Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Маленко А.І.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Маленку Андрію Ігоровичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 47-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Маленко А.І.
(прізвище та ініціали)

АНОТАЦІЯ

Маленко А.І. Програмне забезпечення системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

Метою розробки є програмне забезпечення системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

Результат роботи – програмна реалізація системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, моніторинг та сигналізація TCP/IP з'єднань

ABSTRACT

Malenko A.I. Software for monitoring and signaling system of TCP/IP connections based on ESP32 with support for JSON templates. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the monitoring and signaling system of TCP/IP connections based on ESP32 with support for JSON templates.

The purpose of the development is the software for monitoring and signaling system of TCP/IP connections based on ESP32 with support for JSON templates.

The result of the work is the software implementation of the monitoring and signaling system of TCP/IP connections based on ESP32 with support for JSON templates.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A user-friendly interface has been developed. Instructions for working with the software are provided.

The program can be used on PCs with Windows 10/11.

The program is developed in the Python environment.

Keywords: computer engineering, monitoring and signaling of TCP/IP connections

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ЛОМ	–	локальна обчислювальна мережа
ASP		Active Server Pages – активні серверні сторінки
DHCP	–	Dynamic Host Configuration Protocol – протокол динамичної конфігурації вузла
HTTP	–	HyperText Transfer Protocol – протокол передачі гіпер тексту
IMAP	–	Internet Message Access Protocol – протокол доступу до електроної пошти Інтернету
ICMP	–	Internet Control Message Protocol – міжмережний протокол керуючих повідомлень
MMC	–	Microsoft Management Console
POP3	–	Post Office Protocol Version 3 – протокол поштового відділення, версія 3
SQL	–	Structured Query Language – мова структурованих запросів
SMTP	–	Simple Mail Transfer Protocol – простий протокол передачі пошти
SNMP	–	Simple Network Management Protocol – простий протокол керування мережею
Syslog	–	стандарт відправки повідомлень про зміни які відбуваються в мережі
UDP	–	User Datagram Protocol – протокол користувальницьких дейтаграм

ВСТУП

Актуальність теми. З розвитком мережевих технологій технічне обслуговування, керування та моніторинг мережі є важливими для забезпечення безперебійної роботи мережі та підвищення економічної ефективності. У мережі сніффер пакетів використовується, щоб зробити мережу більш безпечною шляхом аналізу мережевих дій, що принесе користь як інженерам мережевого програмного забезпечення, так і адміністраторам мережі. TCP – це протокол зв'язку, який використовується між фізично розділеними комп'ютерними системами. Це надійний протокол транспортування потоку. Він визначає, як встановити з'єднання між двома мережами та підтримувати мережеву розмову. Він відстежує всі вхідні та вихідні пакети, що надходять через канал. Він покаже детальний перелік кінцевих точок TCP системи, включаючи локальну адресу, номер порту, віддалену адресу та стан з'єднання TCP. За результатами аналізу ми можемо визначити, що це за мережа.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

– Дослідження системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

– Програмна реалізація системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Інструменти моніторингу мережі – це технології, які працюють за лаштунками, щоб переконатися, що все функціонує належним чином. Але що це за інструменти і чому вони такі важливі?

У сучасному взаємопов'язаному світі простої мережі можуть призвести до значних фінансових втрат і завдати шкоди вашій репутації. Інструменти моніторингу мережі служать вашою першою лінією захисту від цих проблем, оскільки вони постійно відстежують продуктивність вашої мережі, виявляють аномалії та сповіщають вас про проблеми до їх загострення. Надаючи дані в реальному часі та історичні тенденції, інструменти моніторингу мережі забезпечують проактивне керування, забезпечуючи ефективну та безпечну роботу вашої мережі.

1.2 Область застосування

Експлуатація мережі без засобів моніторингу залишає вас сліпими щодо її продуктивності та стану. Інструменти моніторингу мережі забезпечують необхідну видимість, щоб усе було під контролем. Інструменти моніторингу мережі мають широкий спектр застосувань, оскільки вони можуть контролювати апаратні пристрої, такі як маршрутизатори, комутатори та сервери, а також програмні програми та служби. Збираючи та аналізуючи дані, інструменти та служби моніторингу мережі допомагають вам приймати обґрунтовані рішення щодо оптимізації та розширення мережі.

Наприклад, відстежуючи використання пропускної здатності, ви можете визначити, які програми чи користувачі споживають найбільше ресурсів. Ця інформація дозволяє ефективніше розподіляти смугу пропускання та запобігати

перевантаженням. Крім того, інструменти моніторингу мережі можуть виявляти незвичайну активність і сповіщати про неї, допомагаючи виявити потенційні порушення безпеки.

Розгляд інструмента моніторингу мережі

Багато рішень моніторингу розглядають лише інформацію на поверхневому рівні про умови мережі, як-от моніторинг послуги або пристрою.

Сучасні мережі потребують кількох різних переглядів різних елементів мережі. Приклади:

– Доступність.

– Використання ресурсів.

– Екологічний моніторинг.

– Виявлення та ідентифікація несправностей.

Щоб задовольнити вищезазначені потреби, зазвичай потрібно підтримувати кілька різних методів доступу:

– Моніторинг SNMP.

– Моніторинг сервера/служби/ресурсу WMI.

– Моніторинг шляху/усунення несправностей.

– Моніторинг/аналіз потоку.

– Моніторинг журналу.

Для розширеного розгляду було б, якщо рішення має можливість автоматично збирати вищезазначену інформацію та виконувати аналіз, щоб досягти вирішення першопричини проблем – ніхто не хоче вручну налаштовувати елементи, а потім виконувати ручну інтерпретацію для визначення несправності.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

PathSolutions TotalView

TotalView від PathSolutions виділяється як один із найкращих інструментів моніторингу мережі у 2025 році. Він пропонує комплексні можливості моніторингу та усунення несправностей, що робить його чудовим вибором для підприємств. TotalView надає інформацію про продуктивність мережі в режимі реального часу, визначає проблеми та пропонує дієві рекомендації щодо вирішення.

Завдяки зручному інтерфейсу та надійним функціям звітування TotalView допомагає IT-фахівцям випереджати потенційні проблеми. Його здатність інтегруватися з іншими інструментами та системами робить його універсальним варіантом для компаній будь-якого розміру.

Монітор продуктивності мережі SolarWinds Orion

Монітор продуктивності мережі SolarWinds Orion (NPM) є ще одним настійно рекомендованим інструментом. Відомий своєю масштабованістю та простотою використання, NPM може працювати з мережами будь-якого розміру. Він пропонує розширені функції, такі як відображення топології мережі, інтелектуальне сповіщення та настроювані інформаційні панелі.

Здатність NPM контролювати як локальне, так і хмарне середовище робить його гнучким рішенням для сучасних підприємств. Його потужна підтримка спільноти та обширна документація гарантують, що користувачі можуть отримати максимальну віддачу від інструменту.

Мережевий монітор PRTG

Мережевий монітор PRTG від Paessler – це універсальне та економічно ефективне рішення для моніторингу мережі. Він пропонує широкий спектр датчиків для моніторингу різних аспектів вашої мережі, від використання пропускної здатності до стану обладнання. Інтуїтивно зрозумілий інтерфейс і просте налаштування PRTG роблять його доступним для компаній з різним рівнем технічного досвіду.

Однією з видатних особливостей PRTG є його здатність надавати детальні звіти та візуалізації. Ці відомості допомагають IT-командам зрозуміти продуктивність мережі та приймати рішення на основі даних. PRTG також пропонує гнучкі варіанти ліцензування, що дозволяє компаніям масштабувати свої можливості моніторингу за потреби.

Розглянемо ще список інструментів моніторингу мережі:

– Auvik. Провідне рішення для керування мережею SaaS. Швидке розгортання, засноване на мережевому картографі та спрощує моніторинг мережі як для MSP, так і для внутрішніх IT-команд. Забезпечує централізоване керування системою в одному місці. Почніть 14-денну безкоштовну пробну версію.

– Моніторинг мережі Datadog. Надає чудовий візуальний огляд ваших мережевих компонентів і потоків мережевого трафіку між кожним компонентом. Цей підрозділ системних моніторів із хмарної платформи забезпечує перевірку працездатності пристроїв і аналіз потоків трафіку. Почніть 14-денну безкоштовну пробну версію.

– Paessler PRTG Network Monitor. Безкоштовне програмне забезпечення для моніторингу мережі, яке використовує SNMP, аналіз пакетів і WMI для моніторингу мережі. Доступ до 30-денної безкоштовної пробної версії.

– Domotz. Ця платформа SaaS пропонує ряд послуг моніторингу мережі від опитування SNMP до сканування безпеки. Отримайте доступ до 14-денної безкоштовної пробної версії.

– Checkmk. Служба моніторингу для всієї системи, а не лише для мереж, доступна у безкоштовній і платній версіях. Працює на Linux або фізичному пристрої.

– NinjaOne RMM. Цей пакет на основі RMM надає постачальникам керованих послуг інструменти для догляду за мережами та кінцевими точками. Доставлено з хмари. Почніть 14-денну безкоштовну пробну версію.

– ManageEngine OpManager. Мережевий монітор, який може контролювати пристрої SNMP, комутатори, сервери та віртуалізовані мережеві служби. Отримайте доступ до 30-денної безкоштовної пробної версії.

– Atera. Комплексна агентська платформа штучного інтелекту, яка дає змогу IT-відділам, командам і MSP контролювати, керувати та автоматизувати IT-операції. Розпочніть з безкоштовної пробної версії.

– Obkio. Цей пакет SaaS забезпечує моніторинг мережевих пристроїв і трафіку для локальних мереж і підключень до Інтернету, перевіряючи умови для VoIP та інших інтерактивних протоколів.

– Моніторинг мережі Site24x7. Комбінована IT-інфраструктура, програми та служба моніторингу поведінки користувачів, доступна в хмарі.

– Fortra's Intermapper. Цей простий інструмент починається з інструменту автоматичного виявлення та відображає вашу мережу, а потім пропонує постійний моніторинг продуктивності.

– AdRem NetCrunch. Цей локальний пакет забезпечує моніторинг мережі як частину своїх повних можливостей спостереження. Працює на Windows Server.

– Progress WhatsUp Gold. Ця локальна система керує службою моніторингу пристроїв основної мережі з додатками для відстеження продуктивності інших ресурсів. Працює на Windows Server.

– ExtraHop Reveal(x) Ця служба моніторингу безпеки мережі сканує загрози та підозрілу поведінку в реальному часі. Доступний як пакет SaaS або мережевий пристрій.

– AKIPS Ця система моніторингу мережі забезпечує опитування SNMP і аналіз трафіку з NetFlow, що працює на FreeBSD у VMware або Hyper-V.

– SuperOps Це пакет RMM, який включає служби моніторингу для мереж, серверів, програмного забезпечення та хмарних служб. Пакет розміщено в хмарі.

– Монітор продуктивності мережі SolarWinds Набагато більше, ніж просто сканер. Більшість проблем безпеки мережі виникають, коли конфігурації змінюються, і SolarWinds NPM визначає зміни та може автоматично вирішити багато з них.

– Zabbix Програмне забезпечення для моніторингу мережі з відкритим кодом із моніторингом SNMP та IPMP. Включає систему сповіщень і плагіни спільноти.

– Catchpoint Network Experience Цей модуль є частиною SaaS-пакету інструментів моніторингу повного стеку, які відстежують доставку веб-додатків.

– Nagios Core Один із найкращих інструментів моніторингу мережі з відкритим кодом. Включає панель інструментів, систему сповіщень, плагіни спільноти тощо.

– Icinga Система моніторингу мережі з відкритим кодом і DSL. Включає розширення.

У рамках нашого порівняння я звернув увагу на такі важливі функції платформи керування мережею, як SNMP, відображення мережі, час безвідмовної роботи/простою, сповіщення, моніторинг пропускної здатності, працездатність мережі, інформаційні панелі тощо, щоб виділити інструменти з винятковою зручністю для користувачів.

Ми розбили наш аналіз для вас на основі цих ключових критеріїв:

– Система автоматичного виявлення для реєстрації всіх мережевих пристроїв.

– Програма відображення топології мережі.

– Можливість збирати живі стани мережевих пристроїв за допомогою SNMP.

– Засіб для аналізу продуктивності мережі з часом.

– Графічна інтерпретація даних, таких як діаграми та графіки.

– Безкоштовний пробний період, демонстрація або гарантія повернення грошей для оцінки без ризику.

– Хороша ціна, яка відображає співвідношення ціни та якості порівняно з пропонованими функціями.

Переглянуті мною інструменти містять поєднання безкоштовного, платного та відкритого програмного забезпечення для Windows, Mac і Linux.

Якщо вас цікавить лише безкоштовне програмне забезпечення для моніторингу домашньої мережі, натисніть посилання, щоб переглянути наш список для Windows, Mac і Linux.

1. Auvik

Auvik – це хмарна система моніторингу мережі, яка включає низку інструментів керування системою. Доступ до пакета здійснюється через веб-браузер, і коли ви відкриваєте обліковий запис, процес налаштування встановлює збірники у вашій системі. Пакет Auvik здатний контролювати кілька сайтів і централізує їх контроль. Це робить пакет ідеальним для моніторингу WAN.

Послуга, яку надає Auvik, починається з процесу виявлення мережі. Це автоматично заповнює всю базову інформацію, необхідну для роботи монітора. Служба виявлення безперервна, тому вона виявляє, коли нові пристрої додаються до мережі.

Основні характеристики:

– Автоматичне налаштування: наші висновки показують, що автоматичний процес налаштування Auvik спрощує початкове налаштування та розгортання системи моніторингу.

– Відображення мережі: Платформа пропонує можливості відображення мережі, що дозволяє користувачам візуалізувати макет і підключення в межах своєї мережевої інфраструктури.

– Сповіщення про використання ресурсів: Auvik містить сповіщення про використання ресурсів, що дозволяє проактивно виявляти проблеми, пов'язані з використанням мережевих ресурсів.

– Керування конфігураціями: користувачі можуть ефективно керувати конфігураціями мережевих пристроїв.

Чудовою особливістю Auvik, яку я виявив, є те, що його вищий план забезпечує як аналіз трафіку, так і моніторинг продуктивності мережевих пристроїв. Зі свого хмарного розташування цей пакет може стежити за кількома сайтами, надаючи сповіщення, які виграють час, щоб запобігти системним катастрофам.

Ви можете детально ознайомитися з кожним пристроєм на карті та дослідити його підключення. Оскільки Auvik є хмарною системою, розроблено для віддаленого моніторингу системи – навіть ваша домашня мережа віддалена від процесів на серверах Auvik. Моніторинг мережі працює за системою порогів. Сервіс відстежує список показників мережевої активності, а до кожної з цих умов додається поріг для використання ресурсів або продуктивності системи. Якщо цей пороговий рівень пройдено, служба Auvik створить сповіщення. Це означає, що вашій команді технічних спеціалістів не потрібно звертати увагу на мережу, якщо не виникає проблема.

Службу Auvik можна розширити за допомогою інструментів сторонніх розробників, для яких мережевий монітор має інтеграцію. Є два рівні плану для Auvik – Essential і Performance. План продуктивності є вищим планом і пропонує функції аналізу трафіку та керування системним журналом на додаток до служб моніторингу мережі, доступних у плані Essentials.

Я вважаю, що компанії з кількома сайтами найбільше виграють від послуги Auvik. Цей пакет моніторингу мережі дає змогу централізувати керування системою в одному місці.

Переваги:

– Моніторинг кількох мереж і сайтів: пакет моніторингу Auvik може контролювати кілька мереж на різних сайтах і уніфікувати дані, забезпечуючи централізований перегляд усієї мережевої інфраструктури.

– Доступ з будь-якого місця: Консоль працює через браузер, що забезпечує гнучкість доступу з будь-якого місця з підключенням до Інтернету, розширюючи можливості віддаленого керування.

– Автоматизоване керування інвентаризацією системних активів: Auvik автоматизує створення та підтримку інвентаризації системних активів, спрощуючи відстеження та керування мережевими пристроями.

Недоліки:

– Пропозиція з одним пакетом: ця відсутність різноманітності може обмежити можливості для користувачів, яким потрібні додаткові або спеціалізовані функції, окрім тих, що надає Auvik у своєму основному пакеті.

Auvik не публікує свої ціни. Подорож вашого покупця починається з запиту на 14-денну безкоштовну пробну версію та пошуку ціни.

2. Моніторинг мережі Datadog

Мережеві пристрої позначені кольором для легкого визначення проблемних з'єднань. Datadog Network Performance Monitoring – це хмарна служба моніторингу інфраструктури SaaS, яка перевіряє потоки мережевого трафіку. Його партнером є служба моніторингу мережевих пристроїв, яка зосереджується на статусах кожного пристрою в мережі, наприклад комутаторів, маршрутизаторів і пристроїв.

Основні характеристики:

– Відображення мережі та аналіз протоколів: ця функція дозволяє користувачам візуалізувати структуру мережі та аналізувати протоколи зв'язку в реальному часі.

– Сповіщення про порогові значення продуктивності: платформа містить сповіщення на основі порогових значень продуктивності, які регулюються за

допомогою машинного навчання, забезпечуючи адаптивні та інтелектуальні можливості попередження.

– Кореляція з SNMP: Datadog корелює дані з SNMP та інших джерел, забезпечуючи всебічне уявлення шляхом поєднання інформації з кількох потоків даних.

Datadog пропонує готову інтеграцію із сотнями популярних сервісів і платформ, починаючи від веб-серверів, баз даних, хмарних сервісів і навіть спеціального програмного забезпечення, наприклад Shopify і WordPress, що дозволяє командам відстежувати й аналізувати дані з різних джерел.

Ви можете підписатися на моніторинг пристроїв, моніторинг трафіку або обидва з цим гнучким пакетом. Хмарне розміщення Datadog Network Monitoring дає змогу контролювати будь-яку мережу в будь-якій точці світу з одного операційного центру.

Легко визначити напружені з’єднання, які додають навантаження на мережу. Мережевий монітор продуктивності здатний об'єднати мережевий моніторинг для багатьох сайтів, а також включити хмарні ресурси. Як пакет SaaS послуга включає процесор для запуску програмного забезпечення моніторингу та місце для зберігання зібраної статистики. Окрім відображення поточних статусів, ця служба моніторингу мережі пропонує утиліти захоплення та аналізу пакетів.

Поєднуючи монітор продуктивності мережі та монітор мережевих пристроїв, ви отримуєте повний нагляд за своїми мережами. Система реалізує функцію автоматичного виявлення під час реєстрації, яка ідентифікує всі пристрої, підключені до вашої мережі. Служба створює інвентаризацію IT-активів, а потім генерує з неї карту топології мережі. Процес автовиявлення постійно повторюється, тому будь-які зміни, які ви вносите у свою інфраструктуру, автоматично відображаються в інвентаризації мережі та топологічній карті.

Монітор мережевого пристрою використовує простий протокол керування мережею (SNMP) для запиту комутаторів і маршрутизаторів щодо звітів про стан.

Ці звіти інтерпретуються, надаючи живі дані про стан пристрою на екрані. Ця система також збирає інформацію про кожен пристрій, наприклад марку, модель, вік і потужність процесора.

Кожен мережевий пристрій має агент SNMP, попередньо завантажений його виробником, і якщо ця програма виявляє проблему з пристроєм, вона надсилає сповіщення. Служба Datadog Network Device Monitoring збирає ці повідомлення та відображає їх на системній консолі як сповіщення.

Механізм попередження в Datadog Network Device Monitoring можна розширити, щоб генерувати попередження, якщо будь-який із показників, зібраних системою, перевищить вказане порогове значення. Ви можете встановити власні порогові значення, щоб отримувати сповіщення за будь-яких умов, які ви виберете. Система сповіщень також доступна в Network Performance Monitor, тому обмеження трафіку будуть виділені, коли почнуть утворюватися вузькі місця.

Ви можете налаштувати правило пересилання сповіщень, щоб отримувати сповіщення електронною поштою, SMS, повідомленнями PagerDuty або Slack. Це означає, що ви можете залишити дві системи моніторингу мережі для автоматичного моніторингу мережі, оскільки ви знаєте, що отримаєте сповіщення, якщо потрібна буде увага людини.

Служби моніторингу трафіку в модулі Network Performance Monitoring не просто підраховують пакети, що циркулюють мережею. Він також здатний ідентифікувати трафік даних у системі віртуалізації та може здійснювати моніторинг активності для контейнерів. Інструмент поширюється на хмарні служби, де він може показувати вам вхідний і вихідний трафік з кожного вашого хмарного облікового запису.

Централізовані центри обробки даних найбільше виграють від служби Datadog Network Monitoring. Пакет можна налаштувати віддалено для моніторингу будь-якої мережі без відвідування сайту.

Переваги:

– Автоматична інвентаризація активів: Datadog підтримує автоматичне виявлення, створюючи та підтримуючи інвентар мережевих активів, який постійно оновлюється, коли додаються нові пристрої або відбуваються зміни.

– Шаблони трафіку та віртуалізації: Datadog надає можливості моніторингу шаблонів трафіку, включаючи активність у віртуалізованих середовищах, надаючи розуміння використання мережі та продуктивності.

– Доступність з будь-якого місця: як хмарна система, Datadog пропонує перевагу доступності з будь-якого місця, позбавляючи користувачів від необхідності керувати власною інфраструктурою.

Недоліки:

– Бажання більш тривалого випробувального періоду: продовження випробувального періоду може дати користувачам більше часу, щоб повністю вивчити та оцінити платформу, перш ніж прийняти рішення.

Моніторинг продуктивності мережі Datadog і моніторинг мережевих пристроїв Datadog – це лише два модулі платформи Datadog. Ці системи добре працюють разом з іншими моніторами Datadog, які включають службу синтетичного моніторингу та Datadog APM. Ви можете спробувати будь-які пристрої Datadog за допомогою 14-денної безкоштовної пробної версії.

3. Мережевий монітор Paessler PRTG

Візуалізація характерного стилю калібрування PRTG стала основою для мережевих адміністраторів. PRTG Network Monitor від Paessler – це безкоштовний пакет моніторингу мережі, який використовує SNMP, аналіз пакетів і WMI для моніторингу мережі. Скануйте сегменти мережі, щоб виявити та додати пристрої для моніторингу. Ви можете вибирати між рядом датчиків для моніторингу різних сегментів вашої мережі. Кожен датчик відстежує окреме значення у вашій мережі, наприклад, є датчики моніторингу смуги пропускання, датчики апаратних параметрів, вимірювачі використання даних мережі, датчики SNMP, датчики VOIP і QoS тощо.

Основні характеристики:

– Моніторинг SNMP: Моніторинг PRTG SNMP дозволяє збирати дані про продуктивність мережевих пристроїв із підтримкою SNMP.

– Моніторинг пропускної здатності: Платформа включає можливості моніторингу пропускної здатності, що дозволяє відстежувати та аналізувати використання пропускної здатності мережі.

– Виявлення пристроїв у сегменті IP: PRTG дозволяє сканувати мережеві пристрої за сегментом IP, спрощуючи процес виявлення та додавання пристроїв до системи моніторингу.

– Система сповіщень на основі порогових значень: PRTG має систему попереджень на основі порогових значень, яка сповіщає адміністраторів про виконання певних умов або порогових значень.

– Настроювані карти мережі: Настроювані карти мережі надають візуальне представлення топології мережі відповідно до вподобань користувача.

– Підтримується безкоштовна версія: PRTG пропонує безкоштовну версію, що дозволяє користувачам безкоштовно випробувати та використовувати основні функції.

З локальних пакетів у цьому списку я виявив, що PRTG є, мабуть, найкращим варіантом моніторингу часу відповіді для веб-сайтів. Система PRTG містить датчик веб-сторінки для реєстрації часу завантаження сторінок. Ви можете використовувати його на вимогу або налаштувати на виконання за розкладом. Я також виявив, що набір включає монітор доступності на основі Ping для веб-сайтів. Усі датчики моніторингу веб-сайтів можна включити в безкоштовну версію системи PRTG.

Циферблати вказують, наскільки близько кожен пристрій і процес до своєї максимальної потужності.

Paessler PRTG – це дуже гнучкий сервіс. Ви можете вибрати, які датчики вмикати, і створити власну індивідуальну систему моніторингу для мереж, серверів і програм.

Я був вражений чудовою візуалізацією, інструмент має настроювану інформаційну панель, яка дозволила мені створювати мережеві карти нашої інфраструктури в реальному часі. Ви можете створювати карти за допомогою редактора перетягування та використовувати понад 300 об'єктів карти для створення карти. Ви навіть можете створювати власні об'єкти карти за допомогою спеціального HTML.

Система сповіщень на основі порогових значень інформує вас про зміну статусу датчика, значення або перевищення порогового значення. Сповіщення доступні в різних форматах, включаючи електронну пошту, push-повідомлення, SMS, повідомлення Slack, повідомлення системного журналу, перехоплення SNMP, дію HTTP, виконання програми тощо.

Комплексний характер PRTG забезпечує широкий моніторинг для всіх категорій IT-систем, що робить його хорошим вибором для великих організацій. Однак безкоштовний варіант, обмежений 100 датчиками, дуже привабливий для малого бізнесу.

Переваги:

– Багатопротокольний збір даних: PRTG використовує комбінацію аналізу пакетів, WMI та SNMP для звітування даних про продуктивність мережі, забезпечуючи повне уявлення.

– Повністю настроювана інформаційна панель: повністю настроювана інформаційна панель підходить як для самотніх адміністраторів, так і для команд NOC, пропонуючи гнучкість у відображенні релевантної інформації.

– Датчики для певних програм: PRTG надає готові датчики, призначені для моніторингу певних програм, наприклад датчики для захоплення та моніторингу активності VoIP.

Недоліки:

– Крива навчання: багато функцій PRTG можуть вимагати часу для адміністраторів, щоб навчитися та повністю використовувати всі аспекти системи.

Підприємства, які шукають економічно ефективні, але комплексні інструменти моніторингу мережі, повинні спробувати PRTG Network Monitor. Платформа доступна безкоштовно для менш ніж 100 датчиків, після чого вам доведеться платити. Платні версії починаються від 1600 доларів США (~1335 фунтів стерлінгів) за 500 датчиків і 1 встановлення сервера. Програмне забезпечення для PRTG працює на Windows Server, і ви можете отримати його на 30-денну безкоштовну пробну версію.

4. Domotz

Штрих у стилі піктограм чітко класифікує різні аспекти топології мережі. Domotz – це хмарна система, яка пропонує моніторинг мережі. Він може об'єднати нагляд за кількома мережами, а також має конфігурацію з кількома клієнтами для використання керованими постачальниками послуг.

Доступ до системи здійснюється через привабливу хмарну консоль із будь-яким стандартним веб-браузером. Налаштування облікового запису передбачає встановлення агента на хості, підключеному до мережі, за якою буде здійснюватися моніторинг. Пакет також включає засоби керування мережею та кінцевими точками та засоби автоматизації робочого навантаження.

Основні характеристики:

– Автоматичне виявлення мережі: це спрощує процес ідентифікації та додавання пристроїв до мережі.

– Автоматичне відображення мережі: ця функція забезпечує візуальне представлення топології мережі.

– Інвентаризація IT-активів: Domotz підтримує управління інвентаризацією IT-активів, допомагаючи користувачам відстежувати пристрої, присутні в мережі.

– Моніторинг пристрою: дозволяє збирати дані про продуктивність із пристроїв із підтримкою SNMP.

Domotz приділяє велику увагу віддаленому доступу та можливостям керування. Ви можете безпечно підключатися до пристроїв у їхній мережі з будь-

якої точки світу, спрощуючи дистанційне усунення несправностей і конфігурацію, що особливо корисно для MSP і компаній з кількома місцезнаходженнями.

До цієї хмарної системи можна отримати доступ через будь-який стандартний веб-браузер, і вона може контролювати кілька сайтів одночасно. Щойно нам було надано доступ до сайту, функція автоматичного виявлення Domotz склала та підтримувала інвентаризацію та карту мережі.

Пакет Domotz має велику кількість функцій. Інструмент зосереджений на моніторингу мережі через SNMP та інші мережеві протоколи. Це включає можливість легкого налаштування датчиків SNMP і сповіщень. Ви також можете вказати власні умови оповіщення.

Звіти SNMP дозволяють системі складати інвентаризацію активів і складати карту мережі. Ці служби оновлюються з кожним циклом відповіді агента SNMP. Інформація, додана до кожного вузла на карті та в інвентарному списку, деталізує марку та модель кожного розділу та його пропускну здатність.

Пакет також містить програму відображення портів комутатора та інструменти керування в пакеті, які дозволяють віддалений доступ і керування конфігурацією мережевих пристроїв. Сервіси IoT також можуть бути доступні та керовані віддалено. Система здатна перетинати мережеві платформи для доступу до бездротових мереж і пристроїв завдяки сотням інтеграцій і можливості впровадження індивідуальних драйверів моніторингу.

Domotz пропонує версію з кількома клієнтами, яка добре підходить для використання постачальниками керованих послуг. Внутрішні операційні групи також можуть використовувати цю систему для моніторингу на кількох об’єктах.

Переваги:

– Кілька облікових записів користувачів на клієнта: Domotz підтримує кілька облікових записів користувачів на клієнта, полегшуючи співпрацю та доступ до даних моніторингу.

– Моніторинг кількох сайтів: Domotz об’єднує моніторинг кількох сайтів, забезпечуючи централізований перегляд продуктивності мережі в різних місцях.

– Шифрування для покращеної безпеки: Платформа використовує шифрування для захисту взаємодії між платформою Domotz і мережею, забезпечуючи безпечне спілкування.

Недоліки:

– Відсутність агента кінцевої точки для macOS: відсутність агента кінцевої точки, спеціально розробленого для macOS, може вплинути на ефективність моніторингу на пристроях macOS порівняно з іншими платформами.

Існує два тарифні плани для Domotz. Плата за пристрій коштує до 1,50 доларів США на місяць за пристрій, а їхній план розташування коштує 35 доларів США або менше на місяць за кожне місце. Ви можете спробувати Domotz у 14-денній безкоштовній пробній версії.

5. Checkmk

Візуальне визначення точок стресу в різних часових масштабах Checkmk – це пакет системного моніторингу, який здатний відстежувати продуктивність мереж, серверів і програм. Монітор мережі можна використовувати для локальних і бездротових мереж, тому він також може бути корисним для відстеження активності в мережах, які використовують дротові та бездротові технології.

Основні характеристики:

– Виявлення мережі: Платформа включає можливості виявлення мережі, полегшуючи ідентифікацію та додавання пристроїв до системи моніторингу.

– Карта топології мережі. Карта топології мережі надає візуальне представлення зв’язків і зв’язків у межах мережевої інфраструктури.

– Дротові та бездротові мережі: відстежує як дротові, так і бездротові мережі, забезпечуючи видимість продуктивності різних типів мережевих підключень.

Checkmk виділяється своїм автоматичним виявленням служб для моніторингу інфраструктури, що спрощує початкове налаштування та будь-яке

поточне керування. Коли ви підключаєте Checkmk до сервера, він автоматично визначає операційну систему сервера та запущені на ньому служби, а потім пропонує відповідні перевірки для цих служб.

Checkmk Raw – фантастичний безкоштовний пакет моніторингу системи. Ця система охоплює бездротові мережі, а також локальні мережі, і вона відстежуватиме шаблони трафіку разом із статусами мережевих пристроїв.

Під час тестування ця система моніторингу мережі розпочала роботу з пошуку в мережі та ідентифікації всіх підключених пристроїв. Пакет склав перелік пристроїв за результатами цього пошуку, і отриманий список обладнання став основою для звітів про стан панелі моніторингу мережі. Пакет також створив живу карту мережі.

Базовий пакет Checkmk абсолютно безкоштовний для використання. Існує також платна версія системи під назвою Checkmk Enterprise, яка має версію, призначену для постачальників керованих послуг.

Як безкоштовний інструмент базова послуга дуже приваблива для малих підприємств і компаній з обмеженим бюджетом. Монітор запитує комутатори, маршрутизатори та брандмауери через систему SNMP. На всіх ваших мережевих пристроях уже будуть встановлені агенти SNMP, але цю функцію, можливо, потрібно буде ввімкнути. Програмне забезпечення Checkmk опитує агентів щодо звітів про статус. Відповіді збираються в поточні звіти про готовність мережі та продуктивність пропускної здатності.

Інформаційна панель дозволяє адміністраторам мережі отримувати огляд усієї активності, а потім має ряд опцій для перегляду звітів про окремі пристрої як у реальному часі, так і за певний час. Пакет Checkmk також пропонує моніторинг смуги пропускання, який дає вам індикатори того, де ємності пристрою та кабелю недостатньо для задоволення попиту.

Система Checkmk Raw є ідеальним рішенням для малих підприємств і стартапів, які жорстко контролюють витрати. Варіант на основі пристрою для платного Checkmk Enterprise зберігає сервери вільними для розміщення програм.

Переваги:

– Локальне програмне забезпечення: Checkmk – це локальне програмне забезпечення, яке забезпечує перевагу контролю для користувачів, які віддають перевагу керувати своїм рішенням локально.

– Рентабельність для малого бізнесу: Checkmk вважається хорошою угодою для малого бізнесу.

– Інтегрований моніторинг продуктивності серверів і додатків: забезпечує єдине уявлення про все IT-середовище.

Недоліки:

– Походить від Nagios Core: Хоча це не обов'язково є недоліком, це може сприйматися як недолік для тих, хто має особливі переваги щодо самого Nagios Core.

Платна версія Checkmk оцінюється за ковзною шкалою на основі кількості хостів у мережі, що контролюється. Отримайте доступ до безкоштовної версії, яка називається Checkmk Raw, щоб оцінити пакет. І Checkmk Raw, і Checkmk Enterprise працюють у Linux. Версію Enterprise також можна придбати на фізичному пристрої. Для Checkmk Enterprise доступна 30 -денна безкоштовна пробна версія

6. NinjaOne RMM

Проблемні зони мережі позначені кольором і їх легко побачити. NinjaOne RMM надає послуги віддаленого моніторингу та керування з хмарної платформи. Доступ до цієї системи можна отримати з будь-якого місця за допомогою будь-якого стандартного веб-браузера, тому вашому бізнесу не потрібне спеціальне обладнання чи навіть сервери для адміністрування кількох віддалених мереж.

Основні характеристики:

– Автовиявлення: можливості автоматичного виявлення NinjaOne спрощують процес ідентифікації та додавання пристроїв до мережі.

– Моніторинг стану пристроїв: Моніторинг стану пристроїв NinjaOne дає змогу в режимі реального часу оцінювати справність і продуктивність мережевих пристроїв.

– Аналіз трафіку: NinjaOne підтримує аналіз трафіку, що дозволяє користувачам аналізувати пропускну здатність і шаблони мережевого трафіку.

RMM – це багатокористувацький пакет для використання керованими постачальниками послуг. MSP створює окремий суб-акаунт для кожного клієнта, зберігаючи дані про послуги кожного клієнта окремо. Кожен клієнт також може мати декілька сайтів, зареєстрованих у пакеті моніторингу.

NinjaOne RMM оптимізує продуктивність ваших IT-технічних спеціалістів. Сповіщення в пакеті означають, що нікому не потрібно сидіти й спостерігати за діяльністю на кожному віддаленому місці, оскільки технічну групу буде повідомлено, якщо виникне проблема. Система перевіряє доступність мережевого пристрою та відстежує обсяги пропускної здатності трафіку.

Коли MSP має нового клієнта для моніторингу, техніку потрібно встановити агент NinjaOne на комп'ютері, підключеному до віддаленої мережі. Цей агент надсилає запити на звіти про стан мережевої служби, а потім пересилає отримані дані на сервер NinjaOne.

Ці звіти про пристрої показують, яке обладнання є в мережі. Під час тестування NinjaOne RMM склав інвентаризацію обладнання з цієї інформації. Окрім довідкової інформації, інструмент повернув дані про активність пристрою в реальному часі, які він відображав на інформаційній панелі. Ці звіти містять більше наборів даних, ніж може відобразити консоль, і ми отримали можливість налаштувати інформаційну панель, щоб вибрати показники, які, на вашу думку, важливі.

Кожна зібрана статистика може мати порогові значення продуктивності, і коли ці рівні перевищуються, система запускає сповіщення. RMM також збирає трафік через дані за допомогою протоколів NetFlow, J-Flow, sFlow і IPFIX. Ці статистичні дані також можуть мати порогові значення для сповіщень.

NinjaOne RMM розроблено для використання керованими постачальниками послуг. Однак ніщо не заважає IT-відділам використовувати платформу для внутрішнього моніторингу. Ми виявили, що інструмент може контролювати кілька сайтів одночасно, а також включає системи моніторингу та керування кінцевими точками, як-от менеджер виправлень.

Переваги:

– Одночасний моніторинг кількох сайтів: NinjaOne RMM дозволяє здійснювати одночасний моніторинг кількох сайтів, забезпечуючи централізований перегляд продуктивності мережі в різних місцях.

– Субрахунки для розділення даних: Платформа підтримує субрахунки, що дозволяє розділяти дані для кожного клієнта.

– Аналіз пропускної здатності трафіку: аналіз пропускної здатності трафіку дає користувачам уявлення про обсяг і моделі потоку даних у мережі.

Недоліки:

– Немає публічного прайс-листа, потрібні індивідуальні котирування: хоча це дозволяє встановлювати індивідуальні ціни на основі конкретних потреб, деякі користувачі можуть віддавати перевагу прозорій інформації про ціни заздалегідь.

11-разове підтвердження від G2 поставило NinjaOne на перше місце в RMM, Patch Management і Endpoint Management. Ви можете отримати індивідуальну ціну для NinjaOne відповідно до ваших потреб. Ви можете почати з реєстрації на 14-денну безкоштовну пробну версію.

7. ManageEngine OpManager

Ця настроювана інформаційна панель дозволяє вам зосередитися на ваших унікальних сферах інтересів ManageEngine OpManager – це рішення для моніторингу мережі, яке може контролювати продуктивність мережевих пристроїв, серверів, маршрутизаторів, комутаторів і віртуальних машин у режимі реального часу. Інформаційні панелі, які можна налаштувати, містять понад 200 віджетів для створення унікального досвіду моніторингу.

Основні характеристики:

– Автоматичне виявлення: можливості автоматичного виявлення OpManager спрощують процес ідентифікації та додавання нових пристроїв до мережі.

– Відображення мережі: функція відображення мережі дозволяє користувачам візуалізувати макет і з'єднання в межах своєї мережевої інфраструктури.

– Моніторинг SNMP: OpManager підтримує моніторинг SNMP, дозволяючи збирати дані про продуктивність із пристроїв із підтримкою SNMP.

На відміну від інших локальних інструментів моніторингу мережі в цьому списку, OpManager не обмежується відстеженням мережевого обладнання, оскільки він також моніторить сервери. Засоби моніторингу сервера в системі OpManager записують важливі показники ресурсів, такі як ЦП, пам'ять, ємність і використання диска.

Система ManageEngine OpManager є однією з небагатьох детальних систем моніторингу мережі, яка пропонує графічний інтерфейс користувача для Linux. Хоча більшість мережевих моніторів Linux є системами командного рядка, високоякісні графіки та діаграми OpManager полегшують розпізнавання стану.

Моніторинг SNMP тримає вас в курсі продуктивності пристроїв у вашій мережі. Для кращої видимості ви можете використовувати функцію відображення мережі для автоматичного виявлення та відображення нових пристроїв. Ви можете запланувати виявлення мережі, щоб знайти нові пристрої, щойно їх буде додано до вашої мережі.

Ми протестували систему сповіщень, яка допомагає вам швидко реагувати на зміни продуктивності. Він працював бездоганно. ManageEngine OpManager корелює мережеві події та надає користувачеві лише відповідні сповіщення, мінімізуючи помилкові спрацьовування. Програма надсилає сповіщення електронною поштою та SMS, щоб інформувати вас про будь-які проблеми, що виникають.

ManageEngine OpManager – це універсальний мережевий монітор, рекомендований користувачам, яким потрібен простий інструмент моніторингу інфраструктури.

Цей програмний пакет встановлюється на Linux, Windows Server, Azure та AWS. Незважаючи на те, що він добре працює на всіх цих платформах, з усіх систем моніторингу мережі, призначених для роботи в Linux, ManageEngine OpManager вважається найкращим варіантом.

Переваги:

– Унікальні інформаційні панелі та звіти: надає понад 200 настроюваних віджетів для створення унікальних інформаційних панелей і звітів, що забезпечує гнучкість для користувачів.

– Зменшена кількість помилкових спрацьовувань: OpManager використовує інтелектуальне сповіщення, щоб зменшити кількість помилкових спрацьовувань і усунути втому від сповіщень.

– Інтеграція в екосистему ManageEngine: OpManager добре інтегрується з іншими продуктами в екосистемі ManageEngine, підвищуючи його функціональність і сумісність з іншими інструментами управління IT.

Недоліки:

– Крива навчання: OpManager – це багатофункціональний інструмент, який може потребувати вкладень часу, щоб правильно вивчити та використовувати всі його функції.

ManageEngine OpManager – це універсальний мережевий монітор, рекомендований користувачам, яким потрібен простий інструмент моніторингу інфраструктури. Платні версії починаються від 245 доларів США (~195 фунтів стерлінгів) для 10-1000 пристроїв до 11 545 доларів США (~9 000 фунтів стерлінгів для 250-10 000 пристроїв. Ви можете завантажити 30-денну безкоштовну пробну версію.

8. Atera

Atera – це комплексна система віддаленого моніторингу та керування (RMM), розроблена в основному для постачальників IT-послуг і постачальників керованих послуг (MSP). Він поєднує потужні засоби моніторингу з функціями віддаленого керування, щоб забезпечити цілісне рішення для підтримки та оптимізації IT-середовища. Завдяки своїй хмарній платформі Atera дозволяє користувачам ефективно контролювати свої мережі, виявляти проблеми та впроваджувати рішення в режимі реального часу.

Основні характеристики:

– Моніторинг у реальному часі: забезпечує постійну видимість продуктивності мережі та стану пристрою, сповіщаючи користувачів про проблеми, щойно вони виникають.

– Автоматичні сповіщення: настроювані сповіщення про критичні події гарантують, що користувачі можуть оперативно реагувати на аномалії мережі.

– Віддалене керування: забезпечує віддалений доступ до пристроїв для усунення несправностей і обслуговування, скорочуючи час простою та підвищуючи ефективність обслуговування.

Інтегрована система виставлення рахунків і виставлення рахунків Atera дозволяє MSP керувати своїми фінансами разом із моніторингом мережі, оптимізуючи операції та забезпечуючи точне виставлення рахунків за надані послуги. Ця функція підвищує операційну ефективність і допомагає користувачам здійснювати чіткий фінансовий контроль.

Універсальна платформа Atera поєднує надійний моніторинг мережі з потужними можливостями віддаленого керування, що робить її ідеальною для IT-фахівців. Його зручний інтерфейс і інтегровані функції виставлення рахунків спрощують операції, дозволяючи командам зосередитися на наданні високоякісних послуг.

Atera надає надбудову для виявлення мережі, автоматично ідентифікуючи пристрої та служби в мережі. Ця функція дає IT-фахівцям змогу отримати точне

уявлення про свою інфраструктуру, полегшуючи проактивне керування та швидку ідентифікацію неавторизованих пристроїв. Інтуїтивно зрозумілий інтерфейс дозволяє легко візуалізувати топологію мережі, допомагаючи командам краще зрозуміти своє середовище.

Що стосується функцій безпеки, Atera надає інструменти для моніторингу вразливостей мережі та забезпечення відповідності галузевим стандартам. Регулярне сканування допомагає виявити потенційні загрози, а автоматичні сповіщення сповіщають користувачів про будь-які порушення безпеки або аномалії. Цей проактивний підхід дозволяє організаціям ефективно зменшувати ризики та захищати свої мережеві активи.

Можливості Atera виходять за межі моніторингу мережі; він об'єднує різні інструменти управління IT в єдину платформу. Користувачі отримують переваги від системи продажу квитків, управління взаємовідносинами з клієнтами (CRM) і автоматизованих робочих процесів, створюючи згуртоване середовище для IT-операцій. Ця інтеграція не тільки підвищує ефективність, але й покращує співпрацю між членами команди, полегшуючи надання виняткових послуг клієнтам.

Atera рекомендовано постачальникам IT-послуг, постачальникам керованих послуг (MSP) і організаціям, які шукають комплексне рішення для моніторингу та керування мережею. Його потужні функції призначені як для малих підприємств, так і для великих підприємств, які прагнуть оптимізувати свої IT-операції.

Переваги:

– Інструменти звітування: широкі функції звітування дозволяють користувачам аналізувати продуктивність мережі та використання ресурсів з часом.

– Управління активами: автоматично відстежує апаратні та програмні активи, надаючи інформацію про інвентаризацію та управління життєвим циклом.

– Інтегрована система виставлення рахунків: оптимізує фінансове управління разом із моніторингом, покращуючи операційну ефективність.

– Ефективність робочого процесу: допомагає оптимізувати робочі процеси та підвищити ефективність

Недоліки:

– Обмежене налаштування: певні аспекти платформи можуть бути недоступні для налаштування в тій мірі, в якій цього бажають деякі користувачі.

Ця хмарна система включає все програмне забезпечення, необхідне постачальнику керованих послуг для роботи, включаючи моніторинг мережі, а також є плани для IT-відділів. Atera пропонує гнучкі плани передплати залежно від кількості технічних спеціалістів, із чотирма планами для MSP та чотирма для IT-відділів. Плани включають різноманітні функції, які гарантують користувачам доступ до основних інструментів для ефективного керування мережею. Atera також надає 30-денну безкоштовну пробну версію, що дозволяє потенційним користувачам досліджувати її можливості без зобов'язань.

9. Obkio

Працюючи з хмарної платформи, Obkio виходить на мережу через встановлення локальних агентів. Він здатний перевіряти шаблони трафіку між агентами, тому чим більше агентів ви встановите, тим більше точок даних ви матимете. Пакет також реалізує моніторинг пристроїв за допомогою простого протоколу керування мережею (SNMP).

Основні характеристики:

– Моніторинг мережі в режимі реального часу: безперервно відстежує продуктивність мережі за допомогою розподілених агентів, розгорнутих у різних місцях мережі.

– Підходить для гібридних систем: монітори (локальні локальні мережі, посилання на хмарні служби та віддалені з'єднання.

– Показники якості обслуговування: інформація про затримку, втрату пакетів і тремтіння.

Obkio пропонує наскрізну видимість мережі в режимі реального часу, що є критично важливим для компаній, які покладаються на розподілені мережі та хмарну інфраструктуру. Здатність платформи виявляти проблеми в різних місцях робить її дуже ефективним інструментом для оптимізації продуктивності мережі. Крім того, його проактивне сповіщення гарантує, що IT-команди зможуть вирішити проблеми до того, як вони вплинуть на бізнес-операції.

Система розроблена, щоб дати компаніям глибоке уявлення про продуктивність своєї мережі за допомогою розподіленої агентської архітектури. Це забезпечує постійний моніторинг продуктивності мережі між різними сегментами мережі, такими як офіси, центри обробки даних і хмарні служби, забезпечуючи оптимальне функціонування всіх мережевих компонентів.

Платформа дуже інтуїтивно зрозуміла, що робить її доступною як для невеликих IT-команд, так і для великих організацій. Завдяки таким функціям, як автоматичне сповіщення, історичне відстеження продуктивності та вбудовані інструменти діагностики, Obkio пропонує комплексне рішення для безперебійної та ефективної роботи мереж.

Чотири видання Obkio роблять його доступним для компаній будь-якого розміру. Його можуть використовувати постачальники керованих послуг або IT-відділи для внутрішнього моніторингу. Система особливо важлива для компаній, які обслуговують VoIP та інші інтерактивні послуги, такі як потокове відео. Інструмент шукає проблеми з пристроєм, а також перевіряє посилання на інші сайти та хмарні служби.

Переваги:

– Автоматичні сповіщення та сповіщення: надсилає сповіщення на основі налаштованих порогових значень, щоб повідомити адміністраторів про потенційні проблеми з продуктивністю, перш ніж вони посиляться.

– Інструменти діагностики мережі: містить розширені інструменти діагностики, як-от трасування маршрутів і тести швидкості для швидкого виявлення та усунення вузьких місць мережі.

– Аналіз історичних даних: зберігає дані про продуктивність мережі для аналізу тенденцій, планування потужності та усунення несправностей.

Недоліки:

– Без керування мережею: не включає такі інструменти, як керування конфігурацією мережі.

Obkio пропонує чотири рівні плану: початковий, базовий, преміум і корпоративний, кожен наступний вищий випуск включає більше послуг. Ви можете отримати 14-денну безкоштовну пробну версію версії Premium.

10. Моніторинг мережі Site24x7

Мережеві сповіщення відображаються на помітному місці та позначаються кольором. Хмарний інструмент моніторингу мережі Site24x7, який охоплює IT-інфраструктуру, програми та поведінку користувачів. Розділ моніторингу мережі інструменту може автоматично виявляти всі пристрої, підключені до мережі. Site24x7 складає інвентаризацію обладнання з результатів цього пошуку, а потім автоматично складає карти топології мережі. Інвентар і карти оновлюються автоматично щоразу, коли обладнання додається, переміщується або видаляється.

Основні характеристики:

– Рішення «все-в-одному»: комплексне рішення для моніторингу, яке охоплює мережі, інфраструктуру та моніторинг реальних користувачів на одній платформі.

– Моніторинг у реальному часі: забезпечує моніторинг у реальному часі як програм, так і мережевих пристроїв, що дозволяє швидко ідентифікувати проблеми та реагувати на них.

– Потужні звіти, аналітика та сповіщення: пропонує надійні інструменти звітності та аналітики для поглибленого аналізу показників ефективності та тенденцій.

Кожен пакет Site24x7 включає інструменти моніторингу веб-сайтів, які перевищують базові перевірки підключення та доступності. Пакет також

перевіряє підключення до хмарних платформ. Отже, Site24x7 пропонує моніторинг інтернет-посилань, а також локальних мереж.

Сервіс моніторингу мережі Site24x7 постачається разом з іншими моніторами для серверів, програм та веб-активів. Ви обираєте, які сервіси відстежувати, та налаштовуєте порогові значення продуктивності, які запускають сповіщення.

Функція автоматичного виявлення Site24x7 керується простим протоколом керування мережею (SNMP). Ця система вбудована в усі мережеві пристрої. Йому потрібен лише менеджер SNMP, щоб запитувати інформацію про статус у програмного забезпечення агента, яке завантажується на кожен комутатор і маршрутизатор їхніми виробниками. Site24x7 виконує цей менеджер SNMP.

SNMP містить процедуру, яка дозволяє агентам пристрою надсилати сповіщення менеджеру, коли він виявляє критичний стан пристрою, який він контролює. Інформаційна панель Site24x7 перетворює це повідомлення на попередження або сповіщення залежно від рівня серйозності, зазначеного в повідомленні від агента.

Site24x7 відстежує комутатори, маршрутизатори, брандмауери, джерела живлення, балансувальники навантаження, бездротові мережі, хмарні сервіси та з'єднання WAN. Це не монітор трафіку. Монітор фокусується на підключеному обладнанні. Більш широкий погляд на пакет Site24x7 показує, що система також контролює інше обладнання, наприклад кінцеві точки та сервери. Він також відстежує продуктивність програм. Модуль поведінки користувача в пакеті особливо корисний компаніям, які керують веб-сайтами. Він включає моніторинг реальних користувачів для аналізу типових шляхів покупців, а також синтетичний моніторинг веб-транзакцій, який імітує доступ до веб-сайту.

Плани Site24x7 особливо сильні щодо моніторингу веб-додатків і поєднання цих послуг із послугами тестування мережі та з'єднання. Тому компанії, які розміщують веб-сайти, виграють від вибору цього варіанту.

Переваги:

– Поєднання технічних і бізнес-показників: реальні можливості моніторингу користувачів допомагають подолати розрив між технічними проблемами, поведінкою користувачів і бізнес-показниками.

– Безкоштовна версія: надає безкоштовну версію для тестування, що дозволяє користувачам ознайомитися з функціями та можливостями перед тим, як оформити повну підписку.

Недоліки:

– Крива навчання: як і деякі інші інструменти в цій категорії, Site24x7 має детальну платформу, яка може потребувати значного часу, щоб повністю зрозуміти та використовувати всі функції та параметри.

Site24x7 пропонує безкоштовну службу моніторингу мережі. Однак це лише звичайний монітор безвідмовної роботи на основі Ping для до 50 URL-адрес або серверів. Хмарна платформа пропонує низку інструментів моніторингу та керування системою, і вони упаковані в випуски. Всього п'ять видань:

– Моніторинг сайту
– Інфраструктура
– APM
– Все в одному
– MSP

Усі ці пакети включають службу моніторингу мережі. Кожен із планів Site24x7 доступний у 30-денній безкоштовній пробній версії.

11. Fortra's Intermapper

Вичерпні карти та часові графіки у вас під рукою Fortra's Intermapper пропонує карту мережі на головному екрані. Однак ми виявили, що цей інструмент пропонує більше, ніж просто візуальне представлення вашої мережі. Для початку ця карта мережі отримана з функції автоматичного виявлення. Зонд мережі постійно циклічно обертається, і якщо в макеті мережі відбуваються будь-які зміни, карта мережі оновлюється автоматично. Система працює цілодобово.

Основні характеристики:

– Автоматичне виявлення: функція автоматичного виявлення пристроїв у мережі, що спрощує процес початкового налаштування.

– Моніторинг мережі на основі SNMP: використовує SNMP для моніторингу мережі, дозволяючи збирати дані про продуктивність мережевих пристроїв.

– Сповіщення про продуктивність: пропонує сповіщення про продуктивність, щоб повідомити користувачів про будь-які проблеми або відхилення від нормальної роботи мережі, сприяючи проактивному вирішенню проблем.

– Інструмент планування потужності: містить інструмент планування потужності, який допомагає користувачам оцінювати ресурси та можливості своєї мережі та керувати ними.

Візуальна інформація про справність мережі в режимі реального часу допомагає виявити вузькі місця, виявити збої пристроїв і надати кращу чіткість впливу будь-яких змін у мережі.

Intermapper накопичує багато інформації на одному екрані, що заощадило мені час на перегляд інформації про продуктивність мережі, оскільки мені не потрібно було перемикати сторінки в інтерфейсі. Intermapper реалізував автоматичне виявлення та створив рудиментарну карту для ілюстрації звітів про стан продуктивності в реальному часі.

Карта мережі функціонує як меню з деталями кожного пристрою в мережі. Клацніть піктограму, щоб побачити активність на цьому конкретному вузлі. Ви побачите дані про пропускну здатність, а також звіти про стан.

Мені вдалося встановити порогові рівні продуктивності для кожного з показників, які відстежує монітор, наприклад потужність ЦП або інтерфейс. Це допомогло мені відстежувати доступність та інші умови, пов’язані з угодою про рівень обслуговування, оскільки система видасть сповіщення, якщо один із цих порогових значень буде перевищено.

Ви можете налаштувати параметри в системі так, щоб Intermapper надсилав вам електронний лист або текстове повідомлення, якщо виникне сповіщення. Це означає, що мені не потрібно було весь час сидіти й дивитися на екран моніторингу мережі.

Fortra Intermapper – це безкоштовний інструмент, тому це чудовий варіант для підприємств, які намагаються скоротити витрати. Це локальний пакет, і це одна з небагатьох систем моніторингу мережі, які працюватимуть на macOS – він також доступний для Windows і Linux.

Переваги:

– Сервер не потрібен: Fortra не потребує виділеного сервера для кожного хоста; достатньо ПК, що спрощує розгортання та зменшує вимоги до інфраструктури.

– Безперервний автоматизований моніторинг: забезпечує безперервний автоматизований моніторинг, гарантуючи, що мережа перебуває під постійним наглядом на предмет можливих проблем.

– Доступна безкоштовна версія: Fortra надає безкоштовну версію, що дозволяє користувачам випробувати основні функції без необхідності негайного фінансового зобов'язання.

Недоліки:

– Немає хмарної версії: бракує хмарної версії, що може обмежити гнучкість для організацій, які віддають перевагу або потребують хмарних рішень.

– Обмеження пристроїв у безкоштовній версії: безкоштовна версія обмежена моніторингом лише п'яти пристроїв, що потенційно обмежує її корисність для великих мереж.

Intermapper від Fortra – це локальне програмне забезпечення, яке встановлюється на Windows, Linux або macOS. Доступна безкоштовна версія, яка може контролювати до п'яти пристроїв. Платна версія має масштабовану ціну, яка починається з десяти пристроїв. Ви можете вибрати між підпискою або постійною

ліцензією. Платна версія Intermapper пропонується з 30-денною безкоштовною пробною версією.

12. AdRem NetCrunch

AdRem NetCrunch надає послуги моніторингу для спостереження за мережами, кінцевими точками та програмним забезпеченням. Це локальний пакет програмного забезпечення, і він може охопити вашу мережу, щоб перевірити всі кінцеві точки, а не лише хост пакета. Послуга включає процедуру виявлення мережі, яка виявляє всі пристрої, записує інвентаризацію обладнання та створює карту топології мережі.

Основні характеристики:

– Виявлення системи: сканування повторює та автоматично оновлює документацію

– Інвентаризація мережі та карта топології: надає огляд мережі та всіх пристроїв у ній

– Моніторинг продуктивності в реальному часі: аналізує мережеві служби, а також пристрої

Однією з ключових переваг AdRem NetCrunch є простота використання. Після встановлення програмного забезпечення користувачеві достатньо виконати простий посібник із налаштування, а потім інструмент здійснить пошук у вашій мережі та самостійно заповнить усі її таблиці. Карта мережі діє як індекс для перегляду деталей пристрою та сповіщень, які сповіщають вас, коли в системному компоненті виникають проблеми.

Незважаючи на те, що NetCrunch здатний контролювати сервери та хмарні служби, його головною особливістю є система моніторингу мережі. Механізм попередження інструмента містить заздалегідь написані правила. Правило – це ланцюжок дій, який починається зі збору даних – це моніторингова частина процесу. Кожне вимірювання порівнюється з діапазоном допустимих значень. Якщо статистика виходить за межі діапазону, NetCrunch генерує сповіщення.

Сповіщення системи NetCrunch з’являються на консолі, і ви можете налаштувати службу для пересилання цих попереджень як сповіщень певним людям у вашій команді. Ці повідомлення можна надсилати електронною поштою, Slack, SMS, Teams або іншими системами співпраці. Інструмент також може передавати квитки у вашу систему керування проектами, наприклад Jira або пакет маршрутизації Service Desk, наприклад Trello, Zendesk або Asana.

Також можна налаштувати сповіщення для запуску дій, що створює механізм для автоматичного усунення проблеми. Ця можливість є дуже потужною, але вимагає певного планування, і ви, ймовірно, не отримаєте цю можливість, доки не станете досвідченим користувачем системи NetCrunch. Сповіщення можна налаштувати та активувати за низкою умов.

Цей пакет підходить для будь-якого типу бізнесу. Він здатний контролювати віртуальні системи, такі як VMware або Hyper-V, а також стежитиме за часом відповіді ваших хмарних служб і підключення до них. Простота використання цього пакету робить його особливо привабливим для малих підприємств і стартапів, у яких може не бути великої групи технічної підтримки.

Переваги:

– Автоматизований моніторинг системи: сповіщення усувають необхідність активно стежити за продуктивністю системи

– Можливість спостереження для гібридних середовищ: моніторинг хмарних систем, а також локальних активів

– Автоматизація реагування: надсилайте сповіщення у свій інструмент Service Desk і налаштовуйте автоматичні відповіді, які ініціюються сповіщеннями

Недоліки:

– Доступно лише для Windows Server: немає версії для Linux і SaaS

Існує три версії пакета NetCrunch, що робить систему придатною для компаній будь-якого розміру. Є Essentials, який забезпечує моніторинг пристроїв на основі SNMP, Professional, який додає моніторинг серверів і віртуальних

машин, і Enterprise, який також забезпечує моніторинг трафіку та відстеження хмарних служб. Ви можете оцінити AdRem NetCrunch за допомогою 30-денної безкоштовної пробної версії.

13. Прогрес WhatsUp Gold

Progress WhatsUp Gold – це інструмент моніторингу мережевих пристроїв, який надає підприємствам просте у використанні рішення для нагляду за мережевою інфраструктурою. Він забезпечує видимість продуктивності мережі в режимі реального часу завдяки розгортанню процесів SNMP. Інструмент постійно відстежує різноманітні пристрої, сервери та додатки, пропонуючи інформацію про стан пристроїв, що має вирішальне значення для підтримки безвідмовної роботи мережі та оптимальної продуктивності.

Основні характеристики:

– Моніторинг у реальному часі: безперервний моніторинг мережевих пристроїв у реальному часі, що забезпечує негайне виявлення проблем з продуктивністю.

– Настроювані інформаційні панелі: користувачі можуть створювати персоналізовані інформаційні панелі для відображення ключових показників.

– Автоматичне виявлення пристроїв: Виявляє та відображає пристрої в мережі.

Progress WhatsUp Gold надає детальні інформаційні панелі з можливістю налаштування, які дають користувачам можливість миттєво відстежувати стан усієї мережі. Це дозволяє створювати детальні звіти та сповіщення, які допомагають командам залишатися проактивними, сповіщаючи їх про потенційні проблеми, такі як збої в роботі пристроїв або зниження продуктивності.

Базовий пакет WhatsUp Gold налаштовується за допомогою процесу автоматичного виявлення. Це документує та відображає всі пристрої, а потім негайно починає безперервний моніторинг. Служба високоавтоматизована, і користувачам не потрібно буде звертати увагу на консоль, якщо не виникне

проблема. Інструмент попереджатиме та надсилатиме сповіщення, коли потрібні технічні спеціалісти.

Після того, як цей основний блок запрацює, покупці можуть розглянути інші блоки, які пропонує Progres. Ці інші модулі підключаються до основної консолі та додають моніторинг мережевого трафіку, серверів, програм і віртуальних систем.

Цей пакет ідеально підходить для мережевих адміністраторів та IT-фахівців у компаніях будь-якого розміру, пропонуючи повний моніторинг мережі, виявлення проблем у реальному часі та настроювані інформаційні панелі. Він підходить організаціям зі складною багатосайтовою інфраструктурою, забезпечуючи проактивний моніторинг, автоматичні сповіщення, детальну звітність та інтеграцію сторонніх розробників для ефективного управління IT.

Переваги:

– Розгортає простий протокол керування мережею: ця галузева стандартна система звітності вже встановлена на всіх ваших мережевих пристроях.

– Комплексна система сповіщень: інструмент надсилає сповіщення про збої в мережі, проблеми з продуктивністю пристрою або перевищення порогових значень.

– Підходить для змішаних мереж: відстежує бездротові мережі, а також локальні мережі.

Недоліки:

– Без хмарної версії: це локальний пакет для Windows Server.

Progress WhatsUp Gold працює на Windows Server, і я знайшов його дуже простим у встановленні та використанні. Ви можете випробувати інструмент, отримавши доступ до 14-денної безкоштовної пробної версії.

14. ExtraHop Reveal(x)

Ключові статистичні дані та час обробки помітно відображаються в цифровому вигляді. ExtraHop Reveal(x) – це система моніторингу безпеки мережі, здатна блокувати виявлені нею загрози. Цей інструмент використовує штучний

інтелект для виявлення нелогічної або нестандартної поведінки, яка може вказувати на захоплення облікового запису або внутрішню загрозу. Служба використовує метод виявлення аномалій.

Основні характеристики:

– Аналітика поведінки користувачів і суб'єктів (UEBA): включає в себе можливості UEBA для аналізу моделей і поведінки, покращуючи здатність виявляти аномалії та потенційні загрози безпеці.

– Автоматичне виявлення мережі: використовує автоматичне виявлення мережі, щоб спростити процес початкового налаштування та забезпечити оновлення системи моніторингу в міру розвитку мережі.

– Ідентифікує шахрайські пристрої: пропонує функції, які спеціально ідентифікують шахрайські пристрої, сприяючи безпеці мережі, вирішуючи потенційні загрози в режимі реального часу.

ExtraHop Reveal(x) забезпечує постійно оновлюваний мережевий інвентар і негайно виявляє неавторизовані пристрої в мережі. Він також використовує виявлення аномалій для виявлення вторгнень і внутрішніх загроз. Пакет забезпечує важливе виявлення мережі та відповідь (NDR).

Під час тестування сервіс ExtraHop налаштувався шляхом сканування мережі та створення списку всіх підключених пристроїв. Потім інструмент працював як сніффер пакетів і аналізатор протоколів, класифікуючи весь трафік у мережі. Аналітика поведінки користувачів і суб'єктів (UEBA) реєструвала показники активності для кожного облікового запису користувача та джерела IP.

Механізми реагування в Reveal(x) можна налаштувати на автоматичний запуск. Вони використовують оркестровку для припинення загроз шляхом координації зі сторонніми системами. Платформа ExtraHop також доступна у версії для захисту хмарних систем. ExtraHop Reveal(x), ймовірно, надто складний для потреб малого бізнесу. Однак середні та великі компанії виграють від використання цієї системи моніторингу безпеки мережі.

Переваги:

– Операції на кількох сайтах: підтримує роботу на кількох сайтах, що дозволяє організаціям із розподіленою інфраструктурою ефективно контролювати свою мережу та керувати нею.

Недоліки:

– Відсутність прайс-листа: відсутність загальнодоступного прайс-листа може стати проблемою для організацій, які прагнуть до прозорості ціноутворення.

ExtraHop Reveal(x) пропонується як платформа SaaS. Також можливе придбання системи як мережевого пристрою. Ви можете перевірити ExtraHop, перейшовши до демо-версії.

– Ці демонстрації:

– Демо

– Демонстрація для самостійного керування

– Демонстрація в прямому ефірі

15. AKIPS

AKIPS – це пакет моніторингу мережі, який забезпечує як перевірку стану пристрою, так і моніторинг трафіку. Система включає службу виявлення, яка генерує інвентаризацію мережі. Кожен запис у списках інвентаризації забезпечує доступ до деталей пристрою за клацанням миші. Цей детальний екран також показує всі останні події на пристрої.

Основні характеристики:

– Виявлення мережі: створює інвентаризацію системи

– Статуси пристроїв: безперервний моніторинг за допомогою SNMP

– Аналіз трафіку: витягує дані NetFlow із комутаторів

Система AKIPS працює на FreeBSD. Вам не потрібно запускати цю операційну систему, щоб використовувати AKIPS, оскільки інсталяційний образ інструменту встановлюється у віртуальну машину та встановлює операційну систему FreeBSD перед програмним забезпеченням моніторингу мережі. Таким

чином, ви можете запустити цей інструмент у будь-якій операційній системі, яка підтримує віртуальну машину, такій як Hyper-V або VMware.

AKIPS поєднує моніторинг мережевих пристроїв та аналіз мережевого трафіку. Це дві функції, які більшість постачальників систем моніторингу розділяють на два окремі блоки. Таким чином, з однією покупкою ви отримуєте всі засоби моніторингу, необхідні для запобігання проблемам та підтримки працездатності вашої мережі.

Інформаційна панель пристрою використовує SNMP і Ping для перевірки кожного пристрою. Це дає змогу системі перевіряти доступність, тому що якщо пристрій переходить у мережу, він перестає існувати в системі SNMP. Якщо пристрій має проблеми, але все ще працює, консоль AKIPS отримає повідомлення SNMP Trap. Це інтерпретується як сповіщення. Пакет може пересилати сповіщення як сповіщення від Teams, Slack, PagerDuty, ServiceNow або Opsgenie.

Система AKIPS також подає сповіщення, якщо раніше виявлений пристрій зникає. Можна визначити порогові значення продуктивності за такими факторами, як пропускна здатність трафіку. Так, наприклад, якщо рівень трафіку на каналі раптово падає або досягає повної потужності інтерфейсу комутатора, до якого він підключається, система створить сповіщення. Ці умови детально описано на інформаційній панелі подій консолі AKIPS.

AKIPS надає інформацію про мережу, яка може допомогти з пропускною здатністю. Ця категорія даних включає інформацію про невикористані інтерфейси або моделі трафіку за посиланням протягом певного часу. Система може пересилати свої дані як журнали до систем SIEM для аналізу активності.

AKIPS не публікує свій прайс-лист, що ускладнює прийняття рішення щодо його придатності для малого бізнесу. Пакет не включає систему для перевірки інтернет-посилань на хмарні платформи або між сайтами, тому це пакет керування локальною мережею. Сервіс ідеально підходить для моніторингу однієї мережі.

Переваги:

– Повна видимість мережі: комбінований моніторинг мережевих пристроїв та трафіку

– Сповіщення про проблеми з пристроєм: нестача ресурсів і зниження продуктивності

– Пересилання журналів до інструментів SIEM: добре для пошуку загроз

Недоліки:

– На сайті немає загального прайс-листа: вам потрібно отримати індивідуальну ціну

Система AKIPS встановлюється у віртуальну машину, що означає, що вона може працювати на будь-якій операційній системі. У межах гіпервізора програмне забезпечення для моніторингу мережі фактично працює на FreeBSD Unix. Однак ця операційна система включена в інсталяційний пакет. Ви можете отримати демо-версію, щоб побачити, як працює система, а потім отримати доступ до пакета самостійно за допомогою 30-денної безкоштовної пробної версії.

16. SuperOps

SuperOps – це хмарна платформа, яка пропонує інструменти керування системою, наприклад пакет дистанційного моніторингу та керування. Цей RMM містить блок моніторингу мережі, який забезпечує виявлення мережі. Служба генерує інвентаризацію системи, яка є основою для автоматизованих процедур моніторингу.

Основні характеристики:

– Розроблено для постачальників керованих послуг: має мультитенантну архітектуру

– Виявлення мережі: ідентифікує всі пристрої, які створюють мережу

– Карта топології мережі: показує, як пристрої з’єднуються разом

Цей пакет має архітектуру з кількома клієнтами, що означає, що MSP можуть створювати субрахунки на платформі. Кожен субрахунок буде містити

дані одного клієнта MSP. Таким чином, MSP може зберігати дані для кожного клієнта повністю окремо.

Система SuperOps – хороший вибір для нашого списку, оскільки вона пропонує відповідне рішення для постачальників керованих послуг. Абоненти можуть вибрати лише пакет RMM або підписатися на один із двох пакетів, які поєднують функції RMM із функціями PSA. Системи PSA забезпечують системи, необхідні менеджерам MSP для ведення свого бізнесу.

RMM включає зведення всіх пристроїв у мережі, і кожен запис у цьому списку містить посилання на екран деталей. Сервіс забезпечує автоматичний моніторинг, оскільки він попереджає, якщо мережевий сканер виявляє проблему з мережевим пристроєм. Це означає, що технікам не потрібно дивитися на консоль моніторингу, щоб чекати проблем. RMM розроблено для того, щоб один технік MSP міг наглядати за системою кількох підприємств. Ця ефективність має важливе значення, оскільки це ключова функція, яка дозволяє MSP скорочувати внутрішні IT-відділи та заохочувати компанії до аутсорсингу управління своїми IT-активами. Служба моніторингу мережі не лише відстежує комутатори, маршрутизатори та брандмауери; він також містить список кінцевих точок, таких як сервери та робочі станції. Служба моніторингу мережі формує основу для функцій керування кінцевими точками, які включають керування ліцензіями на програмне забезпечення та встановлення виправлень.

Система SuperOps ідеально підходить для постачальників керованих послуг. MSP, ймовірно, візьмуть один із комбінованих пакетів, які включають системи RMM і PSA. Однак SuperOps також пропонує пакет лише для RMM. Це видання підходить для використання IT-відділами, яким потрібні інструменти підтримки системи в системі RMM, але не програмне забезпечення для керування MSP, яке надає PSA.

Переваги:

– Хмарна система: до консолі можна отримати доступ з будь-якого місця через будь-який стандартний веб-браузер

– Кілька аудиторій: RMM підходить як для IT-відділів, так і для MSP

– Спостереженість повного стека: RMM забезпечує моніторинг усіх IT-активів, а не лише мереж

Недоліки:

– Немає локального пакета: ви не можете завантажити програмне забезпечення та запустити RMM самостійно

SuperOps доступний у чотирьох версіях: лише RMM, лише PSA, Pro Unified Basic і Super Unified Advanced. Ви можете випробувати пакет RMM із блоком моніторингу мережі, отримавши доступ до 14-денної безкоштовної пробної версії.

17. Монітор продуктивності мережі SolarWinds

Головна інформаційна панель відстежує доступність і продуктивність підключених мережевих пристроїв із цілісної точки зору. SolarWinds Network Performance Monitor – це комплексний інструмент моніторингу продуктивності мережі, який може відстежувати стан пристроїв за допомогою SNMP. Він може автоматично виявляти мережеві пристрої, підключені до вашої мережі.

Під час тестування Network Performance Monitor ми виявили наступні ключові функції та унікальну функцію порівняно з іншими оглядами в цій публікації.

Основні характеристики:

– Моніторинг SNMP: моніторинг SNMP дозволяє збирати дані про продуктивність мережевих пристроїв, які підтримують SNMP.

– Автоматичне виявлення: функція автоматично виявляє та ідентифікує підключені мережеві пристрої, спрощуючи процес додавання та керування пристроями в мережі.

– Аналіз мережевих пакетів: аналіз мережевих пакетів дозволяє користувачам аналізувати пакети, що проходять через мережу, для усунення несправностей і оптимізації продуктивності.

Унікальною особливістю SolarWinds є інструмент NetPath. Це схоже на функцію TraceRoute, доступну в інших інструментах у цьому списку, наприклад PRTG і Site24x7. Однак NetPath показує маршрут як візуальне представлення.

Монітор продуктивності мережі SolarWinds – це автоматизована система відстеження, яка ідентифікує всі пристрої, підключені до мережі, відображає їх і спостерігає за проблемами продуктивності. Ця система сповістить вас про проблеми зі збором і надішле сповіщення. Поряд із надійним скануванням уразливостей і розширеними параметрами для створення та моніторингу політик, це безперечно найкращий вибір для систем моніторингу мережі.

Будь-які виявлені пристрої, програми чи служби також можна переглянути на карті топології мережі, де ви можете побачити, як ваша інфраструктура пов'язана разом. Функція NetPath дозволяє відстежувати передачу пакетів поетапно, що може допомогти ефективніше діагностувати джерело проблем із продуктивністю мережі. Спеціальна система сповіщень дає змогу встановлювати умови запуску сповіщень. Після виконання умов запуску програмне забезпечення надішле вам сповіщення електронною поштою або SMS, щоб повідомити про подію. Користувач може переглянути повний список сповіщень відповідно до серйозності, перейшовши на сторінку «Усі активні сповіщення».

Я був вражений тим, як цей інструмент автоматизує моніторинг мережі – він навіть налаштовується самостійно, тому підходить для підприємств будь-якого розміру, оскільки для роботи не потрібні спеціальні знання з керування мережею. Сповіщення означають, що ви можете вважати, що все працює нормально, якщо ви не отримаєте сповіщення, тож ви можете витратити свій час на інші проблеми та дозволити NPM подбати про мережу. Наше тестування виявило наступні переваги та недоліки NPM.

Переваги:

– Автоматичне виявлення: підтримка автоматичного виявлення дозволяє створювати карти топології мережі та списки інвентаризації в реальному часі, коли пристрої входять у мережу, що спрощує керування мережею.

– Моніторинг SNMP і аналіз пакетів: це надає користувачам комплексний контроль над можливостями моніторингу, що робить його універсальним порівняно з аналогічними інструментами.

– Настроювана інформаційна панель: SolarWinds NPM дозволяє користувачам налаштовувати інформаційну панель за допомогою віджетів перетягування, створюючи персоналізований та інтуїтивно зрозумілий інтерфейс користувача.

Недоліки:

– Може бути непосильним для невеликих мереж: враховуючи його багатофункціональний характер і зосередженість на функціональності корпоративного рівня, невеликі локальні мережі можуть вважати це надто великим або більшим, ніж вони потребують для своїх конкретних потреб.

Ціна на SolarWinds Network Performance Monitor починається від 2995 доларів США (~2350 фунтів стерлінгів). Ви можете завантажити безкоштовну пробну версію.

18. Nagios Core

Для тих, хто віддає перевагу стовпчикові діаграми та сповіщення пристроїв із кольоровим кодуванням Nagios Core – це мережевий монітор із відкритим кодом, який має веб-інтерфейс для моніторингу продуктивності мережі. За допомогою мережевого інтерфейсу користувача ви можете відстежувати поточний стан мережі за допомогою загальних показників статусу хоста та загальних показників стану мережевих послуг у верхній частині сторінки. Графічний інтерфейс має кольорове кодування, щоб ви могли легко бачити недоступні або скомпрометовані елементи.

Основні характеристики:

– Інформаційна панель продуктивності: надає інформаційну панель продуктивності для моніторингу в реальному часі та візуалізації ключових показників, пов’язаних із продуктивністю мережі та системи.

– Планування потужностей: містить функції планування потужностей, які допомагають користувачам оцінювати ресурси та можливості своєї IT-інфраструктури та керувати ними.

– Плагіни, створені спільнотою: дозволяє використовувати плагіни, створені спільнотою, розширюючи функціональні можливості Nagios Core, надаючи додаткові можливості моніторингу.

Nagios Core – це один із найнадійніших і найпопулярніших пакетів моніторингу IT-активів із відкритим вихідним кодом, який відстежуватиме мережі, програми та сервери. Ця версія Nagios є безкоштовною для використання, але ви також повинні розглянути платну версію Nagios XI. Ми мали змогу відстежувати події продуктивності за допомогою системи сповіщень, яка надсилала сповіщення електронною поштою та SMS. Ви можете переглянути розділ «Історія сповіщень», щоб побачити, які сповіщення було створено та коли. Список сповіщень також позначено кольором, що полегшує визначення пріоритетності критичних сповіщень. Ми змогли використати API для інтеграції інших мережевих служб. Якщо вам потрібні додаткові функції, ви також можете перевірити тисячі плагінів спільноти, доступних на Nagios Exchange, щоб допомогти вам додати додаткові функції. Наприклад, існує плагін перевірки навантаження ЦП SNMP, який дозволяє відстежувати навантаження SNMP або використання ЦП вашого мережевого пристрою. Nagios Core не має належного інтерфейсу користувача, але має дуже гнучкий і потужний механізм виявлення та збору даних. Таким чином, це хороший варіант, якщо ви хочете створити власну програму моніторингу з безкоштовним інструментом представлення даних, таким як Kibana або Prometheus, і надсилати дані з цього інструменту.

Переваги:

– Прозорий інструмент із відкритим кодом: Nagios Core – це інструмент із відкритим кодом, який забезпечує прозорість і дозволяє користувачам отримувати доступ до вихідного коду та змінювати його за потреби.

– Надійний сервер API: Надійний сервер API робить Nagios Core придатним для розробників, які хочуть інтегрувати свої програми або сценарії з системою моніторингу.

Недоліки:

– Підтримка версії з відкритим вихідним кодом: версії з відкритим вихідним кодом може бути недостатньо рівня підтримки, який є в платних продуктах, що може вимагати від користувачів покладатися на форуми та ресурси спільноти для отримання допомоги.

– Технічна та складна інсталяція: інсталяція може бути технічною та складною, особливо для користувачів без великого досвіду системного адміністрування.

Nagios Core – чудове безкоштовне програмне забезпечення для моніторингу мережі, однак налаштування може бути трохи більш практичним, ніж інші продукти в цьому списку. Завантажте Nagios Core безкоштовно або, залежно від ваших потреб, подивіться його порівняння з Nagios XI. Обидві версії пакета працюють на RHEL, Ubuntu, CentOS і Debian Linux. Програмне забезпечення працюватиме в Windows через гіпервізор VMWare або Hyper-V.

19. Досвід мережі Catchpoint

Інформаційна панель з чітким макетом, яку легко деталізувати. Catchpoint Network Experience є одним із чотирьох модулів моніторингу продуктивності на хмарній платформі Catchpoint. Метою системи Catchpoint є забезпечення успішної доставки веб-додатків. Платформа пропонується як окремий пакет, а не як окремі модулі.

Основні характеристики:

– Моніторинг продуктивності віртуальної мережі: дозволяє контролювати продуктивність віртуальної мережі, забезпечуючи видимість продуктивності віртуалізованої інфраструктури.

– Оцінки BGP: виконує оцінки протоколу прикордонного шлюзу (BGP) для оцінки інформації про маршрутизацію та шляхів через Інтернет.

– Ефективність доставки веб-сайту: ця функція дозволяє організаціям відстежувати та оптимізувати роботу кінцевих користувачів.

Catchpoint Network Experience призначений для використання як частина набору інструментів, який забезпечує успішну доставку веб-сторінок та інших веб-додатків. Тому цей інструмент чудово підходить для перевірки успішної доставки веб-систем. Він є частиною ланцюжка моніторингу, який може швидко визначити першопричину проблем із продуктивністю.

Інші модулі платформи Catchpoint – це монітор додатків і інструменти реального користувача та синтетичні засоби моніторингу веб-додатків. Мережева служба перевіряє ефективність підключень до різних серверів, які сприяють успішній доставці веб-сторінки та її вмісту. Ці інші сервери включають мережі доставки контенту (CDN), DNS-сервери та безсерверні хости мікросервісів, які запускаються під час взаємодії користувача на веб-сторінці.

Система Network Experience також аналізує діяльність віртуальних мереж, таких як Secured Access Service Edge (SASE) і SD-WAN. Він визначить проблеми з продуктивністю віддаленого доступу та мереж VPN «сайт-сайт», а потім відстежить через локальну мережу веб-сервери, на яких розміщено веб-сайти компанії.

Повноцінна платформа Catchpoint розроблена спеціально для відстеження продуктивності веб-сайтів, тому вона не є гарним вибором для підприємств, яким потрібна служба моніторингу локальної мережі. Підприємства, які розміщують власні сайти, будуть найбільш прийнятними клієнтами для системи Catchpoint. Однак ті компанії, які використовують послуги хостингу та CDN для оптимізації доставки, також можуть скористатися послугою Catchpoint.

Переваги:

– Аналіз першопричини: пропонує можливості аналізу першопричини для виявлення та вирішення основних проблем, що спричиняють проблеми з продуктивністю.

– Оцінює продуктивність ISP: Інтернет-провайдери та інші служби для оцінки їх продуктивності та надійності.

Недоліки:

– Не підходить для моніторингу локальної мережі: Catchpoint може не підходити для моніторингу локальної мережі, що може обмежити його застосування для організацій, які в основному зосереджені на продуктивності внутрішньої мережі.

Хоча Catchpoint є пакетом SaaS, він має локальний елемент, який називається Enterprise Node. Він встановлюється на одному з ваших серверів і керує моніторингом вашого веб-хосту та мережевих з'єднань між ним і зовнішнім світом. Цей агент працює на RHEL і CentOS Linux. Його також можна запускати як віртуальний пристрій через Hyper-V або VMware. Catchpoint доступний для 14-денної безкоштовної пробної версії.

20. Icinga

Мережеві сповіщення та навантажені пристрої відображаються на помітному місці. Icinga – це інструмент моніторингу мережі з відкритим кодом, який відстежує продуктивність вашої мережі, хмарного сервісу та центру обробки даних. Програмне забезпечення є веб-інтерфейсом і може бути налаштовано за допомогою графічного інтерфейсу користувача або за допомогою доменної мови (DSL). Вибір між двома дає вам можливість контролювати, як завгодно.

Основні характеристики:

– Графічний веб-інтерфейс: надає веб-графічний інтерфейс користувача (GUI) для зручного налаштування та моніторингу.

– Доступна конфігурація DSL: пропонує параметри конфігурації доменно-специфічної мови (DSL), що дозволяє користувачам налаштовувати Icinga за допомогою спеціальної мови сценаріїв.

– Модулі/розширення Icinga: підтримує модулі та розширення, розширюючи функціональні можливості базової інсталяції для задоволення різних потреб моніторингу.

– Вбудована візуальна звітність: містить вбудовані функції візуальної звітності, що покращує здатність аналізувати та інтерпретувати дані моніторингу.

Icinga була розроблена як вдосконалення Nagios, і це дає цій системі велику перевагу перед багатьма іншими інструментами в цьому списку. Він абсолютно безкоштовний у використанні, як і Nagios Core, але, як і Nagios XI, він має корисний графічний інтерфейс користувача, який постачається з попередньо написаними екранами, але також можна налаштувати. Icinga все ще сумісна з плагінами Nagios, тому користувачі цієї системи можуть покращити інструмент моніторингу від Nagios Exchange, а також плагінів від набагато меншої Icinga Exchange. Нам подобається, що Icinga – це безкоштовний пакет моніторингу з відкритим кодом для мереж, програм і серверів. Його розробила команда творців Nagios, які були незадоволені напрямком еволюції цієї системи. Ви можете увійти в графічний інтерфейс і використовувати інформаційну панель, щоб переглянути огляд управління ефективністю. Інформаційна панель показує, чи є проблеми з продуктивністю або доступністю, і позначає їх кольором відповідно до серйозності. Критичні або непрацюючі мережеві пристрої позначені червоним кольором. Розширення або модулі Icinga дозволяють додавати додаткові функції до програми. Наприклад, модуль Icinga для vSphere дозволяє контролювати віртуальні машини та хост-системи. Існує ряд плагінів, створених спільнотою, які можна безкоштовно завантажити з Icinga Exchange. Оскільки Icinga є відкритим кодом, її можна переписати, а її сумісність із Nagios означає, що її можна розширити за допомогою безкоштовної бібліотеки плагінів цієї конкуруючої системи. Ці функції роблять його чудовим для мережевих інженерів, які хочуть створити індивідуальний інструмент моніторингу мережі.

Переваги:

– Модулі для різних функціональних можливостей: використовуються модулі для додавання різних функціональних можливостей, зберігаючи базову інсталяцію легкою, дозволяючи користувачам налаштовувати своє середовище моніторингу.

– Сумісність із різними платформами: може працювати в операційних системах Linux і Windows, забезпечуючи гнучкість у розгортанні.

Недоліки:

– Покладення на спільноту з відкритим кодом: покладається на спільноту з відкритим кодом для підтримки та оновлень, що може призвести до змін у рівні підтримки та часу відповіді.

– Розроблено для технічних користувачів: це може вимагати крутішої кривої навчання для тих, хто менш знайомий із передовими концепціями моніторингу. Інші варіанти можуть пропонувати кращі стандартні функції для менш технічних користувачів.

Загалом, Icinga – це масштабоване рішення, яке дає вам можливість контролювати те, як ви керуєте своїм середовищем. Icinga доступний для Debian, Red Hat, SUSE, Ubuntu, Fedora, OpenSUSE, Raspbian і Windows. Ви можете завантажити безкоштовну пробну версію.

21. Zabbix

Більш загальні візуальні елементи та менші можливості налаштування, ніж інші інструменти в цьому списку. Zabbix – це безкоштовний інструмент моніторингу мережі з відкритим вихідним кодом, який поєднує моніторинг мережі, сервера, хмари, програми та сервісів в одне єдине рішення. Zabbix використовує SNMP та IPMP для моніторингу вашої мережі. Функція автоматичного виявлення автоматично знаходить мережеві пристрої та додає їх для моніторингу. Інструмент також може автоматично виявляти зміни конфігурації, щоб ви могли визначити, чи було оновлено мережевий пристрій.

Основні характеристики:

– Функція автоматичного виявлення: містить функцію автоматичного виявлення, яка автоматично визначає та додає нові пристрої до системи моніторингу.

– Моніторинг SNMP та IPMP: підтримує як SNMP, так і IPMP (Internet Protocol Multipathing) для комплексного моніторингу мережі.

– Моніторинг додатків/серверів: надає можливості моніторингу додатків і серверів, дозволяючи користувачам контролювати продуктивність і доступність додатків.

– Шаблони моніторингу мережі: використовує шаблони моніторингу мережі для ефективного та стандартизованого налаштування моніторингу.

– Система сповіщень: реалізує систему сповіщень для сповіщення адміністраторів про проблеми чи події, сприяючи своєчасному реагуванню.

Zabbix – це безкоштовний інструмент, який можна розширити за допомогою шаблонів. Ця служба може контролювати мережі, сервери та програми, а також використовує SNMP для виявлення та моніторингу мережевих пристроїв. Шаблони моніторингу мережі дозволяють легко налаштувати програму для моніторингу. Наприклад, існують попередньо налаштовані шаблони для таких постачальників, як Cisco, Hewlett Packard Enterprise, Dell, Intel, Netgear тощо. Якщо вам потрібна більша різноманітність, ви можете отримати доступ до ще більшої кількості шаблонів спільноти на сайті спільноти Zabbix. Щоб гарантувати, що ви можете вчасно відповісти на сповіщення, Zabbix має систему сповіщень. Сповіщення надсилаються електронною поштою, SMS, месенджером, jabber або спеціальними сценаріями, щоб інформувати вас про нові події безпеки мережі. Вміст сповіщень можна налаштувати відповідно до одержувача, щоб переконатися, що точка контакту має все необхідне для інтелектуального повторного відкриття. Zabbix – це локальний пакет на базі Linux, і його конкурентна перевага порівняно з багатьма інструментами в цьому списку полягає в тому, що він вільний у використанні. Екрани для цієї системи містять чудові функції візуалізації даних, і їх можна налаштувати, що робить це одним із найпростіших інструментів для адаптації без потреби в технічних навичках.

Переваги:

– Прозорий інструмент із відкритим вихідним кодом: Zabbix – це інструмент із відкритим кодом, який забезпечує прозорість кодової бази та

дозволяє користувачам налаштовувати та розширювати його функціональні можливості.

– Миттєве виявлення пристроїв та змін: Може миттєво виявляти нові пристрої та зміни конфігурації, забезпечуючи точний та актуальний моніторинг.

– Корисні шаблони для швидкого аналізу: пропонує готові шаблони для швидкого налаштування та аналізу, спрощуючи процес налаштування моніторингу.

Недоліки:

– Інтерфейс не настільки інтуїтивно зрозумілий: інтерфейс може бути не таким інтуїтивно зрозумілим, як деякі інші рішення, потенційно вимагаючи від користувачів витрачати більше часу на вивчення системи.

– Бажання покращити функції сповіщень: деякі користувачі висловили бажання покращити функції сповіщень, зокрема щодо зменшення помилкових спрацьовувань.

Zabbix є одним із найкращих інструментів моніторингу мережі Linux на ринку. Автоматичне виявлення та шаблони роблять програму простою для розгортання. Програмне забезпечення для пакета встановлено на Linux, macOS і Unix. Хоча основний сервер для Zabbix недоступний для Windows, існує програма-агент, яка дозволяє контролювати кінцеві точки Windows через мережу. Ви можете завантажити програмне забезпечення безкоштовно.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох

сферах на більшості платформ. Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Як вибрати правильний інструмент моніторингу корпоративної мережі

Розмір і складність мережі

Вибираючи інструмент моніторингу мережі, враховуйте розмір і складність мережі. Деякі інструменти призначені для малих і середніх мереж, тоді як інші можуть працювати з великими складними інфраструктурами. Переконайтеся, що вибраний вами інструмент може масштабуватися з вашою мережею в міру її зростання.

Оцініть здатність інструменту відстежувати різні пристрої та програми у вашій мережі. Хороший інструмент моніторингу мережі повинен забезпечувати повне покриття та адаптуватися до змін у топології мережі.

Бюджет

Бюджет є критичним фактором у виборі інструменту моніторингу мережі. У той час як деякі інструменти пропонують широкі можливості за високою ціною, інші надають важливі можливості моніторингу за нижчою ціною. Визначте свій бюджет і визначте пріоритетність функцій, які є найважливішими для вашої організації.

Розглянемо загальну вартість володіння, включаючи ліцензійні збори, технічне обслуговування та потенційні витрати на навчання. Деякі інструменти можуть пропонувати безкоштовні пробні версії або багаторівневі тарифні плани, що дозволяє перевірити можливості інструменту перед покупкою.

Простота використання

Зручність використання інструменту моніторингу мережі може значно вплинути на його ефективність. Складний інструмент із крутою кривою навчання

може не підходити для всіх організацій. Шукайте інструменти з інтуїтивно зрозумілим інтерфейсом, чіткою документацією та надійною підтримкою клієнтів. Простота використання також поширюється на процес налаштування та налаштування інструменту. Інструмент, який можна швидко розгорнути та інтегрувати в існуючу інфраструктуру, заощадить ваш час і ресурси.

Інтеграція

Інтеграція з іншими інструментами та системами є важливою для безперебійного моніторингу мережі. Переконайтеся, що вибраний вами інструмент може інтегруватися з наявною IT-інфраструктурою, включаючи системи продажу квитків, інструменти безпеки та хмарні служби. Сумісність із галузевими стандартами та протоколами також має вирішальне значення. Інструмент, який підтримує SNMP, ICMP та інші поширені протоколи, забезпечить більш повні можливості моніторингу.

Підтримка та спільнота

Надійна підтримка клієнтів і активна спільнота користувачів можуть значно змінити ваш досвід моніторингу мережі. Виберіть інструмент, який пропонує швидку підтримку та доступ до великої кількості ресурсів, таких як форуми, бази знань і навчальні матеріали. Спільнота активних користувачів може надати цінну інформацію, поради та поради щодо усунення несправностей. Взаємодія з іншими користувачами може допомогти вам максимально використати потенціал вашого інструменту моніторингу мережі.

Інструменти моніторингу мережі незамінні; вони забезпечують видимість і розуміння, необхідні для підтримки здорової та безпечної мережі. Розуміючи важливість цих інструментів і враховуючи такі фактори, як розмір мережі, бюджет, простота використання, інтеграція та підтримка, ви можете вибрати правильний інструмент для своєї організації.

Інвестиції в надійний інструмент моніторингу мережі окупляться в довгостроковій перспективі. Ці інструменти допоможуть вам завчасно керувати мережею, зменшити час простою та забезпечити зручну роботу користувача.

Вікно в уподобання мережевого моніторингу в усьому світі показує конвергенцію бізнесу, технологій і суспільних змін. Оскільки нові технології, такі як генеративний штучний інтелект, виходять у центр уваги, залишається питання: як це вплине на моніторинг мережі?

1. Моніторинг мережі за допомогою AI

Штучний інтелект (AI) і машинне навчання (ML) змінюють наше уявлення про моніторинг мережі. Оскільки вони пропонують прогнозну аналітику, виявлення аномалій і швидке усунення несправностей, за допомогою штучного інтелекту та ML інструменти моніторингу мережі можуть проактивно виявляти проблеми, перш ніж вони вплинуть на бізнес-операції. Програмне забезпечення, яке розробляється у даній роботі, використовує прогнозний аналіз, який дозволяє легко виявляти аномалії в показниках, як-от час відповіді. Використовуючи сповіщення про порогові значення, IT-команди можуть виявляти ймовірні аномалії та швидко вирішувати проблеми для безперебійної роботи мережі.

2. Моніторинг IoT та периферійної мережі

Повсюдне поширення пристроїв IoT і периферійних обчислень змінює моніторинг мережі. У міру розширення Інтернету речей і периферійних мереж інструменти моніторингу повинні масштабуватися, щоб надавати детальну інформацію про ці середовища. Програмне забезпечення, яке розробляється у даній роботі, пропонує надійний моніторинг Інтернету речей і периферійних мереж, забезпечуючи видимість у реальному часі та відстеження продуктивності підключених пристроїв. Завдяки підтримці різноманітних протоколів, таких як SNMP і NetFlow, підприємства можуть залишатися попереду, відстежуючи такі важливі показники, як використання ЦП, трафік, уразливості вбудованого програмного забезпечення тощо. Мережі IoT також можна легко візуалізувати за допомогою мережевих карт.

3. Гібридний і багатохмарний моніторинг

Використання гібридних і багатохмарних середовищ зростає, оскільки організації розподіляють робоче навантаження між локальними, приватними та

публічними хмарами. Моніторинг цих розподілених інфраструктур вимагає надійних інструментів, які забезпечують централізовану видимість на різних платформах. Програмне забезпечення, яке розробляється у даній роботі, пропонує комплексний моніторинг гібридних і багатохмарних середовищ. Незалежно від того, чи це AWS, Azure, Google Cloud або локальні мережі, Програмне забезпечення, яке розробляється у даній роботі, забезпечує уніфіковану можливість спостереження, допомагаючи компаніям легко керувати різними інфраструктурами.

4. Автоматизація мережі

Керування мережею вручну застаріває. Автоматизація моніторингу мережі має вирішальне значення для масштабування IT-операцій, підвищення ефективності та зменшення людських помилок. Автоматизація дозволяє IT-командам зосереджуватися на стратегічних ініціативах, а не на повсякденних завданнях. Програмне забезпечення, яке розробляється у даній роботі, спрощують підключення пристроїв, встановлення порогових значень і створення звітів. Крім того, для ефективної автоматизації мережі підприємства можуть розгортати зміни конфігурації за допомогою конфіглетів. Вони також можуть бути ефективно використані для запобігання невідповідності.

5. Розширена інтеграція безпеки

Оскільки кіберзагрози стають все більш складними, інтеграція моніторингу мережі з інструментами безпеки більше не є обов'язковою. Організаціям потрібне рішення, яке не тільки відстежує продуктивність, але й допомагає виявляти та пом'якшувати загрози безпеці. Програмне забезпечення, яке розробляється у даній роботі, пропонує статистику в режимі реального часу щодо шаблонів підозрілого трафіку. Крім того, завдяки таким функціям, як керування вразливістю вбудованого програмного забезпечення та керування відповідністю мережі, платформа забезпечує додатковий рівень безпеки для захисту мереж.

6. ШІ та синтетичний мережевий трафік

Моніторинг мережі використовує синтетичний трафік для імітації поведінки користувача та завчасного виявлення потенційних проблем. Штучний інтелект відіграє важливу роль в аналізі синтетичних даних для надання корисної інформації. Програмне забезпечення, яке розробляється у даній роботі, використовує синтетичний моніторинг для імітації реальних умов мережі, виявлення аномалій і забезпечення сталої продуктивності. Аналіз, керований штучним інтелектом, розширює ці можливості, забезпечуючи практичну інформацію для IT-команд.

7. 5G-готовий моніторинг

Глобальне впровадження 5G створює нові проблеми для моніторингу мережі. Завдяки вищим швидкостям і меншій затримці інструменти моніторингу повинні адаптуватися до вимог мереж із підтримкою 5G. Програмне забезпечення, яке розробляється у даній роботі, готове до 5G із розширеними можливостями моніторингу, призначеними для обробки збільшених потоків даних і складності мереж 5G. Від моніторингу затримки до забезпечення безвідмовної роботи, Програмне забезпечення, яке розробляється у даній роботі, підтримує підприємства, які використовують технологію 5G.

8. Зосередьтеся на стійкості

Екологічні IT-ініціативи набирають обертів, а компанії прагнуть зменшити свій вуглецевий слід. Інструменти моніторингу мережі, які оптимізують споживання енергії, відіграватимуть вирішальну роль у досягненні цілей сталого розвитку. Програмне забезпечення, яке розробляється у даній роботі, підтримує сталість, надаючи інформацію від постачальників (наприклад, про температуру процесора, поточний рівень потужності, споживаний заряд батареї та енергоспоживання) щодо використання енергії мережевими пристроями. Підприємства можуть використовувати цю інформацію для оптимізації споживання енергії та досягнення своїх екологічних цілей.

9. Налаштовувані інформаційні панелі та звітність

У 2025 році персоналізація є ключовою. Підприємствам потрібні інструменти моніторингу, які надають спеціалізовану статистику та інформаційні панелі для конкретних ролей і потреб, що робить споживання даних більш ефективним. Програмне забезпечення, яке розробляється у даній роботі, дозволяє користувачам створювати налаштовувані інформаційні панелі та звіти, що дозволяє ІТ-командам зосередитися на найважливіших показниках. За допомогою готових шаблонів і віджетів, які можна перетягнути, користувачі можуть створювати інформаційні панелі, які відповідають їхнім робочим процесам.

10. Централізована спостережливість

Централізована можливість спостереження стає головним пріоритетом, оскільки компанії прагнуть керувати та контролювати мережі, програми, сервери та хмарні ресурси з єдиної платформи. Спрощення ІТ-операцій завдяки централізації підвищує ефективність і знижує витрати. Програмне забезпечення, яке розробляється у даній роботі, – це уніфікована платформа спостереження, яка забезпечує наскрізну видимість у ІТ-середовищі. Від мережевих пристроїв до програм, Програмне забезпечення, яке розробляється у даній роботі, пропонує цілісне уявлення для спрощення моніторингу.

11. Зосередьтеся на досвіді користувача

Сучасні інструменти моніторингу мережі віддають перевагу зручності використання. ІТ-команди очікують інтуїтивно зрозумілих інтерфейсів, легкої конфігурації та практичних ідей без крутої кривої навчання. Інтуїтивно зрозумілий інтерфейс і керований процес налаштування програмного забезпечення, яке розробляється у даній роботі, роблять його доступним для ІТ-команд з будь-яким рівнем кваліфікації. Чітка візуалізація інструменту та інтелектуальне розуміння забезпечують безперебійний моніторинг.

12. Програмно-визначена мережа

Оскільки мережі стають більш динамічними та віртуалізованими, програмно-визначена мережа (SDN) стимулює зміни в управлінні мережею.

Інструменти моніторингу повинні адаптуватися, щоб забезпечити видимість і контроль над середовищами SDN. Програмне забезпечення, яке розробляється у даній роботі, обладнано для моніторингу середовищ SDN і SD-WAN, пропонуючи глибоке розуміння віртуалізованих мережевих компонентів. Це гарантує, що підприємства можуть ефективно керувати сучасною мережевою архітектурою.

13. Розповсюдження пристроїв впливає на моніторинг мережі

Швидке розширення пристроїв у корпоративних мережах, від IoT до BYOD, збільшує складність моніторингу. Інструменти повинні масштабуватися, щоб ефективно керувати цією зростаючою різноманітністю. Програмне забезпечення, яке розробляється у даній роботі, спрощує керування розповсюдженням пристроїв, пропонуючи автоматичне виявлення мережі та можливості моніторингу для різноманітних пристроїв. Це забезпечує безперебійну продуктивність у зростаючих IT-середовищах.

14. Рішення SaaS для управління мережею

Хмарні рішення SaaS домінують в управлінні мережею завдяки їх масштабованості, простоті розгортання та економічній ефективності.

Як платформа на основі SaaS, програмне забезпечення, яке розробляється у даній роботі, пропонує неперевершену масштабованість і доступність. Його дружня до хмар архітектура гарантує, що підприємства можуть ефективно керувати своїми мережами, незалежно від розміру чи складності.

3.2 Розробка структурної схеми

Існує два типи систем моніторингу мережі: монітори мережевих пристроїв і монітори мережевого трафіку. У цьому огляді ми зосередимося на інструментах моніторингу пристроїв, які часто називають «моніторами продуктивності мережі». Ці пакети покладаються на технологію під назвою Simple Network Management Protocol (SNMP).

Система SNMP включає центральний блок, який називається SNMP Manage r, і блок, який є резидентним на кожному мережевому пристрої, який називається агентом пристрою. Усі мережеві пристрої постачаються з уже встановленим агентом пристрою. Операційні системи, такі як Windows, Linux і macOS, також мають вбудовані агенти SNMP – їх потрібно лише активувати. Однак менеджер SNMP відсутній у більшості мереж, і це роль, яку виконує система моніторингу мережі.

За протоколом SNMP кожен агент пристрою постійно сканує свій хост, заповнюючи форму, яка називається інформаційною базою керування (MIB). Він нічого не робить із цією MIB, але продовжує її оновлювати. Агент пристрою також прослуховує мережу для певного пакету, який є запитом SNMP.

Коли це надходить, агент пристрою надсилає свій MIB на IP-адресу, з якої надійшов запит. Менеджер SNMP надсилає запит як широкомовну розсилку, що означає, що він не адресований певному адресату. Таким чином, йому не потрібно знати жодних IP-адрес, щоб надіслати запит.

Коли диспетчер SNMP отримує всі відповіді від агента пристрою, він виявляє всі пристрої в мережі, включаючи деталі обладнання, операційну систему, компоненти та адресу. Кожен інтерфейс комутатора або маршрутизатора містить IP-адресу пристрою, до якого він підключений. Таким чином, інформація в MIB дозволяє побудувати карту мережі.

Отже, SNMP дозволяє монітору продуктивності мережі виявляти всі пристрої в мережі, створювати інвентаризацію обладнання, генерувати карту топології мережі та підтверджувати, що всі пристрої працюють належним чином. За SNMP агенту пристрою не потрібно чекати запиту, якщо він виявить проблему. У цьому випадку деталі проблеми записуються в MIB, який негайно надсилається. Це називається перехопленням SNMP, і монітор мережі інтерпретує це як сповіщення. Таким чином, SNMP автоматизує мережевий моніторинг, тому що технічні спеціалісти повинні звертати увагу лише тоді, коли виникають проблеми.

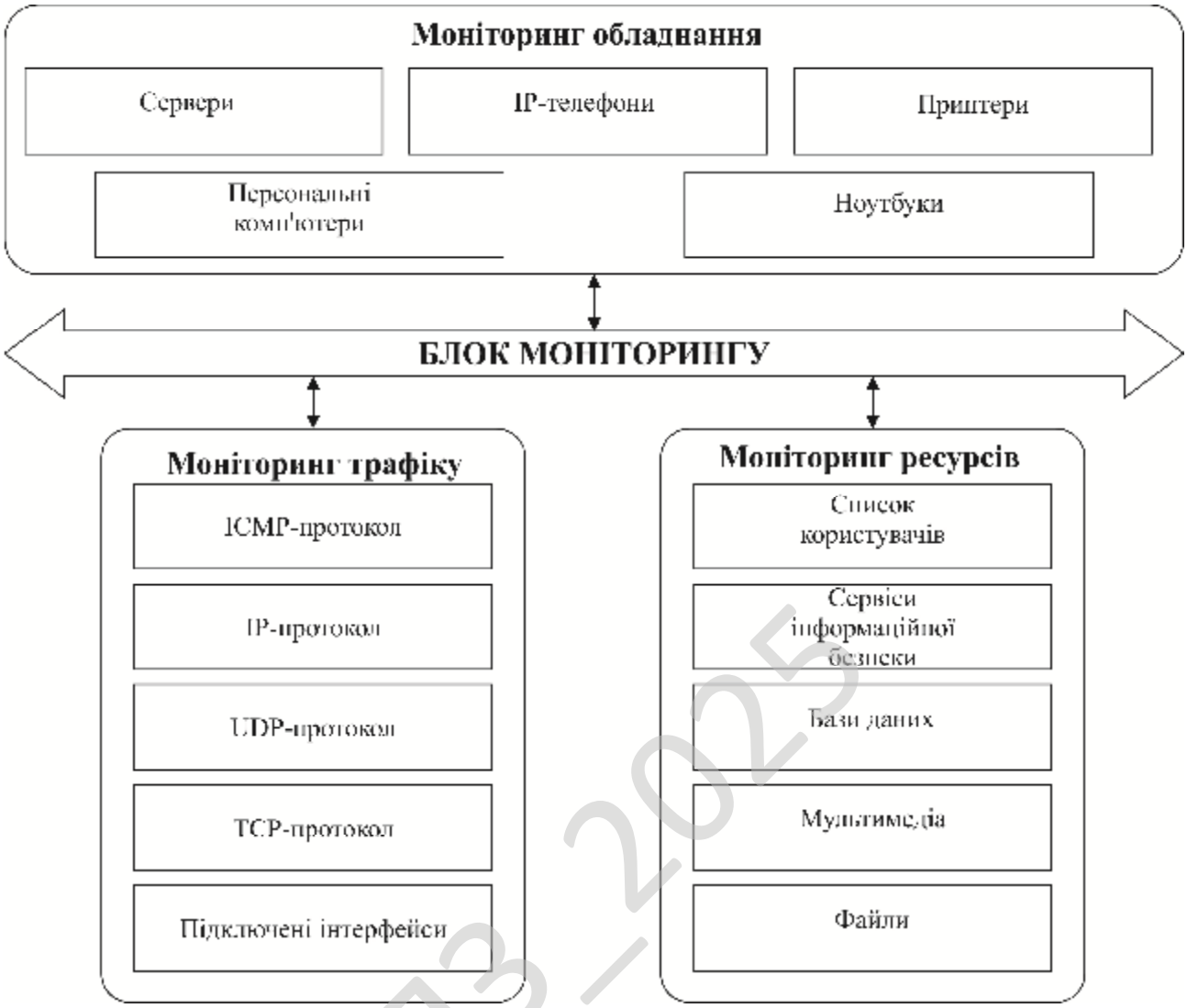

Рисунок 3.1 – Структурна схема системи

Опишемо технології, які використовуються у даній роботі.

Мережевий монітор за допомогою мікроконтролера ESP32

Створимо мережевий монітор за допомогою мікроконтролера ESP32. Проект відповідає двом ключовим потребам: по-перше, ідентифікує IP-адреси пристроїв, підключених до мережі, а по-друге, відстежує доступ до мережі з метою безпеки. Мотивація випливає з необхідності швидкого доступу до IP-адреси, призначеної мікроконтролерам, підключеним через Wi-Fi. Традиційно отримання цієї інформації вимагало підключення пристрою до комп’ютера, але рішення усуває цю проблему.

Цілі проекту ясні: розробити зручний пристрій, який надсилає сповіщення в реальному часі через Месенджер щоразу, коли пристрій підключається до мережі. Можливість Wi-Fi ESP32 дає змогу налаштувати пристрій без перепрограмування. Крім того, дослідимо функцію програмування веб-браузера, яка дозволяє користувачам програмувати ESP32, просто підключивши його до комп'ютера та вказавши браузеру певну адресу. Апаратне забезпечення для цього проекту мінімальне, для нього потрібен лише модуль ESP32, кабель живлення та адаптер.

Заглиблюється в аспекти програмного забезпечення, докладно опишемо налаштування ескізу Arduino та основних бібліотек. Бібліотека Espressif фреймворку ESP32 є обов'язковою умовою, а також спеціальні бібліотеки, такі як Arduino JSON для обробки даних. Використаємо функції ескізу, включаючи функції налаштування та циклу, моніторинг мережі та розбір повідомлень. Потім обробимо вхідні повідомлення з Месенджера.

Покажемо, як інтегрувати сповіщення Месенджера. Створемо бота Месенджера за допомогою BotFather, отримати унікальний ключ-токен та ідентифікатор користувача для автентифікації. Підкреслимо зручність Wi-Fi Manager, який полегшує переналаштування мережі без необхідності перепрограмування. Заземливши контакт 13 під час завантаження, користувачі можуть отримати доступ до диспетчера Wi-Fi для введення облікових даних, що робить пристрій дуже адаптованим для різних мереж.

Результатом є доступний мережевий монітор, який автоматично надсилає сповіщення про підключені пристрої через Месенджера. Чіткі та методичні пояснення в поєднанні з універсальними функціями ESP32 роблять цей проект цінним доповненням до набору інструментів будь-якого виробника. Цей епізод, представлений елементом 14, демонструє потужність технології «зроби сам» у спрощенні повсякденних завдань із програмами, починаючи від керування домашньою мережею та закінчуючи підвищенням безпеки.

Навіщо створювати мережевий монітор?

На відміну від загальнодоступної мережі Wi-Fi, доступної в аеропортах, кафе та на спортивних заходах, ваша домашня мережа Wi-Fi має бути безпечною. Однак, надання облікових даних відвідувачам може створити ризик, особливо якщо цей пароль буде надано іншим. Щоб допомогти собі швидше ідентифікувати невідомі пристрої, створимо недорогий інструмент моніторингу мережі, який міг би сповіщати його, коли щось підключається.

Короткий опис матеріалів

Порівняно з іншими мережевими моніторами Wi-Fi, ця версія була б досить простою як щодо програмних, так і апаратних можливостей. У цій ітерації конструкція складається з одного ESP32 завдяки вбудованій мікросхемі/антені Wi-Fi та блоку живлення USB.

Налаштування пристрою

Перш ніж розпочати будь-яке сканування, пристрій спочатку має знати, куди підключитися, і це робиться шляхом початкового натискання кнопки, підключеної до цифрового контакту вводу-виводу, який змушує ESP32 створити точку доступу. Після підключення веб-сторінка конфігурації представляє форму для введення SSID цільової точки доступу, пароля та часового поясу. Крім того, користувач може додати свій токен Месенджера API та ідентифікатор чату, щоб отримати доступ до сповіщень у реальному часі.

UDP-пакети

Протокол дейтаграм користувача, або UDP, є надзвичайно простим протоколом зв’язку, за допомогою якого повідомлення можна надсилати без необхідності попереднього налаштування чи додаткового виправлення помилок. Завдяки цьому та в поєднанні з IPv4 можна легко отримати IP-адресу відправника та порти, до яких здійснюється доступ. Але це надає лише обмежену інформацію та може надсилатися тисячі разів на день з одного пристрою, тому ми вирішили вибрати лише пакети DHCP, оскільки вони надсилаються, коли пристрій приєднується до мережі або потребує оновлення своєї IP-адреси. Вони містять

MAC-адресу клієнта, IP-адресу та, за бажанням, ім’я хоста, серед багатьох інших параметрів.

Інтеграція Месенджера

Оскільки монітор Wi-Fi тепер може збирати та аналізувати пакети DHCP на корисну інформацію, ми хочемо, щоб він надсилав сповіщення через Месенджер щоразу, коли пристрій приєднується до мережі. Це було досягнуто шляхом налаштування бота Месенджера, отримання маркера API, а потім налаштування клієнта в мікропрограмі ESP32. Бот може не тільки надсилати інформацію про новий пристрій через чат, але й користувачі також можуть надсилати йому команди, наприклад, вимкнути й увімкнути звук, і допомагати легко керувати ним без необхідності перепрограмування чи доступу до локальної мережі.

JSON-T

Шаблон JSON (JSON-T) – це мінімальна, але потужна мова шаблонів, розроблена для поєднання з набором даних JSON. Ці дані надаються Squarespace і генеруються динамічно, містять увесь вміст вашого сайту. Squarespace використовує JSON-T для перетворення даних у веб-сторінку. Цей процес називається «відтворенням» веб-сторінки. Засіб візуалізації об’єднує дані з CMS, також відомі як «контекст», із кодом JSON-T для створення виводу HTML. Потім цей HTML надсилається у ваш браузер і відображається.

Ось загальний огляд JSON-T і принципів його роботи:

JSON-T використовує спеціальний синтаксис, щоб позначити місце вставлення даних на сторінку. Наприклад: {foo}. Вони називаються тегами JSON-T. У JSON-T є два основних типи тегів: змінні та директиви.

– Змінні використовуються для вставки даних на сторінку. Вони повідомляють рендереру, які дані відтворювати. Це тег змінної:{foo}

– Директиви схожі на команди. Вони повідомляють рендереру, як відобразити розділ JSON. Ви можете впізнати їх, оскільки вони використовують додаткову крапку, і вони часто йдуть парами. Наприклад:{.section foo} expand foo {.end}.

Заміна змінної

Змінні вводять дані з контексту JSON на сторінку. Щоб додати значення baz, помістіть його у фігурні дужки:

```
<!-- will print Hello -->
{baz}
<!-- given this JSON context -->
{ "baz": "Hello" }
```

Ви також можете деталізувати структуру JSON за допомогою крапкової нотації. Під час пошуку імені змінної ми починаємо з верхнього рівня контексту та вибираємо вкладений об’єкт, який відповідає кожній частині змінної.

```
<!-- will print Hello -->
{foo.bar.baz}
<!-- given this JSON context --->
{
  "foo": {
    "bar": {
      "baz": "Hello"
    }
  }
}
```

Вбудовані директиви

Директиви – це вбудовані мовні конструкції, які зазвичай починаються з крапки (.). Дві основні директиви, які використовуються в JSON-T, це розділи та повторювані розділи

{.section foo}запускає розділ під назвою foo. Ім'я відповідає ключу JSON. Розділ розгортається, якщо ключ присутній, а не false. Розділи закриваються директивою. {.end}

```
<!-- will print Hello -->
{.section foo}
    {bar}
{.end}
<!-- given this JSON context -->
{
  "foo": {
    "bar": "Hello"
  }
```

```
}
<!-- but prints nothing if "foo" is missing -->
```

{.repeated section bar}починає повторюваний розділ під назвою bar. Ім'я відповідає ключу JSON, значенням якого є список словників або рядків. Розділ розгортається один раз для кожного елемента списку. Спеціальна змінна представляє значення контексту в поточній області або поточній активній частині контексту.{@}

```
<!-- will print Hello World -->
{.repeated section bar}
  {@}
{.end}
<!-- given the JSON context -->
{
  "bar": ["Hello", "World"]
}
```

Збираємо все разом

Поєднання цих простих конструкцій разом із вашим HTML, CSS і JavaScript – це те, як будується шаблон Squarespace, використовуючи CMS для надання даних та інформації, які ми відтворюємо у наших файлах шаблонів.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

З рисунку видно, що розроблена система складається з наступних блоків:

– Робота з ресурсами мережі.

– Робота з сесіями.

– Функції для роботи з мережею.

– Моніторинг трафіку.

– Робота з файлами.

– Монітор з'єднань.

– Статистика подій.

Розглянемо детальніше кожний з блоків.

Система відображає наявні у мережі ресурси у вигляді дерева. Пошук ресурсів можна здійснювати по заданим умовам: локальні чи глобальні ресурси; всі ресурси, тільки файли, чи тільки принтери, тощо. Можна додавати до загальних ресурсів мережі свої власні, а також закривати їх потім.

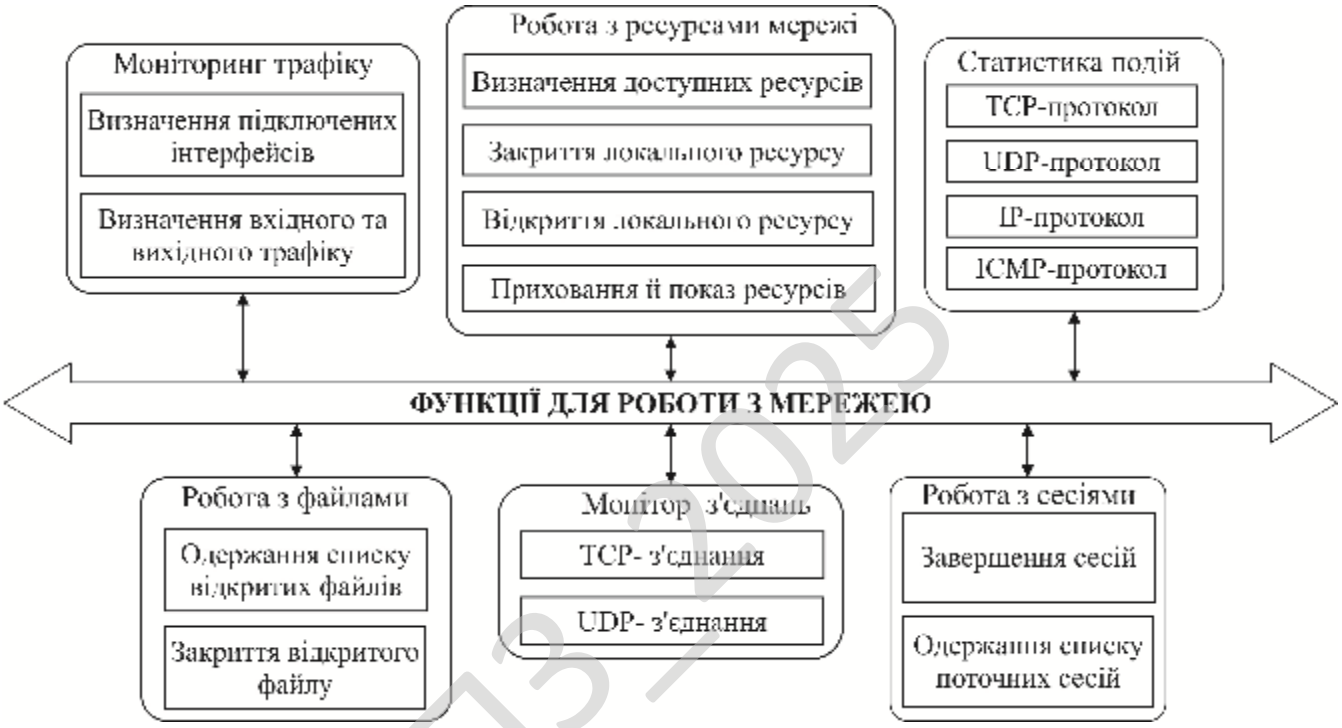

Рисунок 3.2 – Функціональна схема системи

Робота з сесіями включає в себе:

– Одержання списку поточних сесій.

– Завершення сесій.

Програма дозволяє переглянути список відкритих сесій, що включає в себе: назву сесії, користувача, що її розпочав, номер сесії, час роботи та час очікування. Моніторинг трафіку включає в себе:

– Визначення підключених інтерфейсів.

– Визначення вхідного та вихідного трафіку.

Розроблена система відображає всі інтерфейси приєднані до комп'ютера, на якому запущена програма, їх MAC-адреси та вхідний і вихідний трафік на кожному з них. Робота з файлами включає в себе:

– Одержання списку відкритих файлів.

– Закриття відкритого файлу.

Можна переглянути, які з Ваших файлів, що Ви відкрили для загального доступу, переглядають по мережі. Програма відобразить список файлів та користувачів, які їх переглядають. Також можна відкрити чи закрити файл. Монітор з'єднань включає в себе:

– Відстеження TCP- з'єднань.

– Відстеження UDP- з'єднань.

Система фіксує всі підключення по TCP- та UDP-протоколу, та виводить їх на екран у форматі *IP-адреса:порт_призначення*.

Статистика подій включає в себе відстеження подій в наступних протоколах:

– TCP-протокол.

– UDP-протокол.

– IP-протокол.

– ICMP-протокол.

Статистика ведеться по цілому ряду параметрів. Наприклад для TCP-протоколу фіксується: Тип алгоритму повторної передачі, мінімальний тайм-аут, максимальний тайм-аут, максимальна кількість помилок з'єднання, активні з'єднання, пасивні з'єднання, невдалі спроби відкриття, скидання встановлених з'єднань, отримані сегменти, надіслані сегменти, повторно передані сегменти, помилки тощо. Робота з ресурсами мережі включає в себе:

– Визначення доступних ресурсів.

– Закриття локального ресурсу.

– Відкриття локального ресурсу.

– Приховання й показ ресурсів.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

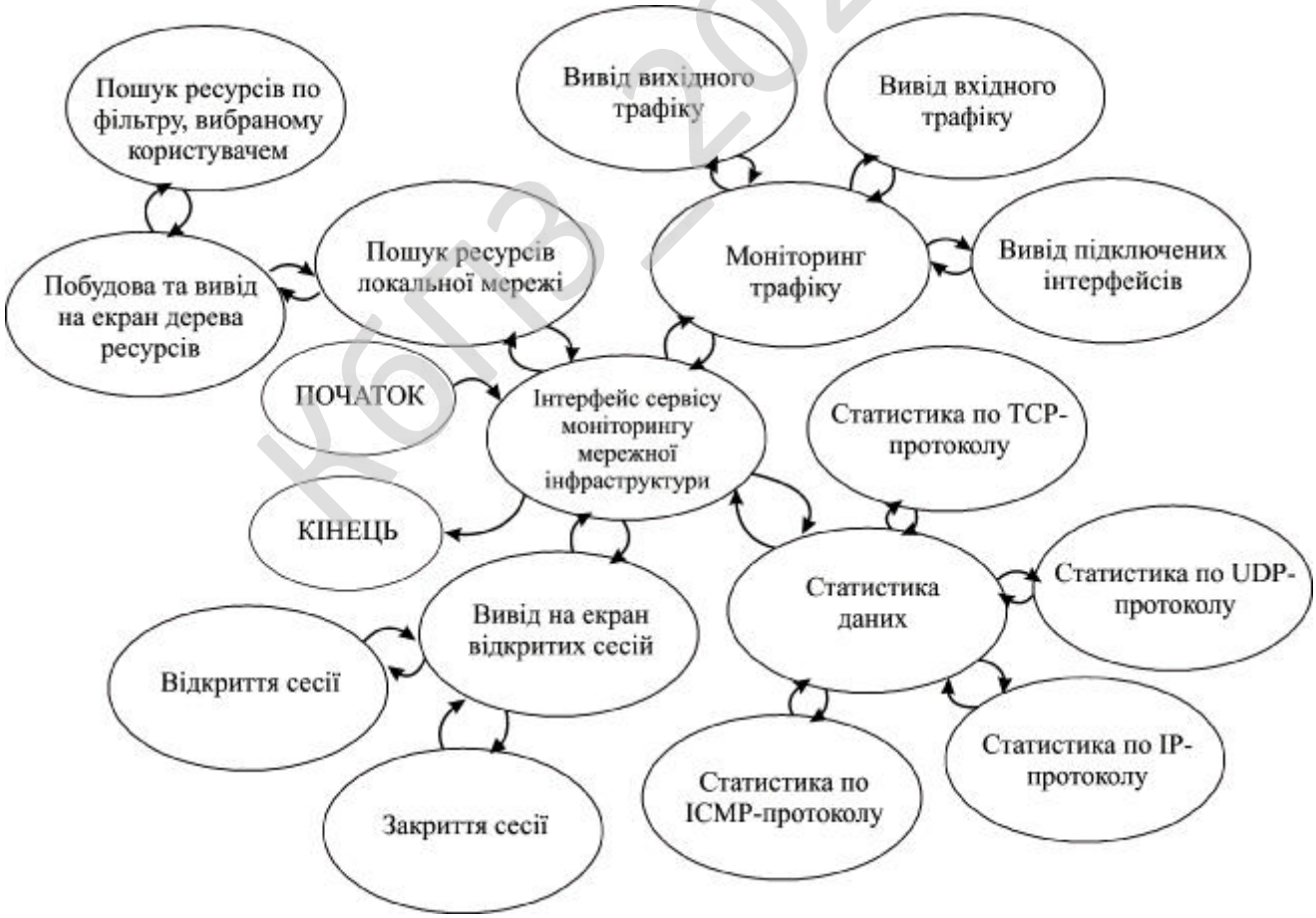

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму. У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів. Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними. У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи. Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю сервісу

моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

При розробці використовувались концепції діаграм діяльності. Тобто в UML, візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.

Це фундаментальна одиниця визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів. Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності.

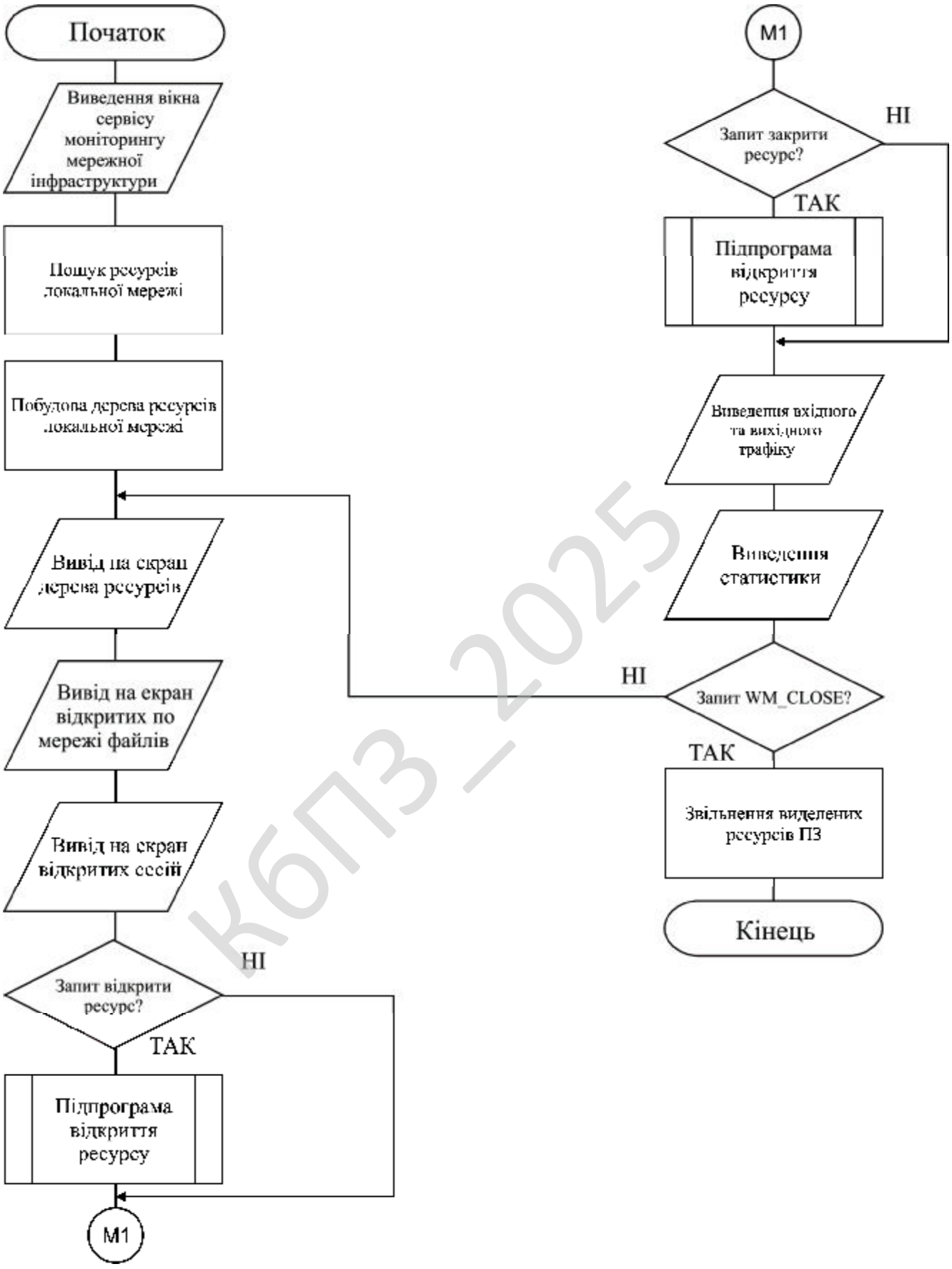

Рисунок 4.1 – Блок-схема основної програми

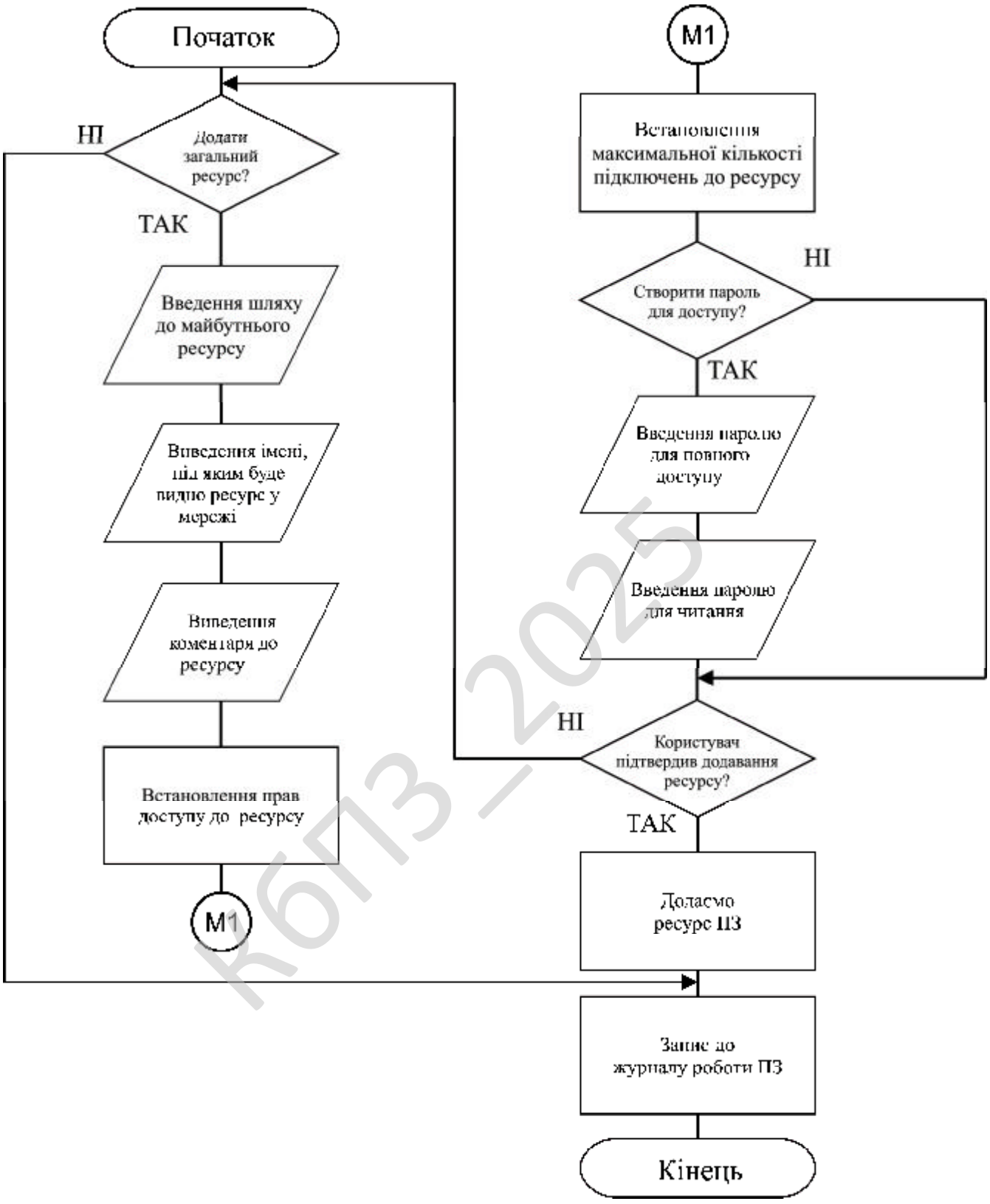

Рисунок 4.2 – Блок-схема роботи підпрограми

Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед

прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0. . *), "одиниця або більше" (1. . *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0.. 1, 3..4, 6. . *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Розглянемо опис програмної реалізації

Система моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 реалізується в межах випускної кваліфікаційної роботи вищого навчального закладу і призначена для автоматичного фіксування встановлених з'єднань протоколом TCP/IP із подальшою генерацією повідомлень у форматі JSON шаблонів та їх передачею на центральний сервер або локальне зберігання з візуально-звуковою сигналізацією.

Апаратною основою виступає мікроконтролер ESP32 який містить два ядра з частотою 240 мегагерц 520 кБ оперативної пам'яті та 4 МБ флеш пам'яті при цьому вбудований модуль Wi Fi забезпечує бездротовий зв'язок з існуючою мережею підтримка бібліотеки ArduinoJson дозволяє ефективно формувати та обробляти JSON шаблони з мінімальними затратами ресурсів.

Архітектура системи побудована як сукупність модулів управління мережею модулю моніторингу TCP/IP модулю формування повідомлень JSON модулю передачі даних та модулю сигналізації.

Модуль управління мережею зчитує налаштування SSID та пароль із файлової системи SPIFFS що дає можливість змінювати параметри без перепрошивки і забезпечує ініціалізацію з'єднання з Wi Fi мережою.

Модуль моніторингу TCP/IP перевіряє стан серверних сокетів що відкриті на заздалегідь заданих портах і виконує опитування кожні 100 мілісекунд.

В разі виявлення нового з'єднання збирається інформація про IP адресу порт клієнта та мітку часу які потім передаються в модуль формування JSON.

Модуль формування JSON використовує об'єкт StaticJsonDocument розміром 512 байт що перевищує середній розмір повідомлення приблизно 200 байт і гарантує запас пам'яті на майбутнє розширення структури.

Формування одного повідомлення займає 1,5 мілісекунди при частоті 10 подій за секунду сумарне навантаження на процесор становить 15 мілісекунд що складає приблизно 1,5 відсотка при тактовій частоті 240 мегагерц.

Модуль передачі даних реалізує функцію sendMessage яка надсилає згенероване JSON повідомлення на віддалений сервер за допомогою HTTP POST або публікації в MQTT залежно від заданих параметрів.

В разі відсутності зв'язку повідомлення зберігаються в черзі файлової системи SPIFFS та відправляються повторно після відновлення мережі що забезпечує надійність доставки.

Модуль сигналізації активує світлодіод або зумер під час кожного зафіксованого з'єднання Функція triggerAlert генерує короткий імпульс тривалістю 50 мілісекунд перевірка за допомогою осцилографа показує що затримка між подією та сигналом не перевищує 2 мілісекунди.

В файлі main cpp визначені функції setup та loop У setup виконується ініціалізація серійного порту на швидкості 115200 бод виклик initWiFi та ініціалізація модулю моніторингу.

У loop відбувається виклик monitorConnections обробка черги накопичених повідомлень функцією processQueue та виклик triggerAlert у разі потреби.

Розрахунки ресурсів показують що обсяг оперативної пам'яті для JSON буфера 512 байт не перевищує доступних 520 кБ, а час обробки одного повідомлення у 1,5 мілісекунди забезпечує запас продуктивності для розширення кількості одночасних з'єднань до 100 при тому що сукупне навантаження не перевищить 15 відсотків продуктивності двоядерного процессора при роботі з мережею 2.4 ГГц.

Таким чином обрані проектні рішення підтверджують свою коректність та забезпечують стабільну роботу системи моніторингу та сигналізації TCP/IP з'єднань у реальному часі.

Нижче наведено код на C++

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <SPIFFS.h>
```

```
const char* ssid = "yourSSID";
const char* password = "yourPASSWORD";
const char* serverUrl = "http://example.com/endpoint";
const uint16_t ports[] = {80, 443, 8080};
const size_t numPorts = sizeof(ports) / sizeof(ports[0]);
const int ledPin = 2;
const int buzzerPin = 15
// Ініціалізація підключення Wi Fi
void initWiFi() {
Serial.begin(115200);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
}
}

// Моніторинг TCP з'єднань на вказаному порту
bool monitorConnections(uint16_t port, String& clientIp, uint16_t&
clientPort) {
WiFiServer server(port);
server.begin();
WiFiClient client = server.available();
if (!client) {
server.stop();
return false;
}
clientIp = client.remoteIP().toString();
clientPort = client.remotePort();
client.stop();
server.stop();
return true;
}

// Формування та відправка JSON повідомлення
void sendMessage(const String& jsonMessage) {
if (WiFi.status() == WL_CONNECTED) {
HTTPClient http;
http.begin(serverUrl);
http.addHeader("Content-Type", "application/json");
int httpCode = http.POST(jsonMessage);
http.end();
if (httpCode <= 0) {
```

```
SPIFFS.begin(true);
File file = SPIFFS.open("/queue.json", FILE_APPEND);
file.println(jsonMessage);
file.close();
}
}
}

// Активація світлодіода та зумера
void triggerAlert() {
digitalWrite(ledPin, HIGH);
digitalWrite(buzzerPin, HIGH);
delay(50);
digitalWrite(ledPin, LOW);
digitalWrite(buzzerPin, LOW);
}

void setup() {
pinMode(ledPin, OUTPUT);
pinMode(buzzerPin, OUTPUT);
initWiFi();
SPIFFS.begin(true);
}

void loop() {
static size_t currentPort = 0;
String ip;
uint16_t portC;
if (monitorConnections(ports[currentPort], ip, portC)) {
StaticJsonDocument<512> doc;
doc["event"] = "connection";
doc["ip"] = ip;
doc["port"] = portC;
doc["timestamp"] = millis();
String output;
serializeJson(doc, output);
sendMessage(output);
triggerAlert();
}
currentPort = (currentPort + 1) % numPorts;
delay(100);
}
```

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм SEED – у криптографії симетричний блоковий криптоалгоритм на основі Мережі Фейстеля, розроблений Корейським агентством інформаційної безпеки (Korean Information Security Agency, KISA) в 1998 році.

В алгоритмі використовується 128-бітний блок і ключ довжиною 128 біт. Алгоритм одержав широке поширення й використовується фінансовими й банківськими структурами, виробничими підприємствами й бюджетними установами Південної Кореї, оскільки 40-бітний SSL не забезпечує на даний момент мінімально необхідного рівня безпеки.

Агентством по захисту інформації специфіковане використання шифру SEED у протоколах TLS і S/MIME.

У той же час, алгоритм SEED не реалізований у більшості сучасних браузерів і інтернет-додатків, що утрудняє його використання в даній сфері поза межами Південної Кореї.

SEED являє собою мережу Фейстеля з 16 раундами, 128-бітовими блоками й 128-бітовим ключем.

Алгоритм використовує дві 8×8 таблиці підстановки, які, як такі з Safer, виведені з дискретного зведення в ступінь (у цьому випадку, x^{247} і x^{251} – плюс деякі «несумісні операції»).

Це є деякою подібністю с MISTY1 у рекурсивності його структури: 128-бітовий повний шифр – мережа Фейстеля з F-функцією, що впливає на 64-бітові половини, у той час як сама F-функція – Мережа Фейстеля, складена з G-функції, що впливає на 32-розрядні половини.

Однак рекурсія не простягнеться далі, тому що G-функція – не Мережа Фейстеля.

В G-функції 32-розрядне слово розглядають як чотири 8-бітових байта, кожний з яких проходить через одну або іншу таблицю підстановки, потім поєднується в помірковано комплексному наборі булевих функцій таким чином, що кожний біт виводу залежить від 3 з 4 вхідних байтів.

SEED має складний ключовий розклад, генеруючи тридцять два 32-розрядних додаткових символу, використовуючи G-функції на серіях обертань вихідного неопрацьованого ключа, комбінованого зі спеціальними раундовими константами (як в TEA) від «Золотого співвідношення» (англ. Golden ratio).

Згідно з дослідженнями KISA, алгоритм SEED «надійно протистоїть відомим атакам».

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи. Розроблене програмне забезпечення сервісу моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів складається з наступних функціональних блоків:

– Підрозділів: Ресурси; Тип Ресурсів; Використання.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ: Оновити список ресурсів; Моніторинг; Статистика; Про програму; Відкрити ресурс;. Закрити ресурс; Додати ресурс.

– Підрозділу виведення мережних даних.

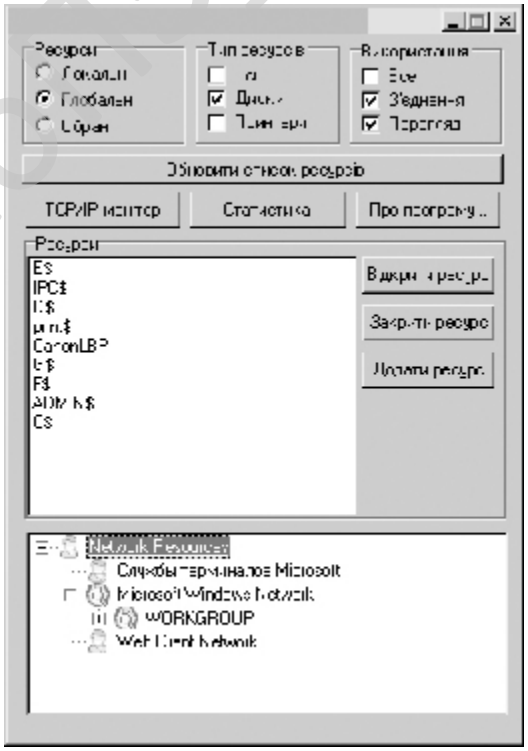

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

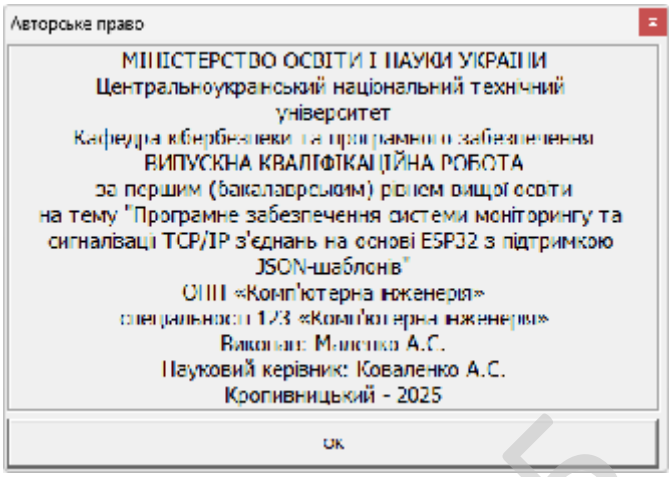

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

– Досліджена система моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи моніторингу та сигналізації TCP/IP з'єднань на основі ESP32 з підтримкою JSON-шаблонів. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм SEED.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оліфер В.Г. Комп'ютерні мережі. Принципи, технології, протоколи. Підручник / В.Г. Оліфер, Н.А.Оліфер. – [5-е вид.]. – 2016. – 944 с.

2. Е. Таненбаум, Д. Уезеролл «Комп'ютерні мережі». – [5-е вид.]. – 2016. – 960 с.

3. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

4. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

5. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

6. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

7. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

8. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

9. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

10. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

11. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

12. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

13. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

14. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

15. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

16. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

17. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

18. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

19. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

20. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

21. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

22. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

23. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

24. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

25. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and

Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

26. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

27. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

28. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

29. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

30. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

31. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

32. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

33. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

34. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

35. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

36. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

37. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

38. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

39. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

40. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

41. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

42. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

43. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

44. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

45. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

46. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

47. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

48. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

49. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

50. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

51. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 47-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 104 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 9.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко А.С.

Програмне забезпечення системи моніторингу та сигналізації TCP/IP з’єднань на основі ESP32 з підтримкою JSON-шаблонів

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 22

Літера: РП

Кропивницький – 2025 року

Основна програма

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <ArduinoJson.h>
#include <Ticker.h>

#define MAX_CONNECTIONS 10
#define JSON_CAPACITY 1024

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
const char* serverHost = "192.168.1.100";
const uint16_t serverPort = 8080;

// ConnectionInfo holds details for each monitored connection
struct ConnectionInfo {
    IPAddress ipAddress;
    uint16_t portNumber;
    bool connectionIsActive;
    unsigned long lastCheckedTimestamp;
};

// JSONTemplateManager creates JSON payloads according to templates
class JSONTemplateManager {
public:
    JSONTemplateManager() {
    }
    // build JSON report for all connections
    String buildConnectionsReport(ConnectionInfo connections[], size_t count) {
        DynamicJsonDocument doc(JSON_CAPACITY);
        JsonArray array = doc.createNestedArray("connections");
        for (size_t idx = 0; idx < count; idx++) {
            JsonObject obj = array.createNestedObject();
            obj["ip"] = connections[idx].ipAddress.toString();
            obj["port"] = connections[idx].portNumber;
            obj["status"] = connections[idx].connectionIsActive ? "active" :
"inactive";
            obj["last_checked"] = connections[idx].lastCheckedTimestamp;
        }
        String output;
        serializeJson(doc, output);
        return output;
    }
};

// ConnectionMonitor scans TCP/IP connections periodically
class ConnectionMonitor {
public:
    ConnectionInfo connections[MAX_CONNECTIONS];
    size_t connectionCount;

    ConnectionMonitor() {
        connectionCount = 0;
    }

    void addConnection(IPAddress ip, uint16_t port) {
        if (connectionCount < MAX_CONNECTIONS) {
            connections[connectionCount].ipAddress = ip;
            connections[connectionCount].portNumber = port;
            connections[connectionCount].connectionIsActive = false;
            connections[connectionCount].lastCheckedTimestamp = 0;
```

```
            connectionCount++;
        }
    }

    void checkAllConnections() {
        for (size_t idx = 0; idx < connectionCount; idx++) {
            bool status = checkSingleConnection(connections[idx].ipAddress,
connections[idx].portNumber);
            connections[idx].connectionIsActive = status;
            connections[idx].lastCheckedTimestamp = millis();
        }
    }

private:
    // attempt to establish TCP connection
    bool checkSingleConnection(IPAddress ip, uint16_t port) {
        WiFiClient client;
        bool result = false;
        if (client.connect(ip, port)) {
            result = true;
            client.stop();
        }
        return result;
    }
};

// AlarmManager signals via LED and buzzer
class AlarmManager {
public:
    uint8_t ledPin;
    uint8_t buzzerPin;

    AlarmManager(uint8_t led, uint8_t buzzer) {
        ledPin = led;
        buzzerPin = buzzer;
        pinMode(ledPin, OUTPUT);
        pinMode(buzzerPin, OUTPUT);
    }

    void signalAlarm() {
        digitalWrite(ledPin, HIGH);
        tone(buzzerPin, 1000);
        delay(200);
        digitalWrite(ledPin, LOW);
        noTone(buzzerPin);
    }
};

// HttpReporter sends JSON to central server
class HttpReporter {
public:
    HttpReporter() {
    }

    void postReport(const String& payload) {
        WiFiClient client;
        if (!client.connect(serverHost, serverPort)) {
            return;
        }
        client.print(String("POST ") + "/report" + " HTTP/1.1\r\n");
        client.print(String("Host: ") + serverHost + "\r\n");
        client.print("Content-Type: application/json\r\n");
        client.print(String("Content-Length: ") + payload.length() + "\r\n");
```

```
        client.print("\r\n");
        client.print(payload);
        client.stop();
    }
};

ConnectionMonitor globalConnectionMonitor;
JSONTemplateManager globalJsonManager;
AlarmManager globalAlarmManager(2, 15);
HttpReporter globalHttpReporter;
Ticker periodicTicker;

void performMonitoringTask() {
    globalConnectionMonitor.checkAllConnections();
    String report =
globalJsonManager.buildConnectionsReport(globalConnectionMonitor.connections,
globalConnectionMonitor.connectionCount);
    globalHttpReporter.postReport(report);
    for (size_t idx = 0; idx < globalConnectionMonitor.connectionCount; idx++) {
        if (!globalConnectionMonitor.connections[idx].connectionIsActive) {
            globalAlarmManager.signalAlarm();
        }
    }
}

void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
    }
    globalConnectionMonitor.addConnection(IPAddress(192, 168, 1, 10), 80);
    globalConnectionMonitor.addConnection(IPAddress(192, 168, 1, 20), 22);
    globalConnectionMonitor.addConnection(IPAddress(192, 168, 1, 30), 443);
    globalConnectionMonitor.addConnection(IPAddress(8, 8, 8, 8), 53);
    globalConnectionMonitor.addConnection(IPAddress(1, 1, 1, 1), 80);
    periodicTicker.attach(5, performMonitoringTask);
}

void loop() {
    // keep the ticker running
}
```

Файл CONNECTION.py

```
#include <WiFi.h>
#include <WebServer.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>
#include <ArduinoOTA.h>
#include <SPI.h>
#include <SD.h>
#include <Ticker.h>

#define MAX_CONNECTIONS 10
#define JSON_CAPACITY 2048
#define SD_CS_PIN 5
#define MAX_USERS 5

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";
const char* mqttServer = "broker.hivemq.com";
const uint16_t mqttPort = 1883;

struct ConnectionInfo {IPAddress ip; uint16_t port; bool active; unsigned long
timestamp;};
struct User {String username; String passwordHash; String role;};

ConnectionInfo connections[MAX_CONNECTIONS];
size_t connectionCount = 0;
User users[MAX_USERS] = {
    {"admin","admin_hash","administrator"},
    {"user1","hash1","operator"},
    {"user2","hash2","viewer"},
    {"guest","hash3","guest"},
    {"tester","hash4","tester"}
};

bool isAuthenticated = false;
String currentUser = "";

WebServer server(80);
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);
Ticker monitorTicker;

class ConnectionMonitor {
public:
    void add(IPAddress ip,uint16_t
port){if(connectionCount<MAX_CONNECTIONS){connections[connectionCount].ip=ip;con
nections[connectionCount].port=port;connections[connectionCount].active=false;co
nnections[connectionCount].timestamp=0;connectionCount++;}}
    void checkAll(){for(size_t
i=0;i<connectionCount;i++){connections[i].active=check(connections[i].ip,connect
ions[i].port);connections[i].timestamp=millis();}}
private:
    bool check(IPAddress ip,uint16_t port){WiFiClient c;bool
r=false;if(c.connect(ip,port)){r=true;c.stop();}return r;}
} monitor;

class JSONTemplateManager {
public:
    String buildReport(){DynamicJsonDocument doc(JSON_CAPACITY);JsonArray
arr=doc.createNestedArray("connections");for(size_t
i=0;i<connectionCount;i++){JsonObject
```

```
o=arr.createNestedObject();o["ip"]=connections[i].ip.toString();o["port"]=connec
tions[i].port;o["status"]=connections[i].active?"active":"inactive";o["timestamp
"]=connections[i].timestamp;}String s;serializeJson(doc,s);return s;}
} jsonManager;

void
handleRoot(){if(!isAuthenticated){server.sendHeader("Location","/login");server.
send(302,"text/plain","");return;}String h="<html><body><h1>ESP32
Monitor</h1><p>Welcome "+currentUser+"</p><a
href='/logout'>Logout</a></body></html>";server.send(200,"text/html",h);}
void handleLogin(){if(server.method()==HTTP_GET){String h="<html><body><form
method='POST' action='/login'><input name='user' placeholder='Username'><input
name='pass' placeholder='Password'
type='password'><button>Login</button></form></body></html>";server.send(200,"te
xt/html",h);}else{String u=server.arg("user");String
p=server.arg("pass");for(int
i=0;i<MAX_USERS;i++){if(users[i].username==u&&users[i].passwordHash==p){isAuthen
ticated=true;currentUser=u;server.sendHeader("Location","/");server.send(302,"te
xt/plain","");return;}}server.send(401,"text/plain","Unauthorized");}}
void
handleLogout(){isAuthenticated=false;currentUser="";server.sendHeader("Location"
,"/login");server.send(302,"text/plain","");}

void
setupServer(){server.on("/",handleRoot);server.on("/login",handleLogin);server.o
n("/logout",handleLogout);server.begin();}
void
setupWiFi(){WiFi.begin(ssid,password);while(WiFi.status()!=WL_CONNECTED)delay(50
0);}
void reconnectMQTT(){while(!mqttClient.connected()){String id="ESP32-
"+String(random(0xffff),HEX);if(mqttClient.connect(id.c_str()))mqttClient.subscr
ibe("esp32/commands");else delay(1000);}}
void logToSD(String m){File
f=SD.open("/log.txt",FILE_APPEND);if(f){f.println(m);f.close();}}
void performMonitor(){monitor.checkAll();String
report=jsonManager.buildReport();mqttClient.publish("esp32/monitor",report.c_str
());logToSD(report);}
void setupOTA(){ArduinoOTA.begin();}
void setupMQTT(){mqttClient.setServer(mqttServer,mqttPort);}
void setupSD(){SD.begin(SD_CS_PIN);}

void setup() {
    setupWiFi();
    setupServer();
    setupMQTT();
    setupSD();
    setupOTA();
    monitor.add(IPAddress(192,168,1,10),80);
    monitor.add(IPAddress(192,168,1,20),22);
    monitor.add(IPAddress(192,168,1,30),443);
    monitor.add(IPAddress(8,8,8,8),53);
    monitor.add(IPAddress(1,1,1,1),80);
    monitorTicker.attach(10,performMonitor);
}

void loop(){
    server.handleClient();
    if(!mqttClient.connected()) reconnectMQTT();
    mqttClient.loop();
    ArduinoOTA.handle();
}
```

Файл MAX_SENSOR.py

```
#include <WiFi.h>
#include <Wire.h>
#include <SNMPAgent.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Ticker.h>

#define MAX_SENSORS 5
#define OLED_WIDTH 128
#define OLED_HEIGHT 64

const char* ssid="YOUR_SSID";
const char* password="YOUR_PASSWORD";
const char* smtpServer="smtp.example.com";
const int smtpPort=587;
const char* emailUser="user@example.com";
const char* emailPass="password";
const char* smsApiUrl="https://api.twilio.com/...";
const char* smsAuth="Basic ...";

IPAddress
sensorIPs[MAX_SENSORS]={{192,168,1,50},{192,168,1,51},{192,168,1,52},{192,168,1,
53},{192,168,1,54}};

struct SensorData{IPAddress ip;bool status;unsigned long timestamp;};

SensorData sensors[MAX_SENSORS];
size_t sensorCount=0;

SNMPAgent snmpAgent;
Ticker emulateTicker;
Ticker analyzeTicker;
Adafruit_SSD1306 display(OLED_WIDTH,OLED_HEIGHT,&Wire);

class VirtualSensorManager{
public:
    void add(IPAddress ip){
        if(sensorCount<MAX_SENSORS){
            sensors[sensorCount].ip=ip;
            sensors[sensorCount].status=false;
            sensors[sensorCount].timestamp=0;
            sensorCount++;
        }
    }
    void emulate(){
        for(size_t i=0;i<sensorCount;i++){
            sensors[i].status=random(0,2);
            sensors[i].timestamp=millis();
        }
    }
}vsManager;

class DataAnalyzer{
public:
    float computeAverage(){
        float sum=0;
        for(size_t i=0;i<sensorCount;i++){
            sum+=sensors[i].status?1:0;
```

```
        }
        return sum/sensorCount;
    }
    bool detectAnomaly(){
        float avg=computeAverage();
        return avg<0.2||avg>0.8;
    }
}analyzer;

void sendEmail(const String& subj,const String& body){
    HTTPClient http;
    http.begin("http://"+String(smtpServer)+":"+String(smtpPort)+"/send");
    http.setAuthorization(emailUser,emailPass);
    http.addHeader("Content-Type","application/json");
    DynamicJsonDocument doc(512);
    doc["to"]="admin@example.com";
    doc["subject"]=subj;
    doc["body"]=body;
    String p;
    serializeJson(doc,p);
    http.POST(p);
    http.end();
}

void sendSMS(const String& msg){
    HTTPClient http;
    http.begin(smsApiUrl);
    http.setAuthorization(smsAuth);
    http.addHeader("Content-Type","application/x-www-form-urlencoded");
    String body="To=%2B1234567890&From=%2B0987654321&Body="+msg;
    http.POST(body);
    http.end();
}

void drawDashboard(){
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    display.setCursor(0,0);
    display.println("Sensor Dashboard");
    for(size_t i=0;i<sensorCount;i++){
        display.print(sensors[i].ip.toString());
        display.print(" ");
        display.println(sensors[i].status?"UP":"DOWN");
    }
    display.display();
}

void setupSNMP(){
    snmpAgent.begin("ESP32_SNMP");
    for(size_t i=0;i<sensorCount;i++){
        snmpAgent.addOID("1.3.6.1.4.1.12345."+String(i+1),[](SNMPAgent::OIDType
t,const void* v){
            return (int)*((bool*)v);
        },&sensors[i].status);
    }
}

void emulateTask(){
    vsManager.emulate();
}

void analyzeTask(){
```

```
    if(analyzer.detectAnomaly()){
        sendEmail("Anomaly Detected","Sensor anomaly detected");
        sendSMS("Anomaly detected");
    }
    drawDashboard();
}

void setup(){
    WiFi.begin(ssid,password);
    while(WiFi.status()!=WL_CONNECTED)delay(500);
    randomSeed(micros());
    for(size_t i=0;i<MAX_SENSORS;i++){
        vsManager.add(sensorIPs[i]);
    }
    display.begin(SSD1306_SWITCHCAPVCC,0x3C);
    setupSNMP();
    emulateTicker.attach(5,emulateTask);
    analyzeTicker.attach(10,analyzeTask);
}

void loop(){
    snmpAgent.loop();
}
```

Файл TELEGRAM_CHAT.py

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <vector>
#include <functional>

#define JSON_CAPACITY 4096
#define TELEGRAM_BOT_TOKEN "YOUR_BOT_TOKEN"
#define TELEGRAM_CHAT_ID "YOUR_CHAT_ID"
#define WHITELIST_CAPACITY 10
#define BLACKLIST_CAPACITY 10

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

const char server_cert[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIID...
-----END CERTIFICATE-----
)EOF";

const char server_key[] PROGMEM = R"EOF(
-----BEGIN PRIVATE KEY-----
MIIE...
-----END PRIVATE KEY-----
)EOF";

WiFiClientSecure secureClient;
WiFiServerSecure secureServer(443);

class EventLogger {
public:
    std::vector<String> events;
    void logEvent(const String& e) {
        events.push_back(e);
    }
    String getEventsJson() {
        StaticJsonDocument<JSON_CAPACITY> doc;
        JsonArray arr = doc.createNestedArray("events");
        for (auto& e : events) arr.add(e);
        String s;
        serializeJson(doc, s);
        return s;
    }
};

class RuleEngine {
public:
    std::vector<std::function<void()>> rules;
    void addRule(std::function<void()> r) {
        rules.push_back(r);
    }
    void run() {
        for (auto& r : rules) r();
    }
};

std::vector<IPAddress> whitelist;
std::vector<IPAddress> blacklist;
```

```
bool isAllowed(IPAddress ip) {
    if (!whitelist.empty()) {
        bool inWhite = false;
        for (auto& w : whitelist) if (w == ip) { inWhite = true; break; }
        if (!inWhite) return false;
    }
    for (auto& b : blacklist) if (b == ip) return false;
    return true;
}

EventLogger logger;
RuleEngine ruleEngine;

void sendTelegramMessage(const String& text) {
    HTTPClient http;
    String url = "https://api.telegram.org/bot" TELEGRAM_BOT_TOKEN
"/sendMessage";
    http.begin(secureClient, url);
    http.addHeader("Content-Type", "application/json");
    StaticJsonDocument<512> doc;
    doc["chat_id"] = TELEGRAM_CHAT_ID;
    doc["text"] = text;
    String payload;
    serializeJson(doc, payload);
    http.POST(payload);
    http.end();
}

void getTelegramUpdates() {
    HTTPClient http;
    String url = "https://api.telegram.org/bot" TELEGRAM_BOT_TOKEN
"/getUpdates";
    http.begin(secureClient, url);
    int code = http.GET();
    if (code == 200) {
        String resp = http.getString();
        StaticJsonDocument<JSON_CAPACITY> doc;
        DeserializationError err = deserializeJson(doc, resp);
        if (!err) {
            JsonArray arr = doc["result"].as<JsonArray>();
            for (auto& item : arr) {
                String text = item["message"]["text"].as<String>();
                if (text == "/status") {
                    String events = logger.getEventsJson();
                    sendTelegramMessage(events);
                }
            }
        }
    }
    http.end();
}

void setup() {
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) delay(1000);
    secureClient.setCACert(server_cert);
    secureServer.setServerKeyAndCert_P((const uint8_t*)server_key,
strlen(server_key), (const uint8_t*)server_cert, strlen(server_cert));
    secureServer.begin();
    whitelist.push_back(IPAddress(192, 168, 1, 10));
    blacklist.push_back(IPAddress(192, 168, 1, 20));
    ruleEngine.addRule([]() {
```

```
        if (logger.events.size() > 10) {
            sendTelegramMessage("More than 10 events logged");
        }
    });
}

void loop() {
    WiFiClientSecure client = secureServer.available();
    if (client) {
        String req = client.readStringUntil('\r\n');
        if (req.startsWith("GET /events")) {
            IPAddress remote = client.remoteIP();
            if (isAllowed(remote)) {
                String resp = logger.getEventsJson();
                client.print("HTTP/1.1 200 OK\r\nContent-Type:
application/json\r\n\r\n");
                client.print(resp);
                logger.logEvent("Served /events to " + remote.toString());
            } else {
                client.print("HTTP/1.1 403 Forbidden\r\n\r\n");
                logger.logEvent("Blocked /events request from " +
remote.toString());
            }
        }
        client.stop();
    }
    getTelegramUpdates();
    ruleEngine.run();
    delay(5000);
}
```

Файл clientQ.py

```
import network
import socket
import sys
from time import sleep
from machine import Pin
from dht import DHT11


print("libray importted")
sensor = DHT11(Pin(21, Pin.IN, Pin.PULL_UP))
print("sensor set")

IP = "192.168.2.115"
PORT = 2431


def do_connect():
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        sta_if.active(True)
        sta_if.connect('<wifi SSID>>', '<wifi password>')
        while not sta_if.isconnected():
            pass

do_connect()
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
client.connect((IP, PORT))
client.setblocking(False)

while True:
    try:
        sensor.measure()  # Poll sensor
        t = sensor.temperature()
        h = sensor.humidity()
        if isinstance(t, int) and isinstance(h, int):  # Confirm sensor results
are numeric
            msg = (b'{0:3d},{1:3d}'.format(t, h)).encode('utf-8')
            client.send(msg)
            print(msg)
        else:
            print('Invalid sensor readings.')
    except KeyboardInterrupt:
        sys.exit()
    sleep(5)
```

Файл Qserverpy

```
import socket
HEADER_LENGTH=10
IP="192.168.2.110"
PORT=2431
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
s.bind((IP,PORT))
s.listen(5)
print("Server Is listening...")
clients_sockets,clients_add=s.accept()
print("Cleint connected")

def receive_message(client_sockets):
    try:
        message=client_sockets.recv(7)
        client_sockets.send(b'True')
        if len(message)==0:
            return False
        return message
    except:
        return False
while True:

    message=receive_message(clients_sockets)
    if message is not False:
        print(message)
    else:
        clients_sockets, clients_add = s.accept()
```

Файл configQ.py

```
import sys
if sys.version_info[0] == 2:
    from ConfigParser import ConfigParser
else:
    from configparser import ConfigParser
CONFIG_SECTION = 'jsonrpctcp'
# For encrypting / decrypting the data on keyed connections.
try:
    from Crypto.Cipher import AES
except:
    AES = None

class Config(object):
    """ Simple object to hold jsonrpctcp configuration options """
    _instance = None

    def __init__(self):
        """ The default values for the configuration. """
        self.timeout = 5 # seconds
        self.verbose = False
        self.buffer = 4096
        # 'secret' is used by server to indicate encryption --
        # if it is set, encryption is enabled.
        self.secret = None
        # 'crypt' can be any class anything that implements 'new' and
        # 'encrypt' / 'decrypt' like they Crypto ciphers.
        self.crypt = AES
        # 'crypt_chunk_size' is the size of the message chunk required
        # by the cipher.
        self.crypt_chunk_size = 16
        # Maximum number of queued connections
        self.max_queue = 10

    @classmethod
    def instance(cls):
        """ Retrieves singleton """
        if not cls._instance:
            cls._instance = cls()
        return cls._instance

    def load(self, path):
        """ Loads settings from a configuration file. """
        conf = ConfigParser()
        conf.read(path)
        if not conf.has_section(CONFIG_SECTION):
            return
        for option in conf.options(CONFIG_SECTION):
            value = conf.get(CONFIG_SECTION, option)
            orig_type = type(getattr(self, option, None))
            if type(value) is not orig_type and \
                orig_type is not type(None):
                # Doesn't work for False.
                value = orig_type(value)
            setattr(self, option, value)
```

Файл Qerrors.py

```
from jsonrpctcp import config
import random
import string
import sys
JSONRPC_ERRORS = {
    -32700: {'code':-32700, 'message':'Parse error.'},
    -32600: {'code':-32600, 'message':'Invalid request.'},
    -32601: {'code':-32601, 'message':'Method not found.'},
    -32602: {'code':-32602, 'message':'Invalid parameters.'},
    -32603: {'code':-32603, 'message':'Internal error.'},
}
# The random characters are used for padding the server error messages
# so that it will hopefully be  harder to brute-force a secret key.
if sys.version_info[0] == 2:
    RANDOM_CHARACTERS = string.letters + string.digits
else:
    RANDOM_CHARACTERS = string.ascii_letters + string.digits
RANDOM_STRING_LENGTH = 12

class ProtocolError(Exception):
    """ Used for system errors and custom errors. """

    def __init__(self, code, message=None, data=None):
        message = message or \
            JSONRPC_ERRORS.get(code, {}).get('message', 'Unknown error.')
        self.message = message
        self.code = code
        self.data = data

    def generate_error(self, *args, **kwargs):
        """
        Return a proper JSON-RPC structure for error messages.
        This also pads a random string on the message to help
        counter brute-forcing "known" messages.
        """
        message = self.message
        if config.secret:
            random_string = ''.join([
                random.choice(RANDOM_CHARACTERS)
                for i in range(RANDOM_STRING_LENGTH)
            ])
            message = '%s (random: %s)' % (self.message, random_string)
        response = {
            'jsonrpc':"2.0",
            'error': {
                'message': message,
                'code': self.code
            },
            'id':kwargs.get('id', None)
        }
        return response

    def __repr__(self):
        return (
            '<ProtocolError> code:%s, message:%s, data:%s' %
            (self.code, self.message, self.data)
        )
class EncryptionMissing(Exception):
    """ Simple exception if a crypt library is missing """
    pass
```

Файл Server-object.py

```
from __future__ import print_function
import threading
import socket
import time
import sys
import types
import traceback
from jsonrpctcp.handler import Handler
from jsonrpctcp import config
from jsonrpctcp import logger
from jsonrpctcp import history
from jsonrpctcp.errors import ProtocolError
from jsonrpctcp.errors import JSONRPC_ERRORS, EncryptionMissing
from inspect import isclass

try:
    import json
except ImportError:
    import simplejson as json

class Server(object):
    """
    This class is the basic Server object. It should be instantiated
    with a (host, port) tuple (and an optional handler), and then
    the Handler subclasses / functions should be attached through the
    add_handler method.
    """

    _shutdown = False

    def __init__(self, addr, handler=None, pool=10):
        if config.secret and not config.crypt:
            raise EncryptionMissing('No encrpytion library found.')
        self.addr = addr
        self.socket = None
        self.threads = []
        # Pool not actually implemented yet
        self.pool = pool
        self.json_request = JSONRequest(self)
        if handler:
            assert hasattr(handler, '__call__') or \
                issubclass(handler, Handler)
            self.json_request.add_handler(handler)

    def serve(self):
        """
        This starts the server -- it blocks, so if there are other
        tasks that need to be performed after the server is started,
        threading / multiprocessing will need to be employed.
        """
        self.socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.socket.bind(self.addr)
        self.socket.listen(config.max_queue)
        self.wait()

    def wait(self):
        """ The principle wait cycle. """
        while True:
            if self._shutdown:
```

```
                break
            clientsock, addr = self.socket.accept()
            args = (clientsock, addr)
            target = self.json_request.process
            thread = threading.Thread(target=target, args=args)
            thread.daemon = True
            thread.start()
            self.threads.append(thread)
            self.check_threads()

        sys.stdout.write('Shutting down...')
        for thread in self.threads:
            thread.join()
        sys.stdout.write('done.\n')

    def shutdown(self):
        """
        Attempts to shutdown the server.
        TODO: Make this work quickly and properly.
        """
        self._shutdown = True
        self.socket.close()

    def check_threads(self):
        """
        Check the thread list for dead threads and finished
        threads.
        """
        for thread in self.threads:
            if not thread.isAlive():
                thread.join()
                self.threads.remove(thread)

    def add_handler(self, method, name=None):
        """ Just a wrapper around JSONRequest.add_handler """
        self.json_request.add_handler(method, name)

class JSONRequest(object):
    """
    This is the class that handles individual requests passed
    from the server wait cycle.
    """

    def __init__(self, server):
        self.server = server
        self.handlers = {}

    def add_handler(self, method, name=None):
        """
        Attach a handler to the request object. It must be either
        callable, or a subclass of Handler.
        """
        if isclass(method):
            assert issubclass(method, Handler)
            # If it's an actual Handler subclass
            handler_instance = method(self)
            for hname, method in handler_instance._handlers.iteritems():
                if name:
                    hname = '%s.%s' % (name, hname)
                self.handlers[hname] = method
        else:
            if not name:
                name = method.__name__
```

```
            assert hasattr(method, '__call__')
            self.handlers[name] = method

    def get_handler(self, name):
        """ Check for an attached handler and return it. """
        if self.handlers.has_key(name):
            return self.handlers[name]
        return None

    def process(self, sock, addr):
        """ Just a wrapper for ProcessRequest. """
        request = ProcessRequest(self)
        request.process(sock, addr)

class ProcessRequest(object):
    """
    This is the class that handles an actual request, passing it through
    the Handler and parsing the response.
    """

    socket_error = False

    def __init__(self, json_request):
        self.json_request = json_request
        self.socket = None
        self.client_address = None

    def process(self, sock, addr):
        """
        Retrieves the data stream from the socket and validates it.
        """
        self.socket = sock
        self.socket.settimeout(config.timeout)
        self.client_address = addr
        requestlines = []
        while True:
            data = self.get_data()
            if not data:
                break
            requestlines.append(data)
            if len(data) < config.buffer:
                break
        request = ''.join(requestlines)
        response = ''
        crypt_error = False
        if config.secret:
            crypt = config.crypt.new(config.secret)
            try:
                request = crypt.decrypt(request)
            except ValueError:
                crypt_error = True
                error = ProtocolError(-32700, 'Could not decrypt request.')
                response = json.dumps(error.generate_error())
        history.request = request
        logger.debug('SERVER | REQUEST: %s' % request)
        if self.socket_error:
            self.socket.close()
        else:
            if not crypt_error:
                response = self.parse_request(request)
            history.response = response
            logger.debug('SERVER | RESPONSE: %s' % response)
            if config.secret:
```

```
                length = config.crypt_chunk_size
                pad_length = length - (len(response) % length)
                response = crypt.encrypt('%s%s' % (response, ' '*pad_length))
            self.socket.send(response)
        self.socket.close()

    def get_data(self):
        """ Retrieves a data chunk from the socket. """
        try:
            data = self.socket.recv(config.buffer)
        except socket.timeout:
            # It may have finished sending without an error if
            # len(message) % buffer == 0.
            data = None
        except socket.error:
            self.socket_error = True
            data = None
        return data

    def parse_request(self, data):
        """ Attempts to load the request, validates it, and calls it. """
        try:
            obj = json.loads(data)
        except ValueError:
            return json.dumps(ProtocolError(-32700).generate_error())
        if not obj:
            return json.dumps(ProtocolError(-32600).generate_error())
        batch = True
        if type(obj) is not list:
            batch = False
            obj = [obj,]
        responses = []
        for req in obj:
            request_error = ProtocolError(-32600)
            if type(req) is not dict:
                responses.append(request_error.generate_error())
            elif 'method' not in req.keys() or \
                type(req['method']) not in types.StringTypes:
                responses.append(request_error.generate_error())
            else:
                result = self.parse_call(req)
                if req.has_key('id'):
                    response = generate_response(result, id=req.get('id'))
                    responses.append(response)
        if not responses:
            # It's either a batch of notifications or a single
            # notification, so return nothing.
            return ''
        else:
            if not batch:
                # Single request
                responses = responses[0]
            return json.dumps(responses)

    def parse_call(self, obj):
        """
        Parses a JSON request.
        """

        # Get ID, Notification if None
        # This is actually incorrect, as IDs can be null by spec (rare)
        request_id = obj.get('id', None)
```

```
        # Check for required parameters
        jsonrpc = obj.get('jsonrpc', None)
        method = obj.get('method', None)
        if not jsonrpc or not method:
            return ProtocolError(-32600)

        # Validate parameters
        params = obj.get('params', [])
        if type(params) not in (list, dict):
            return ProtocolError(-32602)

        # Parse Request
        kwargs = {}
        if type(params) is dict:
            kwargs = params
            params = []
        handler = self.json_request.get_handler(method)
        error_code = None
        message = None
        if handler:
            try:
                response = handler(*params, **kwargs)
                return response
            except Exception:
                logger.error('Error calling handler %s' % method)
                message = traceback.format_exc().splitlines()[-1]
                error_code = -32603
        else:
            error_code = -32601
        return ProtocolError(error_code, message=message)

def generate_response(result, **kwargs):
    """
    TODO: Fix so that a request_id can be Null and not a Notification.
    """
    if type(result) is ProtocolError:
        return result.generate_error(**kwargs)
    else:
        response = {'jsonrpc':"2.0", "result":result}
        response.update(kwargs)
        return response

def start_server(host, port, handler):
    """
    Wrapper around Server that pre-threads it.
    """
    server = Server((host, port))
    server.add_handler(handler)
    server_thread = threading.Thread(target=server.serve)
    server_thread.daemon = True
    server_thread.start()
    return server

def test_server():
    """
    Creates a simple server to be tested against the test_client in
    the client module.
    """

    host, port = '', 8080

    def echo(message):
        """
```

```
        Test method, an example of how simple it *should* be.
        """
        return message

    def summation(*args):
        return sum(args)

    if '-v' in sys.argv:
        import logging
        config.verbose = True
        logger.addHandler(logging.StreamHandler())
        logger.setLevel(logging.DEBUG)

    server = Server((host, port))
    server.add_handler(echo)
    server.add_handler(echo, 'tree.echo')
    server.add_handler(summation, 'sum')
    server_thread = threading.Thread(target=server.serve)
    server_thread.daemon = True
    server_thread.start()

    print ("Server running: %s:%s" % (host, port))

    try:
        while True:
            time.sleep(0.5)
    except KeyboardInterrupt:
        print('Finished.')
        sys.exit()


if __name__ == "__main__":
    test_server()
```