точності моделювання, наближеності до реальних умов, а також масштабованості.

Модуль зчитування даних у межах кіберфізичної системи моніторингу оптичної мережі є критичним компонентом, який забезпечує безперервне отримання телеметричних показників із хостів – реальних або віртуалізованих вузлів. У даному проєкті реалізація виконувалась у віртуальному середовищі, що дозволило моделювати поведінку окремих елементів мережі без необхідності використання дорогого оптичного обладнання.

Програмна реалізація модулів зчитування базується на застосуванні Zabbix Agent, який працює на кожному вузлі та виконує роль постачальника даних. Агент може отримувати інформацію з використанням низькорівневих системних викликів або через спеціалізовані модулі, що розширюють його функціональність.

На логічному рівні взаємодія виглядає наступним чином:

[Zabbix Agent] - [Zabbix Server] - [База даних] - [Zabbix Frontend] - [Користувач]

Zabbix Agent це легковагий демон, встановлений на клієнтську віртуальну машину, що виступає як хост, який підлягає моніторингу. Агент працює у фоновому режимі як системна служба (zabbix-agent.service) і може бути сконфігурований як у пасивному режимі (очікує запитів від сервера на TCP-порті 10050), так і в активному режимі (ініціює підключення до сервера через порт

10051). У конфігураційному файлі агента (/etc/zabbix/zabbix_agentd.conf) задається IP-адреса Zabbix Server, перелік дозволених параметрів, hostname хоста, а також дозволи на користувацькі ключі.

Агент працює за двома моделями:

– Active – сам надсилає дані на сервер;

– Passive – відповідає на запити сервера.

Для забезпечення безпеки зв'язку використовується TLS-шифрування або фаєрволи з whitelisted IP.

Після встановлення агента на віртуальний хост (наприклад, Client-VM) конфігураційний файл /etc/zabbix/zabbix_agentd.conf налаштовується відповідно до IP-адреси сервера та режиму передачі.

Приклад конфігурації:

```
Server=192.168.1.100
ServerActive=192.168.1.100
Hostname=Client-VM
LogFile=/var/log/zabbix/zabbix_agentd.log
EnableRemoteCommands=1
```

Рисунок 3.1 – Приклад конфігурації серверу через консоль.

```
sudo systemctl restart zabbix-agent
sudo systemctl enable zabbix-agent
```

Рисунок 3.2 – Запуск Zabbix-agent.

Zabbix Server є центральним компонентом всієї системи моніторингу – саме він відповідає за координацію взаємодії між агентами, базою даних та веб-інтерфейсом. У запропонованій архітектурі Zabbix Server розгорнуто на віртуальній машині з Ubuntu Server 24.04, яка виступає у ролі ядра всієї кіберфізичної системи моніторингу. Сервер ініціює регулярні запити до агентів (або отримує дані в пасивному режимі), опрацьовує отримані метрики та порівнює

їх із заданими пороговими значеннями, які описані в конфігурації тригерів. У разі виявлення відхилення від нормальних параметрів сервер генерує подію, яка, своєю чергою, може активувати відповідну реакцію - наприклад, надсилання сповіщення чи запуск зовнішнього скрипта.

Технічно Zabbix Server працює як служба системи, яка запускається через systemd під ім'ям zabbix-server.service. Його конфігураційний файл знаходиться за адресою /etc/zabbix/zabbix_server.conf, у якому задаються основні параметри функціонування: дані для підключення до бази DBHost, DBName, DBUser, DBPassword, кількість паралельних процесів збору даних, логування (LogFile), мережеві налаштування ListenPort, StartPollers, StartTrappers тощо. У процесі роботи сервер підтримує підключення як до Zabbix Agent (через TCP-порт 10050), так і до інших джерел даних – наприклад, SNMP-пристроїв, IPMI-контролерів, зовнішніх скриптів або REST API. Він безперервно записує отриману інформацію в базу даних у цьому випадку – MariaDB, де формується історія метрик, лог подій, сповіщення, інформація про стан хостів.

База даних (MariaDB) – використовується як централізоване сховище для всіх отриманих метрик, подій та конфігурацій. Структура бази включає таблиці:

- history_uint, history_float, trends – зберігання історичних значень;
- items, hosts, triggers, events – метаінформація про об'єкти моніторингу;
- alerts, actions, media – опис способів сповіщення і їх прив'язок до користувачів.

Zabbix Frontend – це веб-застосунок, реалізований мовою PHP і розгорнутий на базі веб-сервера Apache2. Він виступає в ролі графічного інтерфейсу користувача до всієї системи моніторингу та забезпечує повноцінне адміністрування Zabbix через браузер. За замовчуванням веб-інтерфейс доступний на стандартному HTTP-порті 80, або на HTTPS-порті 443 – якщо ввімкнене шифрування за допомогою SSL. Файли інтерфейсу розміщуються в каталозі /usr/share/zabbix, а параметри доступу та конфігурації зберігаються у відповідному Apache віртуальному хості, наприклад, /etc/apache2/sites-available/zabbix.conf.

Frontend також взаємодіє з базою даних через PHP-розширення mysqli, і для його повноцінної роботи необхідні встановлені модулі php-mbstring, php-gd, php-xml, php-bcmath, php-ldap тощо.

Функціональність інтерфейсу охоплює всі ключові аспекти моніторингу: додавання хостів, створення шаблонів, налаштування елементів даних (Items), формування тригерів, перегляд дашбордів, логів подій, перегляд історії метрик у вигляді графіків. Через веб-інтерфейс адміністратор також керує правами доступу користувачів, конфігурує канали сповіщення (електронна пошта, Telegram, SMS) та створює сценарії дій у відповідь на події (Actions). Таким чином, Zabbix Frontend є невід'ємною частиною системи, що надає зручну візуалізацію і централізований контроль у режимі реального часу.

Веб-інтерфейс Zabbix Frontend є центральним компонентом взаємодії адміністратора з системою моніторингу. Він реалізований як PHP-додаток, що працює поверх веб-сервера (Apache або Nginx) та звертається до бази даних (MariaDB) для отримання актуальної інформації. Через інтерфейс здійснюється перегляд статусу всіх активних хостів у режимі реального часу з оновленням даних без необхідності перезавантаження сторінки. Відображаються основні метрики: доступність агента, навантаження на CPU, використання пам'яті, мережевий трафік тощо.

Функціонал інтерфейсу дозволяє адміністратору створювати та керувати ключовими об'єктами моніторингу: Items (параметри збору даних, як-от системні лічильники, порти, статуси служб), Triggers (логічні умови, які визначають аварійні або попереджувальні ситуації), Templates (набори елементів і тригерів для типових систем, що дозволяють масштабувати налаштування). Крім того, доступна побудова графіків на основі зібраних даних, створення дашбордів з агрегованими віджетами, а також формування топологічної карти мережі з динамічними статусами хостів.

Забезпечено детальне конфігурування notification rules: на основі спрацьовування тригерів система може запускати сценарії оповіщення, які

передають повідомлення через Email, Telegram, Slack або інші канали. Також можлива прив’язка до дій (Actions), які автоматизують реакцію - наприклад, перезапуск сервісу через SSH або виклик зовнішнього API. Всі ці функції дозволяють підтримувати стабільність інфраструктури, забезпечуючи оперативне втручання в разі відхилень від норми.

Взаємодія між компонентами здійснюється через внутрішні протоколи Zabbix TCP (порт 10050 для агентів, 10051 для сервера) та HTTP/HTTPS для frontend.

3.2 Налаштування та запуск серверної частини моніторингу

Вибір Zabbix як платформи зумовлений її широкими можливостями у сфері моніторингу мереж, гнучкістю налаштувань, а також підтримкою великої кількості протоколів збору даних. Система була розгорнута у віртуальному середовищі VirtualBox на базі операційної системи Ubuntu Server 24.04. Для цього створюється базова віртуальна машина у середовищі VirtualBox із виділеними ресурсами, які дозволять підтримувати роботу бази даних, веб-інтерфейсу та основного серверного демона Zabbix.

Перед встановленням Zabbix необхідно підготувати базову віртуальну машину з такими параметрами:

Конфігурація віртуальної машини включає: два ядра процесора (vCPU), які забезпечують паралельну обробку запитів агента і фронтенду; 4 ГБ оперативної пам’яті для стабільної роботи служб бази даних (MariaDB), веб-сервера (Apache) і самого Zabbix Server; жорсткий диск об'ємом не менше 25 ГБ - цього вистачить для зберігання історичних даних моніторингу, логів та системних файлів; один мережевий адаптер, налаштований у режимі внутрішньої мережі (Internal Network), для взаємодії з іншими віртуальними вузлами без виходу в зовнішню мережу. Така підготовка гарантує стабільність при подальшому розгортанні системи моніторингу та дає змогу масштабувати рішення у майбутньому.

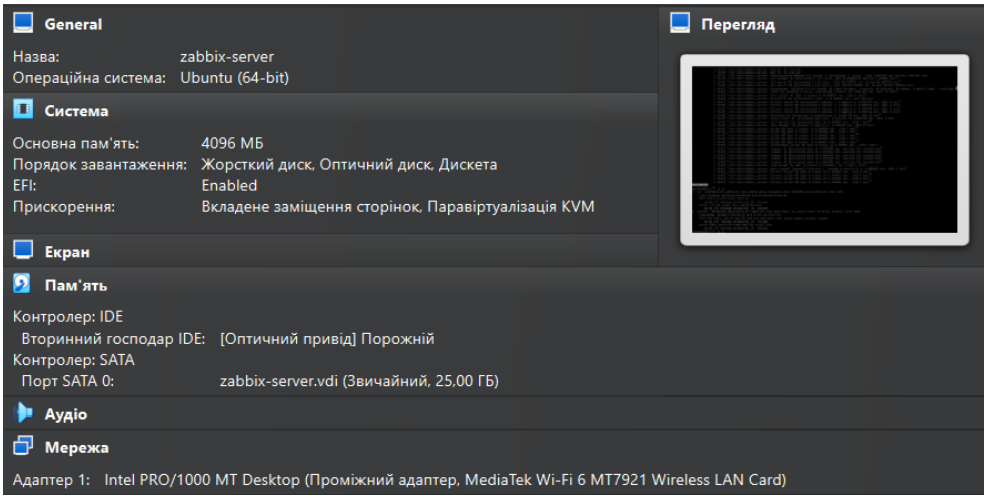


Рисунок 3.3 – Відображення параметрів сервера Zabbix після налаштування конфігурації.

На даній ВМ встановлено Ubuntu Server 24.04 з оновленням усіх базових пакетів за допомогою команд. Далі встановлюємо залежності для роботи Zabbix, зокрема веб-сервер Apache2, PHP, MariaDB та інші необхідні компоненти.

```
sudo apt update
sudo apt upgrade -y
```

Рисунок 3.4 – Встановлення та оновлення наявних пакетів.

Оскільки система Zabbix підтримує автоматизовану інсталяцію за допомогою офіційного скрипта, який значно спрощує процес розгортання всіх необхідних компонентів (включно з базою даних, веб-інтерфейсом, сервером моніторингу та агентом), було прийнято доцільне рішення скористатися саме цим методом. Завдяки використанню офіційного скрипта, всі компоненти інтегруються автоматично, гарантуючи сумісність версій і відповідність документації проєкту.

```
wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.4-1+ubuntu24.04_all.deb
sudo dpkg -i zabbix-release_6.4-1+ubuntu24.04_all.deb
sudo apt update
```

Далі Zabbix використовує реляційну базу даних як ключовий компонент для зберігання усієї інформації про конфігурацію системи (хости, шаблони, елементи даних, тригери), історичні значення метрик, журнали подій та результати моніторингу. У межах реалізації було створено окрему базу даних zabbix, а також користувача з відповідними правами доступу (zabbix@localhost). Процедура включала створення користувача через MySQL та надання йому повного доступу до новоствореної БД через команду GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost';. Після цього була імпортована попередньо згенерована SQL-схема з архіву server.sql.gz, яка містить усі необхідні таблиці для функціонування системи моніторингу.

```
CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
CREATE USER 'zabbix'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Рисунок 3.5 – Налаштування користувача та бази для Zabbix.

Налаштування конфігурації Zabbix Server є одним із ключових етапів у процесі розгортання системи моніторингу, оскільки саме тут задаються параметри взаємодії між сервером моніторингу, базою даних та іншими компонентами інфраструктури. Файл конфігурації `zabbix_server.conf` розташовується за шляхом `/etc/zabbix/zabbix_server.conf` у файловій системі операційної системи Ubuntu Server. Саме в цьому файлі необхідно вказати ім’я бази даних, ім’я користувача та його пароль, які були створені раніше на етапі ініціалізації MariaDB.

Ці значення вказують серверу Zabbix, до якої саме бази даних він має підключатися, яким користувачем здійснювати авторизацію та з яким паролем. Неправильне вказання цих параметрів призведе до неможливості запуску служби `zabbix-server`, оскільки вона не зможе отримати доступ до бази даних для читання та запису конфігураційних даних, збереження історії моніторингу та роботи з

подіями. Після внесення змін у конфігураційний файл, необхідно перезапустити відповідні служби системи.

Після цього можна переходити до налаштування веб-інтерфейсу Zabbix (Zabbix Frontend). Він працює як PHP-застосунок, який обробляється Apache2 і доступний через браузер за IP-адресою сервера. Якщо мережева конфігурація налаштована правильно (наприклад, використовується NAT або Bridge у VirtualBox), то відкривши браузер на хост-машині, можна перейти за адресою де 192.168.1.10X – це IP-адреса віртуальної машини з Ubuntu Server, отримана через DHCP або вручну призначена.

На цій сторінці починається первинна конфігурація через веб-інтерфейс. Користувач проходить через кілька кроків: Цей процес виконується у браузері після переходу за IP-адресою сервера, де доступний інтерфейс Zabbix Frontend. Інтерфейс ініціює покроковий майстер налаштування (installation wizard), який виконує первинну перевірку середовища та проводить користувача через базові кроки конфігурації.

На першому кроці необхідно вказати параметри для підключення до бази даних: ім’я бази (zabbix), логін (zabbix) та пароль, який було заздалегідь задано у конфігураційному файлі zabbix_server.conf. Після цього користувач визначає ім’я інсталяції системи – це логічне найменування, яке надалі відображатиметься у верхній частині веб-інтерфейсу Zabbix і допоможе ідентифікувати сервер у випадку багатосерверної інфраструктури.

Фінальний етап первинного налаштування полягає у валідації всіх попередніх дій. Система перевіряє доступність бази даних, наявність необхідних PHP-модулів, правильність параметрів середовища (зокрема, таймаутів, часової зони, доступу до файлів), і лише після успішного проходження всіх перевірок завершує процес інсталяції. У випадку помилок користувач отримає детальну інформацію про необхідні виправлення для продовження конфігурації.

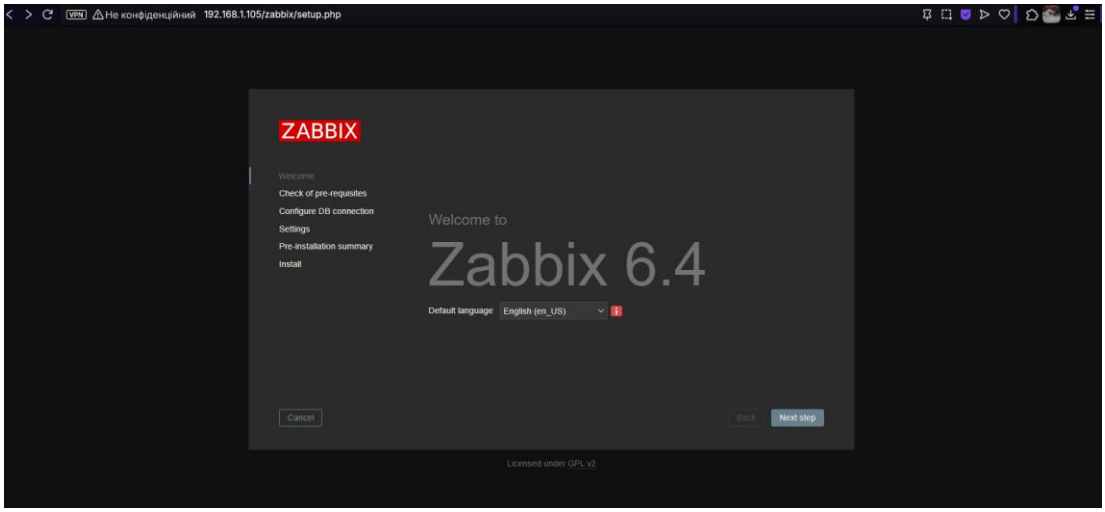


Рисунок 3.6 – Вікно конфігурації для Zabbix.

Після завершення цього процесу користувач автоматично потрапляє на форму входу до Zabbix Frontend. Таким чином, веб-інтерфейс стає повністю доступним для подальшого налаштування шаблонів, додавання хостів, створення тригерів, налаштування Telegram-сповіщень тощо.

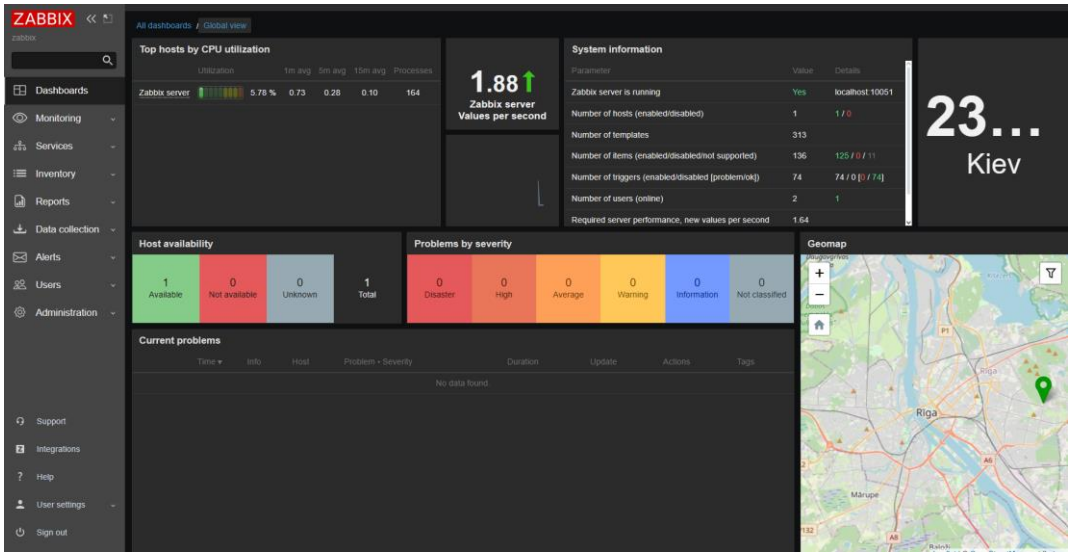


Рисунок 3.7 – Головний інтерфейс Zabbix сервер.

Далі нам потрібний Zabbix Agent щоб коректно відображати стан мережі. Для встановлення використовується стандартний менеджер пакунків APT, що дозволяє швидко та безпечно інсталювати необхідне програмне забезпечення. Команда sudo

apt install zabbix-agent -y ініціює процес завантаження та встановлення пакету zabbix-agent із репозиторіїв Ubuntu. Ключ -y автоматично погоджується з усіма запитами системи, що дозволяє уникнути ручного підтвердження дій.

Після успішної інсталяції агента, обов'язковим є редагування конфігураційного файлу, який знаходиться за шляхом /etc/zabix/zabbix_agentd.conf. Саме в цьому файлі визначаються основні параметри взаємодії агента з сервером моніторингу. Параметр Server=192.168.1.103 вказує IP-адресу сервера Zabbix, з якого дозволено здійснювати пасивний моніторинг (тобто опитування). Аналогічно, параметр ServerActive=192.168.1.103 задає IP-адресу сервера для активного режиму моніторингу - коли сам агент ініціює відправку даних на сервер. Це дозволяє гнучко налаштувати метод взаємодії залежно від топології мережі та політик безпеки. Параметр Hostname=TestHost визначає ім'я, під яким цей хост буде ідентифікуватися в системі моніторингу. Це ім'я має точно збігатися з тим, що буде задано при додаванні хосту у веб-інтерфейсі Zabbix, інакше виникне конфлікт або система не зможе правильно співставити отримані дані з відповідним об'єктом.

Після внесення змін у файл конфігурації потрібно перезапустити службу агента командою sudo systemctl restart zabbix-agent, що дозволяє застосувати нові параметри без потреби перезавантаження всієї системи. Також виконується команда sudo systemctl enable zabbix-agent, яка додає службу до автозавантаження, тобто забезпечує її автоматичний старт після кожного перезавантаження машини. Це критично для забезпечення постійного моніторингу, особливо у продуктивних або тестових середовищах.

Додавання нового вузла (хосту) у систему моніторингу відбувається через веб-інтерфейс Zabbix. Для цього необхідно перейти до розділу Configuration та Hosts та створити новий запис. У відповідному полі задається ім'я хосту яке має збігатися з тим, що вказане у zabbix_agentd.conf, а також IP-адреса клієнтської машини наприклад, 192.168.1.105. Для того, щоб сервер міг почати збирати інформацію з хосту, до нього обов'язково прив'язується шаблон моніторингу. У

випадку Linux-хостів найчастіше використовується шаблон Template OS Linux by Zabbix agent, який уже містить велику кількість попередньо налаштованих елементів даних (CPU usage, RAM usage, disk space, uptime, ping та інші), а також відповідні тригери та графіки.

Після збереження конфігурації хосту система починає збирати метрики. Для перевірки коректності взаємодії між сервером та агентом необхідно перевірити статуси на веб-інтерфейсі - активність хосту, отримані дані, активні тригери. Додатково до цього виконується перегляд логів, що дозволяє отримати технічну інформацію про можливі помилки або проблеми у взаємодії компонентів. Логи агента зберігаються у файлі /var/log/zabbix/zabbix_agentd.log, а серверні логи - у /var/log/zabbix/zabbix_server.log.

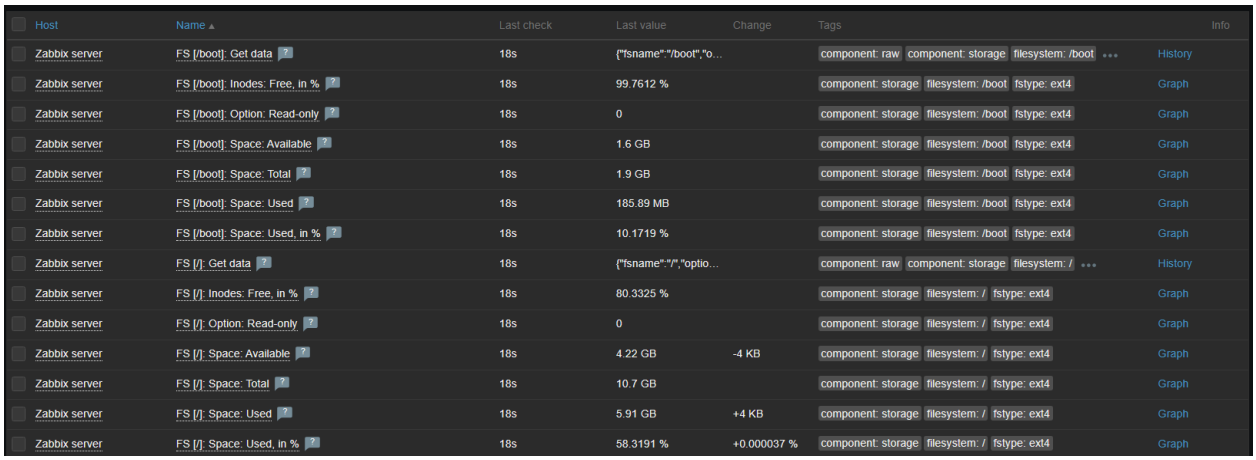
Це дозволяє оперативно реагувати на помилки з'єднання, помилки автентифікації, невірні значення хостнеймів або некоректну конфігурацію шаблонів. Таким чином, налаштування агента - це ключовий крок для забезпечення ефективного моніторингу в системі, що базується на Zabbix.

3.3 Результати роботи програмного забезпечення кіберфізичної системи моніторингу

Після завершення етапів інсталяції, налаштування та конфігурації системи, що базується на платформі Zabbix, було отримано стабільно функціональне середовище для моніторингу оптичної мережі у віртуальному просторі. Програмна реалізація успішно продемонструвала працездатність усіх ключових компонентів: Zabbix Server, бази даних, агента, веб-інтерфейсу та системи сповіщення через Telegram-бота. Система повністю симулює логіку реальної інфраструктури, дозволяючи не лише відслідковувати ключові метрики, але й оперативно реагувати на критичні події.

Одразу після налаштування Zabbix Agent на клієнтській віртуальній машині з Linux, з'явилась можливість зчитування поточних показників операційної системи. У розділі Monitoring – Latest data були відображені дані по центральному

процесору (CPU), оперативній пам’яті (RAM), файловій системі, аптайму, ICMP-доступності та інших критичних параметрах.



Host	Name ▲	Last check	Last value	Change	Tags	Info
Zabbix server	FS [/boot]: Get data	18s	{"fsname":"/boot","o...		component: raw component: storage filesystem: /boot •••	History
Zabbix server	FS [/boot]: Inodes: Free, in %	18s	99.7612 %		component: storage filesystem: /boot fstype: ext4	Graph
Zabbix server	FS [/boot]: Option: Read-only	18s	0		component: storage filesystem: /boot fstype: ext4	Graph
Zabbix server	FS [/boot]: Space: Available	18s	1.6 GB		component: storage filesystem: /boot fstype: ext4	Graph
Zabbix server	FS [/boot]: Space: Total	18s	1.9 GB		component: storage filesystem: /boot fstype: ext4	Graph
Zabbix server	FS [/boot]: Space: Used	18s	185.89 MB		component: storage filesystem: /boot fstype: ext4	Graph
Zabbix server	FS [/boot]: Space: Used, in %	18s	10.1719 %		component: storage filesystem: /boot fstype: ext4	Graph
Zabbix server	FS [/]: Get data	18s	{"fsname":"/","optio...		component: raw component: storage filesystem: / •••	History
Zabbix server	FS [/]: Inodes: Free, in %	18s	80.3325 %		component: storage filesystem: / fstype: ext4	Graph
Zabbix server	FS [/]: Option: Read-only	18s	0		component: storage filesystem: / fstype: ext4	Graph
Zabbix server	FS [/]: Space: Available	18s	4.22 GB	-4 KB	component: storage filesystem: / fstype: ext4	Graph
Zabbix server	FS [/]: Space: Total	18s	10.7 GB		component: storage filesystem: / fstype: ext4	Graph
Zabbix server	FS [/]: Space: Used	18s	5.91 GB	+4 KB	component: storage filesystem: / fstype: ext4	Graph
Zabbix server	FS [/]: Space: Used, in %	18s	58.3191 %	+0.000037 %	component: storage filesystem: / fstype: ext4	Graph

Рисунок 3.8 – Дані, що надходять у Zabbix від агента клієнтського вузла

Коли значення певних метрик, що зчитуються агентом з клієнтського вузла, перевищують заздалегідь встановлені порогові значення, система автоматично активує тригер. Ці порогові значення визначаються в шаблоні моніторингу та можуть охоплювати широкий спектр показників - від перевищення навантаження на процесор понад 75%, недостатньої кількості доступної оперативної пам’яті, до повної втрати зв’язку з вузлом. Як тільки умова тригера виконується, у системі створюється нова подія, яка миттєво фіксується в модулі Monitoring – Problems. Там подія відображається у вигляді таблиці, де наведено детальний опис проблеми, ім’я хоста, рівень критичності (Warning, High, Disaster), а також час її виникнення. Оскільки оновлення відбувається в режимі реального часу, адміністратор отримує змогу швидко виявити аномалію та перейти до аналізу причин її виникнення.

Кожна подія у списку проблем має індикатор статусу, який може змінюватися з плином часу – наприклад, якщо проблема автоматично усунута (при поверненні параметра в норму), вона переміщується у статус “resolved” та зникає з активного списку. Такий підхід дозволяє зберігати історію подій, оцінювати стабільність роботи вузлів, а також аналізувати тенденції відмов. У разі інтеграції з Telegram-ботом, інформація про критичні події надходить ще й у вигляді миттєвих повідомлень у месенджер, що значно скорочує час реагування. Зрештою,

система моніторингу на базі Zabbix забезпечує надійний механізм виявлення проблем, що є ключовим для підтримання стабільної та безперебійної роботи оптичної мережі.

Для оперативного реагування на події у системі було реалізовано Telegram-бота, який виконує роль каналу нотифікацій. За допомогою BotFather у Telegram було створено нового бота, отримано унікальний токен API, а також визначено ID користувача-одержувача. У каталозі /usr/lib/zabbix/alertscripts/ було розміщено Python-скрипт для надсилання повідомлень через Telegram API. У налаштуваннях Zabbix цей скрипт додано як новий тип медіа - Media type > Script. Коли спрацьовує тригер, система автоматично викликає цей скрипт і надсилає повідомлення адміністратору у Telegram. Приклад повідомлення виглядає наступним чином:

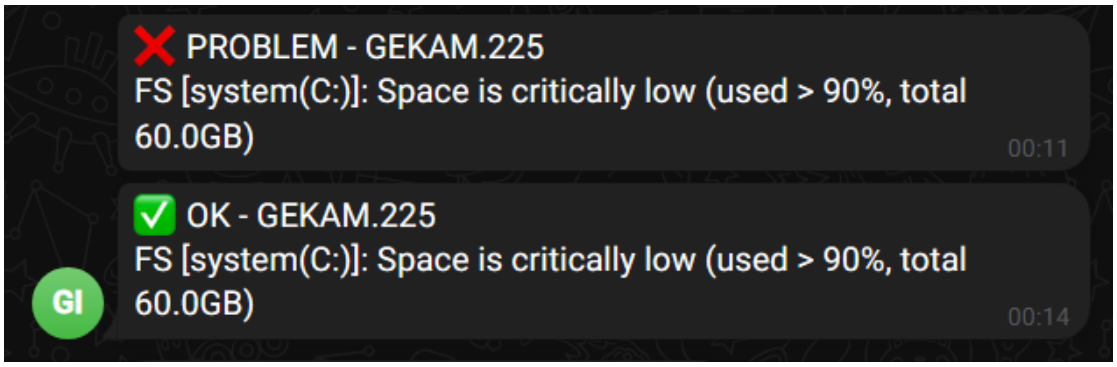


Рисунок 3.9 – Сповіщення у Telegram від Zabbix про критичну подію

Крім моніторингу подій у реальному часі, система Zabbix надає потужний інструментарій для аналітики історичних даних у вигляді інтерактивних графіків. Кожен параметр, який збирається агентом і зберігається в базі даних MariaDB, може бути представлений графічно. Це дозволяє адміністраторам не лише оперативно реагувати на інциденти, а й виконувати ретроспективний аналіз продуктивності вузлів. Наприклад, для метрики system.cpu.util, що відображає рівень використання центрального процесора, можна побудувати графік, який покаже зміну навантаження в реальному часі або за обраний історичний період - годину, день, тиждень, місяць чи навіть рік. Це особливо корисно для виявлення періодичних стрибків навантаження або збоїв, які важко відстежити у момент

їхнього виникнення.

Всі графіки доступні у розділі Monitoring – Graphs, де користувач може вибрати хост, потрібний параметр і період спостереження. Окрім стандартного перегляду графіків, Zabbix також дозволяє створювати дашборди – інтерактивні панелі з великою кількістю віджетів, які відображають узагальнену інформацію з кількох вузлів або груп хостів. Ці дашборди створюються через розділ Monitoring – Dashboards і дозволяють персоналізувати вигляд, наприклад, створивши окрему панель для моніторингу серверів, іншу – для мережевих пристроїв, і ще одну – для критичних сервісів. Таким чином, адміністратор отримує повну картину стану.

Система Zabbix, яка була впроваджена у рамках симульованого середовища, також надає зручні інструменти для аудиту та аналізу змін у системі**.** Вкладка Reports - Audit дозволяє адміністратору переглядати всі дії, що виконувалися користувачами, зокрема додавання чи редагування хостів, налаштування тригерів, зміну користувачів, шаблонів тощо. Також у Monitoring - Event log доступний журнал подій, де відображається вся хронологія змін станів у системі - спрацювання тригерів, надходження нових метрик, втрати зв’язку, зупинка або запуск агентів. Завдяки цьому адміністратор має змогу контролювати цілісність конфігурації**,** виявляти потенційні порушення, а також аналізувати дії користувачів при налагодженні або розслідуванні інцидентів.

У підсумку, реалізована кіберфізична система моніторингу оптичної мережі продемонструвала високу ефективність та адаптивність до віртуального середовища**.** Вона охоплює ключові функціональні компоненти: агентський моніторинг, централізоване управління, журналювання, збереження історичних даних, візуалізацію метрик та оперативне оповіщення через Telegram. Усі ці елементи дозволяють отримувати актуальну інформацію про стан мережевої інфраструктури**,** швидко ідентифікувати потенційні загрози та своєчасно реагувати на критичні ситуації. Гнучкість налаштувань, підтримка кастомних шаблонів та скриптів, а також зручний інтерфейс роблять систему придатною не лише для

навчальних, а й для практичних завдань у сфері адміністрування телекомунікаційних мереж.

3.4 Інтеграція системи оповіщення через Telegram

Веб-інтерфейс Zabbix є одним з ключових компонентів кіберфізичної системи моніторингу оптичних мереж, що надає користувачу можливість у реальному часі контролювати всі процеси, налаштовувати параметри, аналізувати історичні дані та переглядати поточні стани вузлів системи. Інтерфейс реалізований як PHP-застосунок, що функціонує у середовищі Apache2 і є доступним через веб-браузер за IP-адресою сервера. У межах реалізованого рішення доступ здійснюється за адресою http://192.168.1.105/zabbix.

Однією з головних функцій веб-інтерфейсу є доступ до актуальної інформації про стан хостів, події, графіки, тригери та журнали. Інтерфейс дозволяє створювати власні дашборди**,** на яких зручно відображаються критичні показники, зокрема навантаження CPU, об’єм доступної пам’яті, використання дискового простору, аптайм системи, стан мережевих інтерфейсів тощо.

Після додавання хостів до системи (у нашому випадку - вузол з IP-адресою 192.168.1.105 та іменем TestHost), вже за кілька хвилин у вкладці Monitoring - Latest Data можна спостерігати надходження телеметричних даних у реальному часі. Кожен параметр супроводжується міткою часу останнього оновлення, що дозволяє оцінити регулярність передачі метрик та своєчасність реакції системи.

У розділі Monitoring - Problems адміністратор має змогу переглядати список поточних проблем – подій, які були зафіксовані як порушення нормальної роботи хоста. Наприклад, перевищення навантаження на процесор вище 75%, втрата пінгу, зупинка служби або перевищення заповненості файлової системи. Кожна така подія супроводжується рівнем серйозності (Information, Warning, High), міткою часу та описом умови спрацювання тригера.

У вкладці Monitoring - Graphs можна переглядати історію змін кожного з параметрів у вигляді графіків. Наприклад, графік навантаження CPU

(system.cpu.util) дозволяє візуально оцінити динаміку використання ресурсів за останню годину, день, тиждень або будь-який довільно обраний період. Також є можливість експортувати графіки у формат PNG або PDF для подальшої обробки чи включення до звітної документації.

Крім цього, система надає функціонал Dashboards, який дозволяє формувати налаштовувані панелі з виджетами: діаграмами, лічильниками подій, віджетами доступності хостів, топами за використанням ресурсів. Це суттєво підвищує зручність контролю за великою кількістю вузлів або критичних метрик у режимі реального часу.

Усі події, включно з попередженнями, фіксуються у журналі подій (Event log), де їх можна сортувати за типом, рівнем важливості, часом виникнення. Це дозволяє здійснювати ретроспективний аналіз ситуацій, виявляти шаблони або тренди у роботі мережі.

Таким чином, веб-інтерфейс Zabbix у реалізованій системі забезпечує не лише повний контроль за всіма компонентами мережі, а й створює зручне середовище для прийняття рішень, своєчасного реагування та ведення журналу подій. Це відповідає основним принципам побудови ефективної кіберфізичної системи: інтегрованість, прозорість, адаптивність та зручність у використанні.

3.5 Результати візуалізації та роботи веб-інтерфейсу Zabbix

Інтеграція зручного та оперативного механізму нотифікацій є важливою складовою частиною сучасної кіберфізичної системи моніторингу. В межах реалізованого проєкту для організації повідомлень було обрано месенджер Telegram, який є одним із найпопулярніших, кросплатформених засобів обміну повідомленнями. Telegram підтримує доступ до офіційного API, що дозволяє створювати ботів, здатних приймати інформацію у режимі реального часу. Такий підхід забезпечує оперативне інформування адміністратора про всі критичні події, які відбуваються у системі, незалежно від його місця розташування.

Telegram було обрано через низку переваг, серед яких - висока швидкість

доставки повідомлень, підтримка розмітки тексту (HTML або Markdown), гнучка система взаємодії як з приватними повідомленнями, так і з груповими чатами або каналами, а також повна підтримка різних операційних систем. Крім цього, Telegram відзначається простотою інтеграції, зокрема можливістю використання shell- або Python-скриптів для взаємодії з API.

Реалізація починалася зі створення Telegram-бота за допомогою сервісного бота @BotFather. У процесі конфігурації було задано назву, унікальний логін та згенеровано API-токен, що використовується для ідентифікації системи при відправленні повідомлень. Цей токен, який має вигляд комбінації цифр і символів, використовується для виконання HTTPS-запитів до Telegram Bot API, які надсилаються на відповідні URL-адреси.

Для реалізації механізму надсилання повідомлень був написаний Bash-скрипт, що приймає текст повідомлення як аргумент і передає його до Telegram. Скрипт, який було названо telegram_alert.sh, реалізує POST-запит за допомогою утиліти curl. У тілі запиту міститься токен бота, ідентифікатор користувача або групи (Chat ID), а також вміст повідомлення. Сам скрипт розміщується у стандартному каталозі Zabbix для зовнішніх алертів /usr/lib/zabbix/alertscripts/, а його права доступу налаштовуються на виконувані.

Після створення скрипта у веб-інтерфейсі Zabbix виконується налаштування типу медіа. У відповідному розділі створюється новий тип з назвою telegram_alert, типом "Script" та вказанням імені скрипта. Після цього у конфігурації облікового запису адміністратора до переліку методів сповіщення додається Telegram, що дозволяє прив'язати створений канал оповіщення до конкретного користувача.

Заключним етапом є створення правила дії у розділі Actions. Умова для виконання базується на рівні серйозності тригера – наприклад, попередження або вище. У разі настання відповідної події система генерує повідомлення, яке передається через Telegram на основі шаблону, що включає назву хоста, назву тригера та актуальне значення показника, який викликав подію.

Для перевірки працездатності системи проводилося тестування – наприклад,

шляхом зупинення служби агента або штучного навантаження на процесор, що викликало спрацювання тригера. У результаті повідомлення було успішно доставлено до Telegram, що підтвердило коректну інтеграцію і дозволило оцінити практичну користь реалізованої функції.

Таким чином, реалізована система оповіщення через Telegram значно розширює функціональність рішення, забезпечуючи асинхронне інформування у реальному часі. Це підвищує ефективність реагування на інциденти, зменшує залежність від присутності адміністратора за комп’ютером і робить систему моніторингу справді адаптивною та сучасною.

ВИСНОВОК

У цьому розділі було детально розглянуто процес реалізації кіберфізичної системи моніторингу оптичних мереж із використанням програмного забезпечення Zabbix та інструментів для візуалізації, керування та сповіщення. Основна увага була приділена практичним аспектам: налаштуванню серверного середовища, встановленню компонентів системи, конфігурації агентів і підключенню хостів до єдиної інфраструктури.

На практиці було показано, як за допомогою сучасних засобів віртуалізації (зокрема VirtualBox) можливо створити повноцінне середовище для симуляції роботи мережі без залучення фізичного обладнання. Це дозволяє імітувати архітектуру реального телекомунікаційного середовища, протестувати логіку системи та перевірити функціональність збору й обробки даних.

Ключовим досягненням стала інтеграція Telegram як засобу оповіщення про критичні події. Було продемонстровано, як за допомогою API Telegram можна налаштувати автоматичне надсилання повідомлень безпосередньо з системи моніторингу, забезпечивши тим самим швидкий зворотній зв’язок між інфраструктурою та адміністратором. Завдяки цьому система отримала здатність миттєво повідомляти про відмови, перевантаження або інші критичні події, що значно підвищує рівень надійності та знижує час реагування на інциденти.

Також система продемонструвала свою масштабованість, адаптивність та гнучкість. Можливість створення шаблонів, додавання нових хостів, використання тригерів, графіків і дашбордів відкриває шлях до подальшого розвитку рішення. Усі основні функції – збір, обробка, візуалізація, історичне збереження та сповіщення – було успішно реалізовано.

Загалом реалізована система може використовуватись як у навчальних цілях, так і як базовий прототип для впровадження в реальних умовах. Вона ілюструє ефективний підхід до побудови сучасної кіберфізичної інфраструктури моніторингу з відкритим програмним забезпеченням та високим ступенем автоматизації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Muhammad N. Integrated Network Monitoring using Zabbix with Push Notification via Telegram. *Journal of Computing Research and Innovation*. URL: https://journal.uthm.edu.my/ojs/index.php/jcri/article/view/282 (дата звернення 04.03.2025).

2. Cho J. Y., Pedreno-Manresa J.-J., Patri S. K. et al. DeepALM: Holistic Optical Network Monitoring based on Machine Learning. URL: https://arxiv.org/abs/2205.12236 (дата звернення 04.03.2025).

3. Abdelli K., Griesser H., Ehrle P. et al. Reflective Fiber Faults Detection and Characterization Using Long-Short-Term Memory. URL: https://arxiv.org/abs/2203.08907 (дата звернення 05.04.2025).

4. Araujo A. Case Study: Monitoring with Zabbix and AI. URL: https://blog.zabbix.com/case-study-monitoring-with-zabbix-and-ai/19329/ (дата звернення 05.04.2025).

5. Kammer M. The Zabbix Advantage for Business. URL: https://blog.zabbix.com/the-zabbix-advantage-for-business/19137/ (дата звернення 08.04.2025).

6. Monitoring Zabbix with Telegram Notifications. URL: https://www.zabbix.com/integrations/telegram (дата звернення 08.04.2025).

7. Pereira C. Monitoring WDM Optical Networks Using Open-Source Tools. *Journal of Optical Communications and Networking*. 2023. Vol. 15. № 4. P. 245–253.

8. GitHub ilevveli/Zabbix Telegram AI Integration with Flask. URL: https://github.com/ilevveli/zabbix-telegram-ai-integration-with-flask (дата звернення 08.04.2025).

9. GitHub GabrielRF/Zabbix Telegram Notification. URL: https://github.com/GabrielRF/Zabbix-Telegram-Notification (дата звернення 08.04.2025).

10. Zhang L., Wei P. AI for Fiber-Optic Anomaly Detection. *Optical Fiber Technology*. 2023. Vol. 75. P. 101002.

11. Molloy C. Real-Time Optical Network Monitoring using Zabbix for iCONNET OLT. *Int'l Journal of Optical Communication Systems*. 2023. P. 45–53.

12. Herrera J. Using SNMP and Zabbix for Optical Equipment Monitoring. Optical Engineering. 2022. Vol. 61. № 6. P. 065101.

13. Monitoring Methodologies. URL: https://www.airquality.gov.wales/about-air-quality/monitoring/monitoring-methodologies (дата звернення 08.06.2025).

14. Gondosubroto R. Internet of Things from Scratch: Build IoT solutions for Industry 4.0 with ESP32, Raspberry Pi, and AWS. Packt Publishing, 2024. 438 p.

15. Nolan A. Unlocking the Power of Raspberry Pi 5: Your Complete Guide from Setup to Expert Projects, 2024. 84 p.

16. Halfacree G., Everard B. Get Started with MicroPython on Raspberry Pi Pico: The Official Raspberry Pi Pico Guide. 2nd Edition. Raspberry Pi Press, August 6, 2024. 208 p.

17. Smyth N. Android Studio Iguana Essentials – Java Edition: Developing Android Apps Using Android Studio 2023.2.1 and Java. Payload Media, March 18, 2024. 1277 p.

18. Kumar A., Hussain J., Chun A. Connecting the Internet of Things: IoT Connectivity Standards and Solutions. Apress, 2023. 390 p.

19. Smeenk H. G. Internet of Things for Smart Buildings: Leverage IoT for smarter insights for buildings in the new and built environments. Packt Publishing, 2023. 306 p.

20. Dian F. J. Fundamentals of Internet of Things: For Students and Professionals. John Wiley & Sons, 2022. 432 p.

21. Ferrer-Cid P. et al. Sampling Trade-Offs in Duty-Cycled Systems for Air Quality Low-Cost Sensors. *ACM Transactions on Sensor Networks (TOSN).* 2022. Pp. 125.

22. Yadav K. et al. Few-shot calibration of low-cost air pollution (PM2.5) sensors using meta-learning. *ACM Transactions on Sensor Networks (TOSN).* 2021. P. 1–30.

23. Bell C. Beginning MicroPython with the Raspberry Pi Pico: Build Electronics and IoT Projects (Maker Innovations Series). Apress, July 24, 2022. 660 p.

24. Grinberg M. MicroPython for the Raspberry Pi Pico W: A gentle introduction to programming digital circuits with Python. October 29, 2022. P. 201.

25. Zabbix Documentation 6.0 LTS. URL: https://www.zabbix.com/documentation/6.0/en/manual (дата звернення 13.05.2025).

26. Installing and configuring Zabbix with PostgreSQL. URL: https://computingforgeeks.com/how-to-install-zabbix-server-on-ubuntu-linux/ (дата звернення 13.05.2025).

27. Building a Fiber Optic Network Monitoring System. URL: https://www.sciencedirect.com/science/article/pii/S0030401820308752 (дата звернення 08.06.2025).

28. Real-Time Monitoring of FTTH Networks Using Zabbix and SNMP. *International Journal of Optical Communications*. 2023. Vol. 17, № 2. P. 87–95.

29. Karaca M., Soyturk M. Anomaly Detection in Optical Transport Networks Using Autoencoders. *Journal of Lightwave Technology*. 2022. Vol. 40, № 12. P. 301.

30. Pahl C., Helmer S. A Self-Adaptive Monitoring Framework for Cloud Infrastructure and Microservices. Future Generation Computer Systems. 2022. Vol. 129. P. 179–192.

31. Monitoring Passive Optical Networks: Challenges and Solutions. IEEE Communications Surveys & Tutorials. 2023. Vol. 25, № 4. P. 20252045.

32. VirtualBox User Manual – Oracle VM VirtualBox. URL: https://www.virtualbox.org/manual/UserManual.html (дата звернення 08.06.2025).

33. Evolution of Network Monitoring Technologies in the Age of IoT. Computer Networks. – 2023. – Vol. 234. – Article 109942.

34. Manisalidis I., Stavropoulou E., Stavropoulos A., Bezirtzoglou E. Environmental and Health Impacts of Air Pollution: A Review. *Front Public Health*. – 2020. – URL: https://www.frontiersin.org/articles/10.3389/fpubh.2020.00014/full (дата звернення 08.06.2025).

35. Veiga T. et al. Blind Calibration of Air Quality Wireless Sensor Networks Using Deep Neural Networks. *ACM Transactions on Sensor Networks (TOSN).* 2021. P. 1–28.

36. Spinelle L. et al. Review of portable and low-cost sensors for the ambient air monitoring of benzene and other volatile organic compounds. *ACM Transactions on Sensor Networks (TOSN).* 2020. P. 1–40.

37. GitHub – ilevveli/Zabbix Telegram AI Integration with Flask. URL: https://github.com/ilevveli/zabbix-telegram-ai-integration-with-flask (дата звернення 01.06.2025).

38. GitHub – GabrielRF/Zabbix Telegram Notification. URL: https://github.com/GabrielRF/Zabbix-Telegram-Notification (дата звернення 01.06.2025).

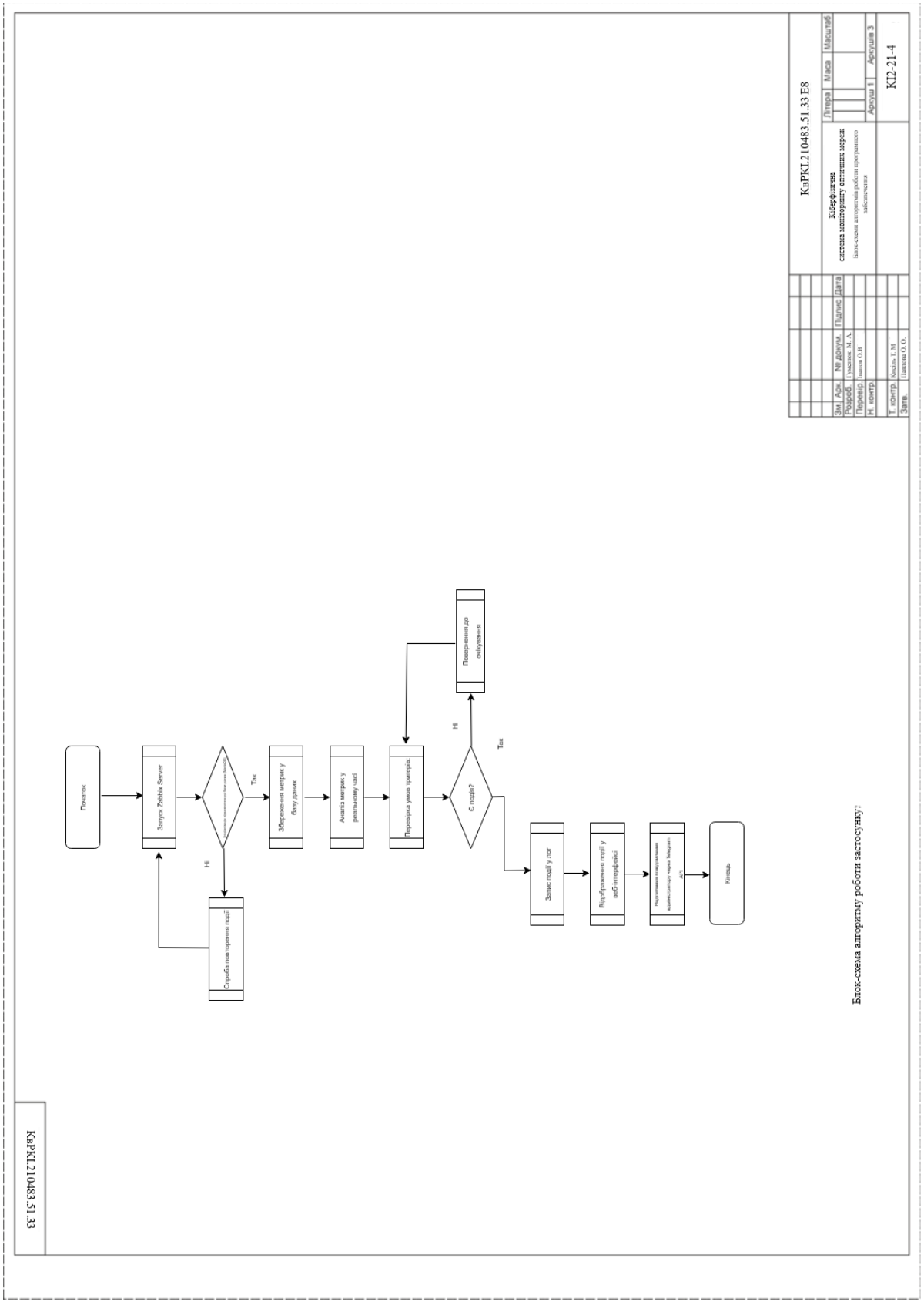
39. Park S., Kim M. Automation and Monitoring Framework for ISP Networks with Zabbix and Ansible. Future Internet, 2022. Vol. 14, № 3. DOI: 10.3390/fi14030077.

40. Fiber Infrastructure Monitoring using Raspberry Pi and Python. *Journal of Network and Computer Applications*. 2023. Vol. 210. Article 103549.

Додаток А

(обов’язковий)

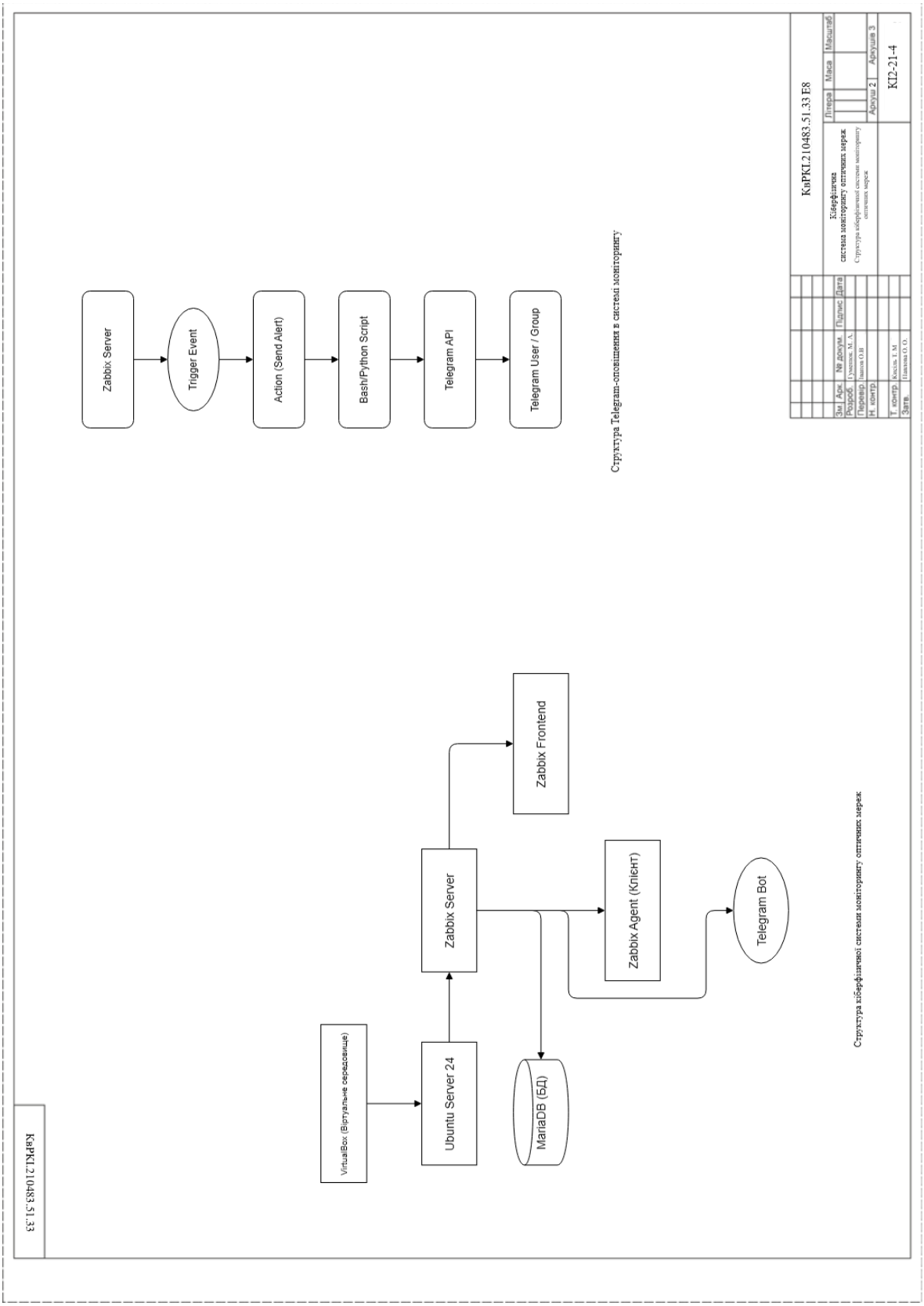
Копія креслення «Блок-схеми алгоритмів роботи програмного забезпечення»



Додаток Б

(обов’язковий)

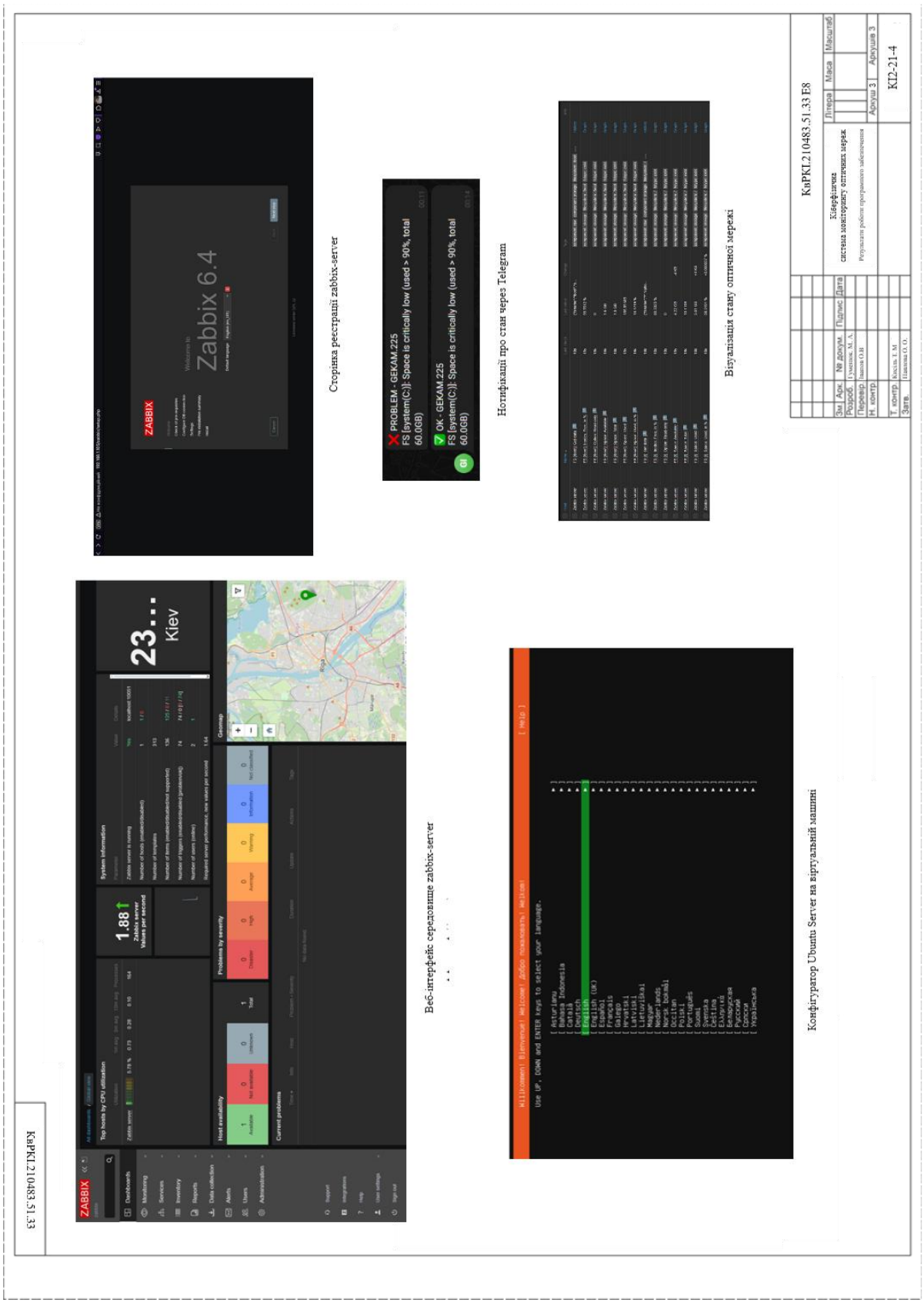
Копія креслення «Структура кіберфізичної системи моніторингу оптичних мереж»



Додаток В

(обов'язковий)

Копія креслення «Результати роботи програмного забезпечення»



Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Максим ГУМЕНЮК

Співавтор:

Назва: Гуменюк_Кіберфізична система моніторингу оптичних мереж

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3%

Коефіцієнт подібності 2:0.2%

Мікропробіли: 39

Заміна букв: 0

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-12 22:18:59.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-13 Доцент Андрій Нічепорук

Дата експерт

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 2.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 13%**

ID: 245508 Title: БКР Кіберфізична система моніторингу оптичних мереж Added in a DB: 2025-06-13 Authors: Максим ГУМЕНЮК Heads: Олексій ІВАНОВ Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	102848	690	2570 (2%)	27 (4%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Гуменюк Максим Андрійович

Тема: Кіберфізична система моніторингу оптичних мереж.

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень ___3____ Кількість сторінок записки ___57___

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є створення кіберфізичної системи моніторингу оптичної телекомунікаційної мережі з використанням серверної платформи, клієнтських агентів та інтеграції Telegram для сповіщень у режимі реального часу.

2. Висновок про відповідність роботи дипломному завданню: Дипломний проект у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині даного проекту.

3. У першому розділі дипломної роботи виконано аналіз предметної області, розкрито поняття кіберфізичних систем, принципи роботи систем моніторингу, а також обґрунтовано актуальність обраної тематики. Особливу увагу приділено порівнянню популярних систем моніторингу та розгляду їхніх переваг і недоліків. У другому розділі обґрунтовано архітектуру системи, визначено функціональні вимоги до програмного забезпечення, проаналізовано обрані програмні, а також наведено покроковий план розгортання системи. У третьому розділі докладно описано реалізацію: встановлення серверного середовища Ubuntu Server, налаштування Zabbix Server і Agent, конфігурація веб-інтерфейсу, імпорт схеми бази даних, створення хостів і шаблонів моніторингу. Також реалізовано функцію оповіщення через Telegram, що значно підвищує оперативність реагування на події. Усі кроки оформлені із залученням скріншотів, блок-схем і логічних діаграм. У роботі використано сучасні open-source технології, що є актуальними в сфері адміністрування та моніторингу телекомунікаційних мереж.

4. Робота має високу практичну значущість: реалізована система є придатною як для навчальних, так і для дослідницьких цілей. Проєкт демонструє можливість створення повноцінного моніторингу в умовах обмеженого доступу до реальної мережевої інфраструктури, використовуючи лише віртуальні ресурси. Включення Telegram-бота забезпечує мобільність та актуальність оповіщень, а використання модульної структури дозволяє масштабувати систему в майбутньому. Вибір сучасного стеку програмного забезпечення свідчить про розуміння технологічних трендів.

5. Негативні сторони роботи: У майбутніх версіях варто передбачити шифрування каналу зв'язку між агентами та сервером, а також реалізацію багатокористувацького доступу з розмежуванням прав..

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науковотехнічному рівні.

8. Інші зауваження: немає

9. Оцінка дипломної роботи: Розглянувши позитивні та негативні сторони представленої дипломної роботи вважаю, що робота заслуговує оцінки «відміно», 4.75 (А).

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Корецька А.О., доцент кафедри АКІТта Р, ХНУ

"16" 06 2025 р. _____(підпис)

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Максим Гуменюк
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-21-4

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

11.06 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Кіберфізична система моніторингу оптичних мереж

Автор: Максим Гуменюу

Спеціальність: 123– Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Олексій ІВАНОВ, к.т.н, доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) Запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, що є загальним оглядом літератури та не стосуються безпосередньо авторського дослідження чи оригінальних результатів роботи.
2) Усі запозичення є фрагментарними та належним чином оформлені відповідно до академічних вимог, з наданням точних посилань на джерела, з яких вони були отримані.
3) Окремі збіги, зафіксовані системою, є загальновживаними фразами або термінами, що часто використовуються в галузі, про що свідчить численні збіги з 10-40 джерелами на один фрагмент тексту. Такі фрази не є результатом авторського плагіату, оскільки належать до загальновизнаних термінів.
4) В окремих випадках система зафіксувала послідовності чотиризначних двійкових кодів, що є типовими вхідними даними для множини задач, і не можуть розглядатися як об'єкт авторських прав, оскільки ці послідовності є стандартними елементами для математичних чи технічних розрахунків.
5) Всі зафіксовані системою ознаки модифікації тексту пов'язані з комбінуванням латинських символів зі скороченнями українських індексів у формулах. Це не є модифікацією змісту, оскільки такі скорочення є стандартною практикою в математичній та технічній літературі і не порушують авторських прав.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 2% і адресується до 401 першоджерела; та системою Anti-Plagiarism складає 1%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Олексій ІВАНОВ

Гарант ОП _____ Андрій НІЧЕПОРУК

Завідувач кафедри КПС _____ Ольга ПАВЛОВА